Masaryk University
Faculty of Informatics

Cyber Situation Awareness via IP Flow Monitoring

Doctoral Thesis

RNDr. Tomáš Jirsík

Brno, Fall 2018

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

RNDr. Tomáš Jirsík

Advisor: doc. Ing. Pavel Čeleda, Ph.D.

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisors doc. RNDr. Václav Račanský and doc. Ing. Pavel Čeleda, Ph.D. for giving me the opportunity to join the cyber security team and for providing continuous support and guidance during my Ph.D. study and related research.

I would also like to thank my fellow labmates for the stimulating discussions, for sharing both times of despairs and joy, and for all fun, we have had during our collaboration. I am particularly grateful to Petr and Milan who were always there for me. Without their continuous feedback and motivation this thesis would not be possible.

Furthermore, I could have hardly finished this work without the selfless and relentless support of my wife Pavla, my daughter Gábi, my parents, sister, and other friends. They all showed a profound belief in my work. I owe them my eternal gratitude.

Abstract

Cyber situation awareness has been recognized as a vital requirement for effective cyber defense. The cyber situation awareness allows cybersecurity operators to identify, understand, and anticipate incoming threats. Achievement of the cyber situation awareness requires an ability to perceive the cyber environment, comprehend the processes in the environment, and to predict its future state based on the gained perception and comprehension. Achieving and maintaining the cyber situation awareness is a challenging task given the continuous evolution of the computer networks. As the computer network evolves, the methods for achieving the cyber situation awareness need to evolve as well. The perception of the computer network needs to keep up with increasing speed and scale of the networks. The large volume of data and information transferred over the networks and the network's dynamic open new challenges for the network comprehension and results in a necessity for advanced techniques for information retrieval. The prediction of the future state of a network demands novel approaches capable of dealing with the high dynamics and volatility of the networks.

This thesis contributes to the continuous evolution of cyber situation awareness by the research of novel approaches to the perception and comprehension of a computer network. We concentrate our research efforts on the domain of IP flow network monitoring. We propose improvements to the IP flow monitoring techniques that enable the enhanced perception of a computer network. Further, we conduct detailed analyses of network traffic, which allows for an in-depth understanding of host behavior in a computer network. Last, but not least, we propose a novel approach to IP flow network monitoring, that enables real-time cyber situation awareness.

The first contribution of this thesis is a systematic overview of the cyber situation awareness and IP flow network monitoring research fields. Apart from the definitions of basic terms, explanation of the fundamental concepts, and description of the current state-of-the-art, the overview includes a summary of contemporary challenges for both presented research fields. We conduct the synthesis of the challenges, and the selected challenges are then addressed through the entire thesis.

At the perception level, we increase the information value of the monitored data from network traffic by enriching IP flow records with information from the HTTP protocol. The side effects of the enrichment on the IP flow monitoring performance are explored to uncover the information – performance trade-off. Next, we inspect the means of information retrieval from the HTTPS protocol as the increasing share of encrypted network traffic is becoming a challenge for cyber situation awareness and network monitoring in general. We present a method that allows for a host identification in encrypted traffic that is built upon a dictionary of Cipher Suite – User Agents pairs.

At the comprehension level, we present the results of several network traffic measurements. First, we focus on the analysis of the Top N statistics. We describe its properties and carry out an evaluation that focuses on application to host identification tasks. Next, we explore the properties of IPv6 tunneling mechanisms. Our analysis reveals the main characteristics of the tunneled traffic such as HOP Limits distribution. Last, we investigate the available host-related information in network traffic. We innovate techniques for operating system fingerprinting in both static and dynamic networks and show possibilities of host identification in encrypted network traffic.

Our main contribution to cyber situation awareness is an application of stream-based data processing paradigm to the IP flow data processing. We research and design a concept for stream-based IP flow data analysis, and present consequences originating in the shift from the original batch-based approach. The stream-based IP flow monitoring allows for real-time monitoring of the computer network. The real-time network monitoring then provides real-time data to a security operator enabling him/her to achieve a real-time situation awareness.

We conclude our research by a proposal of a general approach to real-time cyber situation awareness based on the merge of a batch- and stream-based approaches. We also provide suggestions for future research opportunities.

Keywords

cyber situation awareness, IP flow, monitoring, network, real-time, host identification, perception, comprehension

Contents

1 Introduction

Computer networks have become an inherent part of our everyday life. They facilitate a great variety of services ranging from the ones critical for a society, such as control systems of power plants, communication systems, financial systems, over the ones that are useful in everyday life, such as information portals, social networks, and e-shops, to the ones that serve purely for entertainment, e.g., forums, games, video streaming platforms. As society has become increasingly dependent on these services, the computer network facilitating these services have become lucrative targets for various attackers and fraudsters. Protection of computer networks from harms caused by the attackers has become an important topic. An essential prerequisite for the adequate protection of a computer network is an ability to perceive the processes in the network, comprehend their meaning and relations, and predict the future state of a network – in other words, to gain a *cyber situation awareness*.

Cyber situation awareness is a concept that supports cybersecurity operators in cyber defense. The goal of this concept is to provide an operator with sufficient information that enables him/her to make an informed decision on network protection. The mainstream approach to cyber situation awareness defines three levels that form the awareness: *perception*, *comprehension*, and *prediction*. The perception level serves to obtain data from a monitored environment, the comprehension level aims to understand the information contained in the data, and the goal of the prediction level is to predict the future state of the environment.

Computer networks are dynamic environments that evolve at a high pace. New technologies are continuously deployed, the paradigms of service and network usage changes in time, and new misuses and attacks are produced rapidly. Such a rapid development requires that cyber situation awareness keeps pace with the evolution of computer networks. Techniques for perception need to be updated to be able to obtain data from new protocols, new data processing and analysis methods need to be developed to process and comprehend large volumes of data, and methods with improved forecast accuracy need to be researched to model the future state of a network.

IP flow monitoring represents a widely used approach for obtaining network visibility. It processes information from packet headers and aggregates the packets into connection-like records, so-called IP flows. Compared to other methods of network monitoring, e.g., deep packet inspection, the IP flow monitoring enables to process a significantly larger volume of network traffic at higher speeds in exchange for the lower amount of information processed. Due to its properties, mainly the ability to provide a holistic view of a computer network, IP flow monitoring is frequently used for the perception of information from computer networks. Moreover, analysis of the collected IP flows serves well for network comprehension, and even for prediction of the future state of a network, if relevant techniques are applied.

Continuous development of computer networks and the evolution of their usage imposed several challenges and open issues that are shared by both IP flow monitoring and cyber situation awareness. Since IP flow monitoring serves as an approach to obtain cyber situation awareness, resolving an open issue of IP flow monitoring can then improve the overall cyber situation awareness as well. This thesis aims to address the current challenges of IP flow monitoring that lead to enhancement of the cyber situation awareness. We propose improvements of IP flow monitoring that enable a better perception of computer networks and provide the results of several analyses that bring light into network processes and allow for deeper network comprehension.

1.1 Problem Statement

IP flow monitoring was originally designed for network accounting and profiling. The accounting function of IP flow has determined the design of IP flow monitoring infrastructures. IP flows and associated IP flow monitoring infrastructures are designed to provide a holistic view on a network with the focus on regular network reporting. The original design and purpose of IP flow, however, need to be continuously transformed from the original to a modern one due to technological advances in networking area (faster networks, SDN), the shift in protocol usage (rise of encrypted traffic), and new possibilities and paradigms for data processing in general. The IP flow definitions, monitoring infrastructure, and related toolset, however, remain with the original purpose, which introduces discrepancies in IP flow applications and other issues in flow processing and analyses. The utilization of the IP flow monitoring in the cyber situation awareness domain introduces additional challenges that need to be addressed. The holistic view on the network is not sufficient anymore, as a detailed micro view that provides information of all entities in a network is demanded.

We have identified three main open issues of IP flow network monitoring that affects the cyber situation awareness. First, the IP flow monitoring needs to provide sufficient visibility and information for effective cyber defense. Since the sophistication of the current attacks increases and the cybercriminals form professionalized groups using advanced tools, the information value of the exported IP flows is not sufficient anymore and needs to be increased. Second, the IP flow monitoring is designed for holistic observation of a network from a connection perspective, not for monitoring hosts in a network. Cyber situation awareness would benefit from having the host-oriented view on the network. Identification of a host from IP flow records is a challenging task, though, due to dynamic addressing in a network and increasing share of encrypted traffic. Third, requirements for high analysis speed and real-time network monitoring were not included in the design of the traditional IP flow monitoring infrastructure. Nowadays, the real-time view of a network has become essential as the costs of service downtime rises. Any delays in attack detection or identification of the cause of the downtime are costly and undesirable. However, current IP flow monitoring architectures introduce several delays that slow down the process of obtaining the cyber situation awareness and so does not allow for an instant response.

1.1.1 Lack of Network Visibility and Comprehension

When we started our research in 2013, the IP flow monitoring provided visibility only into the network and transport layers except for Flexible NetFlow that used NBAR for application recognition. However, the used application recognition only provided the type of an application protocol. No additional application data were available. The data from the application layer represents an information-rich source that enables advanced analysis and detection of network threats targeting applications. The absence of application-aware IP flow monitoring prevents us from obtaining the sufficient level of cyber situation awareness, e.g., to be aware of an ongoing application network attack. Therefore, we found it essential to develop an application-aware

IP flow monitoring that would allow for network-wide cyber situation perception even at the application level.

The addition of the application information to the IP flow monitoring increases the volume of the information available on the one hand. On the other hand, it could lead to a performance decline as additional data needs to be retrieved from packets and processed. We believe that the possible performance drop need to be investigated as it could make IP flow monitoring ineffective considering the increasing speed of network traffic and the fact, that ability to process a large volume of network traffic at high speed is the main advantage of IP flow monitoring. Hence, the trade-off between the increased information value of IP flows and the associated decrease of IP flow monitoring performance should be studied. Moreover, the availability of the information for the application layer does not imply the comprehension of the application information straightforward. With new information, additional analyses and measurements need to be conducted to reveal the nature and relations present in the application data.

1.1.2 Host Identification in Network Traffic

IP flow monitoring is generally valued for its ability to provide a holistic view of a network. The holistic view is insufficient, though, if the IP flow monitoring is applied in the cyber situation awareness domain. A deep cyber situation awareness requires a more detailed view of a network than the holistic one. The detailed view should present information both on the whole network and on all hosts in the network at the same time. IP flow monitoring results are usually presented from the network line viewpoint, e.g., how much traffic is transferred over a line. A host view on a network, e.g., a volume of outcoming traffic computed for each host in a network, is usually not available. Nevertheless, the IP flow monitoring can provide information on hosts, as IP flow records represent connections between hosts observed in a line in the network. If conveniently transformed, the IP flow records can provide all statistics that are usually computed for a network line and a host as well.

The comprehension of host information from the transformed IP flow records is hindered by the fact that a host is represented by an IP address in an IP flow record. A single IP address can represent more hosts in a network. Dynamically addressed networks can assign a different IP address to a host each time it connects to the network. Moreover, network translation devices (NATs) are represented by a single IP address in IP flow records despite of the fact that they represent more hosts. The fact that there is not only one-to-one host–IP address relation prevents the correct assignment of relevant information from IP flow records to a host. Auxiliary methods for host identification need to be explored so that we do not need to rely only on IP address as a host's identifier. Several host's characteristics can be observed in network traffic. If correctly defined, these characteristics could create a host's fingerprint that could provide support for IP address based host identification. The recent rise of the share of encrypted network traffic, however, makes the identification of these characteristics impossible at the application layer. Alternative approaches for host identification in encrypted traffic should be studied to be able to identify hosts and to obtain a complete view of a network.

1.1.3 Delays in IP Flow Monitoring Workflow

As the costs of a service downtime increase, any delay present in the perception, comprehension, or prediction of the computer network that postpones the achievement of the cyber situation awareness and reduces response speed and capabilities is unwelcome. Network IP flow monitoring includes several delays by design, however. The first delay occurs during the phase of IP flow creation as the IP flow export is determined by timeouts in some cases. Next delay is introduced during IP flow record analysis. The IP flow records are analyzed in batches which requires to wait until a batch is complete before it enters the analysis. The combined delay of the network IP flow monitoring workflow can reach over several minutes, which can lead to substantial financial losses due to late response to an attack.

We believe that novel approaches to IP flow metering and analysis need to be researched to obtain a real-time cyber situation awareness and instant response and recovery. The delays presented in the IP flow monitoring should be reduced. We cannot reduce the delay without a significant redesign of the IP flow monitoring workflow. It is essential to keep in mind that the redesign of IP flow monitoring workflow changes the nature of the generated IP flow records. Hence, apart from the redesign of the IP flow monitoring process, also the approach to IP flow analysis needs to be revised to provide real-time cyber situation awareness.

1.2 Research Goals

The previous section highlights the current problems of IP flow monitoring that influence the cyber situation awareness. These problems are addressed in this thesis and problems can be summarized into the following main objective of this thesis:

> *Investigate how IP flow monitoring can be improved to enhance the cyber situation awareness.*

In other words, the objective of this thesis is to propose improvements to the network IP flow monitoring that enhance the perception and comprehension of a computer network. The prediction stage of cyber situation awareness is out of the scope of this thesis and is left for future research.

In light of the main objective of this thesis, we identify the following research goals (RG) that are motivated by the discovered research problems:

RG1: Propose and evaluate IP flow monitoring methods that enhance network perception and comprehension and respond to the emerging trends in the cyber situation awareness and the IP flow monitoring.

RG2: Develop methods for host identification in both unencrypted and encrypted network traffic.

RG3: Provide an option for reducing the delays in the network IP flow monitoring workflow leading to the real-time cyber situation awareness.

The identified research goals have been addressed since 2013. Figure 1 presents the research timeline to display when we addressed our research goals along with relevant author's publications. So that the novelty and contribution of our research results can be assessed relative to a time they were delivered.

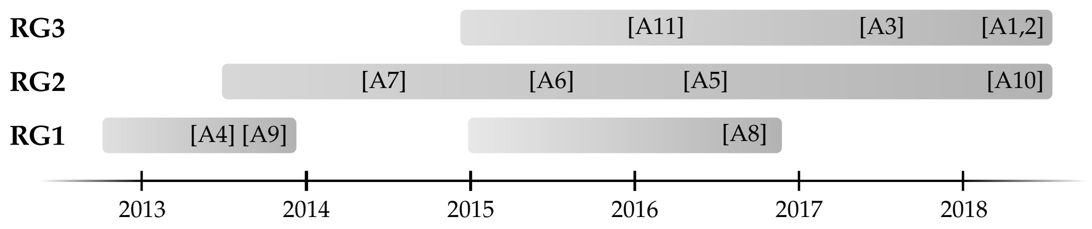


Figure 1: Timeline of the presented research.

1.3 Contributions

In the course of our research, the following original contributions to the state-of-the-art were made in the area of cyber situation awareness and IP flow monitoring:

1. *An overview of the cyber situation awareness and summary of its open issues.* An application of situation awareness to the cyber domain is not straightforward. We provide an overview of the main approaches to the situation awareness and show an application of the selected approach to the cyber domain. The overview summarizes the current open issues of the cyber situation awareness. **(Chapter 2)**

2. *An overview of the network IP flow monitoring and its use in the cyber situation awareness.* We describe the main ideas of the network IP flow monitoring. We extend the overview with an additional analysis of the IP flow monitoring workflow that focuses on the time aspects of the workflow. We identify current trends and open issues in this domain and present an application of the IP flow monitoring in the cyber situation awareness domain including the synthesis of the open issues. **(Chapter 3)**

3. *Evaluation of the impact of extending IP flow records with application layer information.* We compare contemporary parsers for application protocols and design flex-based parser for HTTP protocol. We evaluate the performance of the application layer IP flow probe and demonstrate the effects of the addition of HTTP information to IP flow records on the throughput of the IP flow probe. We show that the performance decrease depends on the distribution of network traffic and it is nearly independent of the number of parsed application layer fields. **(Chapter 4)**

4. *Creation of a dictionary for host identification in encrypted traffic.* We analyze encrypted network traffic, namely HTTPS – an HTTP protocol over SSL/TLS. Based on the analysis, we propose a dictionary of Cipher Suite Lists and User-Agents pairs that can provide additional information on a host communicating via encrypted traffic. We determine the coverage of the dictionary in the network traffic and the cardinality of the Cipher Suite Lists–User-Agent relations. **(Chapter 4)**

5. *Evaluation of the Top N statistics.* We research the characteristics of the Top N statistics and display information that can be retrieved from this statistics. We evaluate the statistics with regard to its availability, uniqueness of the information, and time stability – characteristics that are fundamental for comprehension of host behavior. We show, that Top N statistic's stability depends on the aggregation window and that HTTP based Top N statistics can identify a host in network traffic with 59.5 % true positive rate. **(Chapter 5)**

6. *Analysis of transition mechanisms for IPv6 tunneling.* We provide a coherent investigation of IPv6 network traffic tunneled via Teredo and 6to4 transition mechanisms. On a data gathered from real-world measurement, we explore the Time to Live distributions both of the tunneling and tunneled network traffic, IP flow characteristics, and location of Teredo endpoint servers. The results of the exploration enable a deeper comprehension of the behavior transition mechanisms for IPv6 in the real-world networks. **(Chapter 5)**

7. *Novel approaches to host-based information retrieval from network traffic.* We propose several enhancements for the retrieval of host-based information from IP flow records used for host identification. We analyze retrieval of operating system specific information from both static and dynamic networks and demonstrate the possibilities of the information retrieval from encrypted traffic. **(Chapter 5)**

8. *Real-time IP flow monitoring and cyber situation awareness.* We propose an approach for real-time IP flow monitoring that is based on distributed data stream processing and reduces the delays present in IP flow monitoring workflow. We develop a prototype framework *Stream4Flow* and discuss the implications of the usage of the stream-based approach for IP flow record analysis. We propose a next-generation IP flow monitoring that combines the advantages of stream-based IP flow monitoring with the advantages of the traditional batch-based approach. We show, that the next-generation IP flow monitoring infrastructure can be used, if slightly modified, to obtain the real-time cyber situation awareness. **(Chapter 6)**

1.4 Thesis Outline

This thesis is organized to reflect our contributions to the cyber situation awareness. Figure 2 depicts the thesis structure. The chapters are grouped according to their topic from the cyber situation awareness field. Where relevant, we highlight sections and present their relation to the defined research goals.

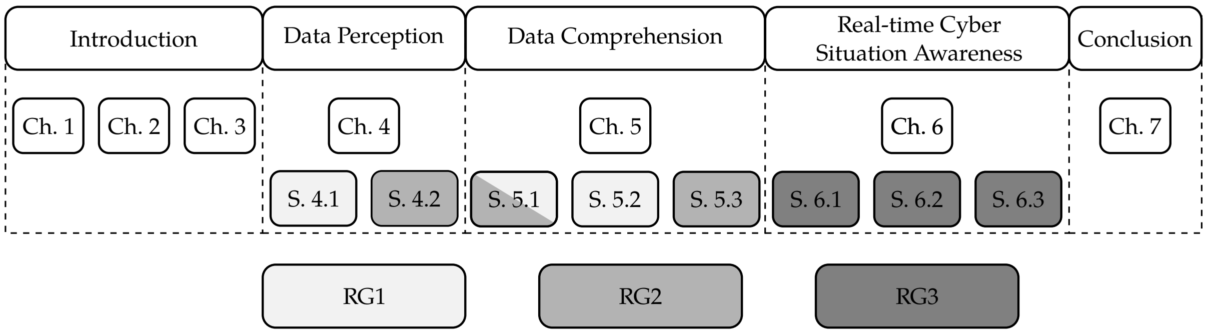


Figure 2: Thesis structure.

The remainder of this thesis is organized as follows:

- **Chapter 2 – Cyber Situation Awareness** discusses three main models for situation awareness. The Endsley's three-level model of situation awareness is then applied to the cyber security domain. We discuss the specifics of this application, present relevant state-of-the-art, and identify current open issues.

- **Chapter 3 – IP Flow Network Traffic Monitoring** presents a comprehensive tutorial on IP flow monitoring. We present both IP flow monitoring and IP flow record analysis and identify their open issues. Further, we show the network IP flow monitoring in the context of cyber situation awareness and revise and merge the current open issues from both research areas.

- **Chapter 4 – Data Perception** presents our contribution to network perception. We investigate how the addition of an application-level information into IP flow influences the performance of IP flow probes, and focus on improvement of data collection for client identification from encrypted network traffic.

- **Chapter 5 – Data Comprehension** displays our research effort in the following three areas: evaluation of properties of Top N statistics, analysis of tunneled network traffic, and retrieval of host-related information.

- **Chapter 6 – Towards Real-time Cyber Situation Awareness** describes our contribution to the improvement of the speed of analysis process in the cyber situation awareness. We reduce the time needed for IP flow analysis by introducing a stream-based IP flow monitoring, present next-generation IP flow monitoring workflow, and propose an approach how to gain real-time cyber situation awareness.

- **Chapter 7 – Conclusion** summarizes our achieved results and elaborates on various directions for future research.

1.5 Further Remarks

The observed network traffic contains privacy-sensitive information. Hereby, we declare that the monitored data used for our research were processed in accordance with the EU General Data Protection Regulation 2016/679. The monitored data were collected for specified purposes, and the appropriate technical and organizational measures were taken to safeguard the rights of the data subjects. We processed the data in a manner that ensured appropriate security of the data including protection against unauthorized or unlawful processing, accidental loss, destruction or damage. We implemented appropriate technical and organizational measures to ensure a level of security appropriate to the risk, including the pseudonymization and encryption of the data, assurance of confidentiality, integrity, availability, and resilience of data processing systems. If used, personal data were processed in a manner that ensured appropriate security and confidentiality of the personal data, including preventing unauthorized access to or use of personal data and the equipment used for the processing.

We are aware that pursuing the RG 2 can lead to the development of new techniques, that enable identification of a device in network traffic. We develop these techniques to serve for network management and security, if necessary. However, considering the purpose of these techniques, they can be naturally misused for tracking and spying on a device or a person. We are aware of this fact. Nevertheless, we still believe, that the overall benefits for the society that brings our research when used for intended, lawful purposes, outweigh the costs of the misuses of our results.

2
Cyber Situation Awareness

Cyber situation awareness is an essential part of cyber defense processes. As computer networks and systems continue to increase in complexity and sophistication, the requirements and demands on a cybersecurity operator increase as well. A concept of cyber situation awareness aims to provide an operator with a coherent methodology to cope with the network's and system's complexity, gather all necessary information, and to comprehend underlying processes in these systems. Developing a cyber situation awareness, an operator is capable of making strategic decisions even in case of complex and sophisticated systems.

This chapter aims to provide a basic introduction to cyber situation awareness to an unfamiliar reader. Situation awareness is a broad research field covering numerous approaches and applications in various areas. Its application to the cyber domain is not straightforward, though. In the first part of the chapter, we offer a brief overview of situation awareness to present fundamental ideas and basic terminology. We provide relevant definitions, outline history, and introduce its essential aspects relevant to this thesis. The purpose of the overview is to provide basic general concepts that we further narrow to an application in the cyber domain. Application to the cyber domain using Endsley's three-level definition is described in the rest of the chapter. We discuss several viewpoints on cyber situation awareness and outline specifics of the application in the cyber domain. A literature review is presented to display state-of-the-art and current research directions in the cyber situation awareness area. Last, but not least, we list open issues of cyber situation awareness. These issues will be addressed throughout this thesis.

Relevant author's publication to this chapter is [A1, A2].

This chapter is structured as follows:

- *Section 2.1 presents a basics concepts and definitions of situation awareness.*
- *Section 2.2 discusses an application of situation awareness in the cyber domain, presents a literature review, and identifies open issues.*

2.1 Situation Awareness

2.1.1 Origins

Situation awareness (SA) has always been in need in everyday day life. A prehistoric hunter undoubtedly needed to capture and understand various inputs from his environment to efficiently hunt down a pray bigger and stronger than him, and not to become a pray himself at the same time. For many years, SA principles were used in everyday life intuitively. They were developed mainly by learning and experiences. However, the intuitive SA began insufficient as the technology improved and the complexity of the world increased. People had to start using SA consciously.

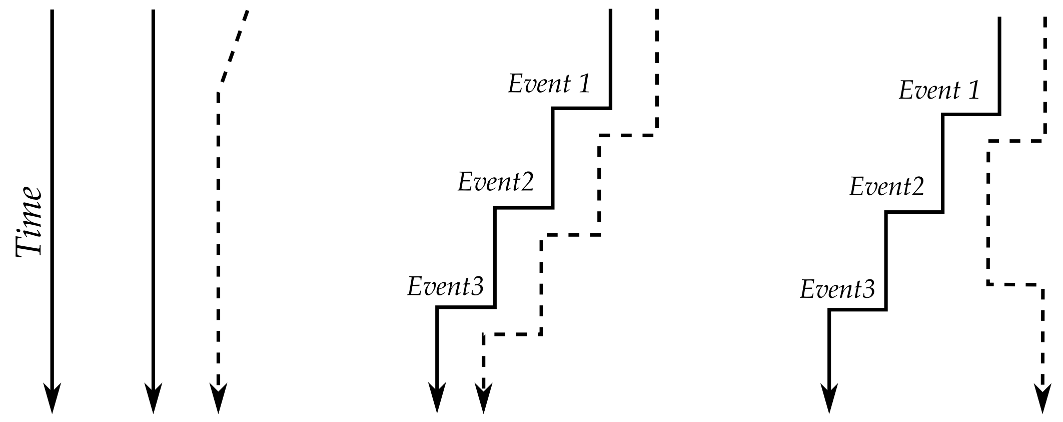
The first formulation of a concept that can be referred to as situation awareness dates back to the beginning of the 20th century. During World War I, Oswald Boelke realized the importance of "gaining an awareness of the enemy before the enemy gained a similar awareness and devise methods for accomplishing this" [1]. A famous pilot of WWI Red Baron is reported to have an ability to achieve and maintain the situation awareness during air combats, which made him successful in many dogfights. He is said to be able to give accurate re-creations and critiques of his fellow pilots even though he was fighting hundreds of meters away [1].

After World War I and II, the concept of SA did not receive too much attention in academic or technical literature till late 1980s [2]. The first push for the research came from the aviation domain. In the aviation domain, there is a continuous pressure on pilots and air traffic controllers to process and understand much information from various sources to retrieve the flight status. To ease the pressure, advanced applications of flight deck automation were introduced. However, the introduction of automation had a negative effect on pilots. The automation systems were no longer optimized for human operation and even overstepped the human's capability to keep track of the current situation in some cases [3]. The idea of separation between the human operators understanding of system status and actual system status emerged and became a crux of the definition of SA [4].

In the late 1980's, a temporal dimension of Situation Awareness has been introduced. It was observed, that for people to maintain adequate awareness of system status, it is necessary to track the development of the events. The incidents evolve and propagate over time. If a human operator fails to follow and adapt to a new system state, it may lead to misunderstanding of the systems status and wrong situation assessment as depicted in Figure 3. Decisions made on the basis of incorrect situation assessment could potentially lead to even worse incident, e.g., events preceding the Chernobyl explosion [5].

The conceptual basis SA had been cloudy before 1990, though. The main theoretic foundations were laid during the last decade of the 20th century. During the decade, underlying theoretical works were published by Endsley (1995) [6], Smith and Hancock (1995) [7], and Bedny and Meister (1999) [8]. The research of SA earned considerable attention in the literature, e.g., special issue on SA published in Human Factors Journal in 1995. Although SA being conceptually defined, it still met with a fair amount of criticism claiming that SA is "too subjective phenomenon to be measured objectively" [3]. Nevertheless, SA represented promising, and useful concept with considerable significance for operational research and thus remained a worthy research topic. Approaches to objective SA measurement were discussed and introduced (e.g., in [9]), and researchers kept their interest in the topic.

Since then, the application of SA concept has quickly spread to other domains than original aviation one. The most prominent driver of this widespread has been the technology. The complexity of systems and their automation increase rapidly due to introduced technologies, e.g., robotics, SCADA systems, and so forth. As a result of the technological progress, the dynamics of systems also increases. The automation of these areas has moved an operator from an active role of searching information in a decision process to a decision maker that consumes informa-



(a) *Unreal evolution of incident. Incident does not evolve, and an operator has a single assessment and execution.*

(b) *Real incident evolution. An operator needs to track the development to maintain Situation Awareness.*

(c) *Real incident evolution. An incident occurs, and operators fail to track event which leads to situation misinterpretation.*

Figure 3: Temporal dimension of situation awareness *(adapted from [4]).*

tion from systems and based on the information formulates his decision. SA is being introduced to various areas (power grids, strategic tactical systems, medicine, and also cyberspace) to ensure, that an operator assesses a situation correctly based on received data and can make an informed decision.

2.1.2 Definitions

Three main definitions dominate the Situation(al) Awareness[1] [2]. All three definitions are underpinned by more general models of human cognitive functioning. We present all three definitions and describe the main conceptual differences in the definitions. Particular attention is devoted to the Endsley's definition of SA as her's definition is widely adopted in the literature and serves as a base ground for further research in cyber situation awareness.

The Perceptual Cycle Definition

> *Situation Awareness is the invariant in the agent-environment system that generates the momentary knowledge and behavior required to attain the goals specified by an arbiter of performance in the environment.*
>
> Smith and Hancock, 1995 [2]

The authors of [2] take SA as adaptive externally directed consciousness. The SA is neither resided in the environment nor the person, but it resides through an interaction of a person with an environment. The definition characterizes SA in terms of perception-action cycle introduced by Neisser in [10]. The perception-action cycle puts forward a view of how human thought is closely connected to personal interaction with the world as depicted in Figure 4. The interactions go around the depicted cycle. Starting at the top, the environment informs an agent modifying its knowledge. The knowledge determines an agent's actions in the environment. The actions samples and eventually changes the environment which in turn informs the agent and so forth. The invariant at the core of the cycle specifies the agent's adaptation to its environment. The authors of the definition claim that the informed directed actions capture the essence of the behavior characteristics of SA.

1. Both terms "Situation Awareness" and "Situational Awareness" are used in literature. These terms appear interchangeable. For the sake of simplicity, the term Situation Awareness has been adopted in this thesis.

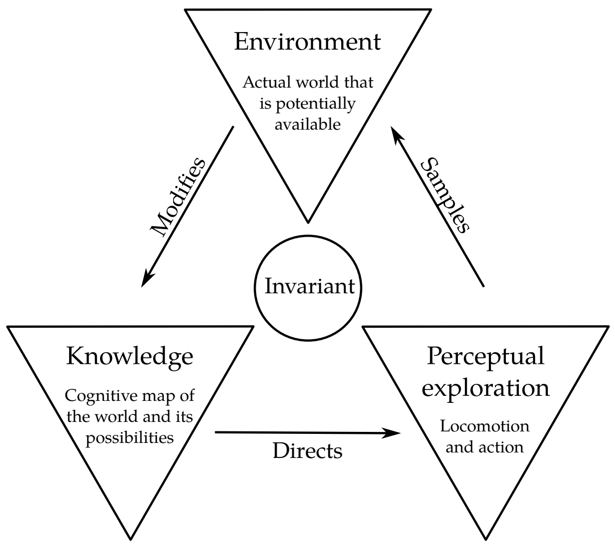


Figure 4: Perception-action cycle *(adapted from [2, 10]*).

To be able to quantify the SA, authors of the definition relate an agent with goals, behavior, and knowledge to match agent's performance specified by dicta from its environment. Unless an external goal and criteria for achieving it are defined, examination of lesser or greater degrees of SA is impossible. The degree an agent reach in meeting the expectations in fulfilling the goals given by an external arbiter of performance then serves as a mean for quantification of SA degree.

The SA definition by Smith and Hancock is suitable for explaining the dynamic aspect of SA, such as how the momentary knowledge is updated and how the search for information is initiated. The definition provides a high-level overview of a person interacting with an environment. This view can be used to explain human information processing in, e.g., control rooms of power plants, or in the cockpit of an air fighter.

The Interactive Sub-system Definition

Situation Awareness is the conscious dynamic reflection on the situation by an individual. It provides dynamic orientation to the situation, the opportunity to reflect not only the past, present, and future, but the potential features of the situation. The dynamic reflection contains logical-conceptual, imaginative conscious and unconscious components which enable individuals to develop mental models of external events.

Bedny and Meister, 1999 [8]

The definition by Bedny and Meister [8] presents the SA as one of many components of a reflective-orientational activity in the theory of activity [2]. The activity theory explains the human-psychological process of decision making in a system in a theoretical way. According to the theory, an activity can be psychological and internal or practical and external. An activity serves to achieve a given goal. The activity used to achieve a goal can change during the reaching the goal due to increasing knowledge of the situation. Further, within the situation, the actors can develop their own goals based on their past experience and knowledge. Moreover, authors claim that an objectively given goal is interpreted subjectively, so the perceived goal is not always synchronized with the given goal.

The activities performed during situation assessment according to the theory of activity are depicted in Figure 5. As stated in [8] and interpreted by [2], new information is supplied to function block ① and is interpreted using both the individuals conceptual models of the world

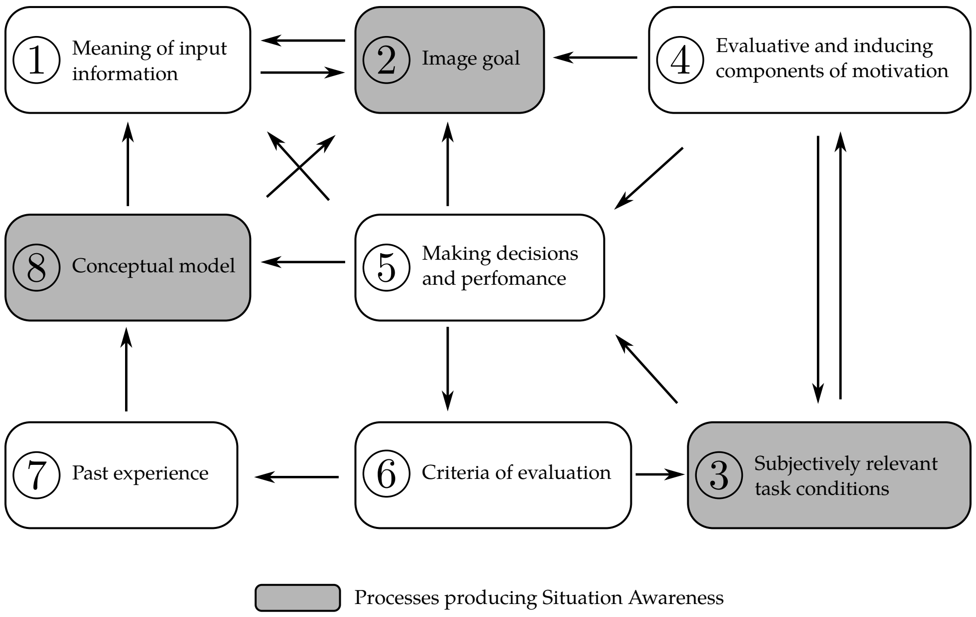

① Meaning of input information
② Image goal
④ Evaluative and inducing components of motivation
⑧ Conceptual model
⑤ Making decisions and perfomance
⑦ Past experience
⑥ Criteria of evaluation
③ Subjectively relevant task conditions
Processes producing Situation Awareness

Figure 5: Activity theory situation assessment.

(function block ⑧) highly dependent on past experience (function block ⑦) and images of the purpose of the task goals (function block ②) along with their orientation about what type of activity is required (function block ⑤). This interpretation informs the person's *pure* image of the task goals (function block ②). The individual then determines which world features are pertinent in function block ③ based on the significance and motivation toward task goals (function block ④) as well as their engagement with the world (function block ⑤). The extent to which they engage the task goals are influenced by the criteria of evaluation (functional block ⑥), and the current state of the world (function block ③). The outcome of this evaluation directs performance and the person's engagement with the world (function block ⑤) from which further criteria are developed (function block ⑥). Interaction with the world is stored in past experience (functional block ⑦). Authors of [8] state that processes forming situation awareness are the combination of conceptual model (function block ⑧), the image-goal (function block ②) and the subjective task conditions (functional block ③).

The interactive sub-system definition of SA is suitable when considering underlying functions and how they interact [2]. This view focuses on the processes that are used by an individual during situation assessment.

Three-level Model Definition

> *Situation Awareness is the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future.*
>
> Endsley, 1988 [11]

Definition of SA by Endsley has been widely accepted and is used in a great variety of functional areas, for example in medicine or vehicle operations [12]. Endsley apprehends SA as a state of knowledge and distinguishes it from the processes used to achieve the state. The processes for achieving SA varies across contexts, and she refers them to as *situation assessment*. The SA is also distinguished from subsequent processes, e.g., decision making.

The definition consists of three ascending primary components referred to as *levels* (see Figure 6). The levels should not be taken as definitive categories of SA into which it should be

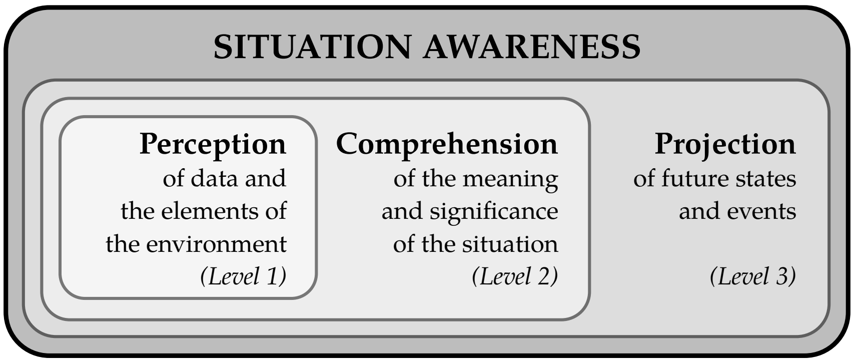

SITUATION AWARENESS
Perception
of data and the elements of the environment
(Level 1)
Comprehension
of the meaning and significance of the situation
(Level 2)
Projection
of future states and events
(Level 3)

Figure 6: Three-level model *(adapted from [6])*.

possible to divide SA of any environment. The levels serve as general descriptions that help in thinking about SA [12].

Level 1: Perception of the elements in the environment

Perception level is the lowest level of SA. At this level, status attributes and dynamics of relevant attributes of an environment are perceived. [11] Attributes of an environment can be perceived both directly from the environment (e.g., a pilot sees a mountain, a driver hears a siren of a firetruck) or indirectly using sensors (e.g., outside temperature, distance from obstacles). In both cases, the correct perception is crucial for the outcome SA. If the data obtained at this level are biased, inaccurate, or misleading, operator gains notion of SA that does not comply with the real world state. Without the correct basic perception of relevant information, the odds of forming an incorrect picture of the situation increases dramatically [13].

No interpretation of the acquired data is performed at this stage. All interpretation is left to the next level. This level is intended to represent to initial reception of information in the raw form. The separation of data capture and data comprehension and interpretation allows identifying discrepancies that might occur when requiring data from senses and sensors. One such a problem solved in SA is the data overload. [14] An operator is overwhelmed with sensors and data, and relevant information is not perceived. Jones et al. [15] found that 76 % of SA errors in pilots could be caused by problems in the perception of relevant information (either failure of the sensors or problems with the cognitive process).

Level 2: Comprehension of the current situation

Situation Awareness goes, however, beyond the simple perception of the data. Its goal is not only to percept the situation but to understand the situation correctly. The comprehension of a situation is based on a synthesis of disjoint elements perceived at Level 1 [11]. The elements are combined, interpreted, assigned with significance concerning given goals, and combined with knowledge of an operator to form a holistic picture of an environment. The difference between the levels is analogous to having a reading comprehension (Level 2) and reading individual words (Level 1). For example, an operator of an oil platform needs to put together pieces of information to derive an actual status of the platform's systems.

An operator's *expertise* is a vital requirement at this level. At the perception level, a novice and an experienced operator would reach the same results in situation assessment. At the comprehension level, a lack of expertise would cause novice's inability to follow basic lines of search coherently for further information or to misinterpretation of current situation [3]. An experienced operator is provided with long-term memory knowledge and has developed a repertoire of pattern-oriented representations of various situations which allow him/she to comprehend the situation more effectively than a novice operator can.

Another vital aspect of the comprehension of the situation is that meaning of the information needs to be considered not only in the sense of element interpretation (awareness) but also in the sense of objective importance with respect to given goals (situation) [16]. A person with situation comprehension can derive operationally relevant conclusions from the perceived data [13]. This level is error-prone; 20 % of SA errors were found to involve problems with situation comprehension [15].

Level 3: Projection of future status

The projection of future status is the highest level of SA. This level is the basis for *"being ahead of the plane"* [3]. In the majority of fields, where SA is of importance, experienced operators rely heavily on future projection. The future projection based on current events and dynamics enables them to anticipate future events and their implications which allows them timely decision making [13]. For example, an experienced driver differs from a common driver. Unlike a common driver, the experienced driver can detect possible future traffic and so prevent from collisions more efficiently.

The projection builds upon knowledge gained at previous two levels. The accuracy of the projection depends on the accuracy of the situation perception and comprehension. An inaccurate perception or comprehension leads to biased or misleading future projection. The accuracy of the prediction is further influenced by the operator's experience and knowledge of the situation dynamics.

Summary

The presented definitions can be linked together as depicted in Figure 7. The elements from the world are observed and internal representation of the world is created. The mental models and knowledge are then combined with the perceived data. Finally, reflection and projection are in place. Endsley's definition focuses mainly on perception and understanding of the world with some extend to projection. In contrast, Smith & Hancock present the SA in terms of the interaction between an individual and outer world and focus on the way, in which these two elements interact together. Bedny and Meister stress the reflective aspect of SA. They highlight the relation of mental models with the real world [2].

In general, the Endsley's definition has outperformed the other definitions in the number of citations and is generally adopted by the community. Being general enough, the definition has been without any significant changes extended to many other areas including the cybersecurity one.

2.1.3 Aspects of Situation Awareness

This subsection presents various aspects of SA that are frequently discussed in the literature. The aspects mainly refer to the three-level definition of SA. These aspects are mentioned as they help to clarify the main implications and consequences of the definition. A misunderstanding of these aspects may lead to the incorrect notion of SA.

Product vs Process

There is a process-product dichotomy which is also reflected in the different approach to SA of the described definitions. The definitions are either concerned with the process of acquiring the SA or are mainly concerned with the product of SA [2]. The interactive sub-system definition and perceptual model definition mainly focus on the process of acquiring the SA whereas three-level definition addresses the SA as a product.

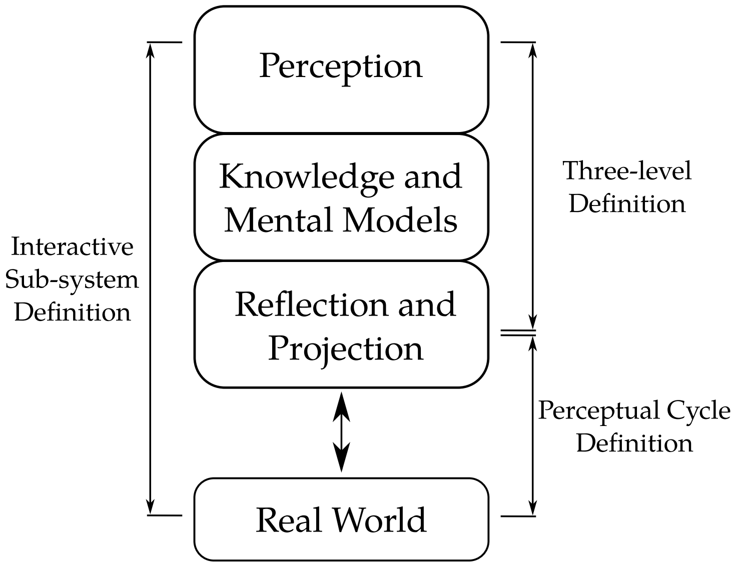

Perception
Knowledge and Mental Models
Reflection and Projection
Real World
Interactive Sub-system Definition
Three-level Definition
Perceptual Cycle Definition

Figure 7: Integration of SA definitions *(adapted from [2])*.

There is a benefit from such a differentiation between the process and product approach. It allows us to define the scope and coverage of the term "situation awareness" clearly and reduces the misunderstanding of the term. For example, three-level definition distinguishes the term *situation awareness* as a state of knowledge from processes used to achieve this status. These processes are referred to as *situation assessment*. The situation assessment may vary across application domain and context. This kind of approach enables to discuss different approaches to situation assessment across various applications leaving the definition of situation awareness general and universal for all application areas.

Time Aspects

The time plays a vital role in the definition of SA. The time aspect enables us to capture the dynamics of the environment. The SA definitions that represent the process approach address the time aspect naturally. The three-level definition covers the time aspect by the *"within a volume of time"* statement. This statement contained in the definition pertains to the fact that operators need to capture the environment not only in terms of volume (where, how much elements is present) but also in time (i.e., how will the environment evolve and what impact it will have on the operator's goal and tasks). Time is a substantial part of the Level 2 and Level 3 of the three-level definition [13].

The SA is not necessarily a product/process that is acquired/finished instantaneously. It is built over time. Thus, it is essential to take into account, that some aspects of the SA can be acquired only over time. Such a piece of knowledge could then be used for better environment perception or more accurate projection of future status. In this context, Endsley [6] introduced two following two terms: working memory and long-term memory.

Working memory stores the perceived information from the environment. In case an operator does not have any previous information on the environment, the majority of information is stored in working memory. Apart from perceived information, the working memory contains all necessary information needed for actual SA - mental models recalled from long-term memory, subsequent decisions, or current goals. All information is processed there and a picture of the current situation including the prediction of the future environment status. It requires the maintenance of present conditions, rules used for prediction, and action resulting from the predictions. All these tasks impose a heavy load on working memory which might be considered as a bottleneck for SA. The working memory is heavily overloaded especially in case of novice operators or an unexpected, novel or extreme situation where long-term memory is of no use.

Long-term memory contains schemata and mental models with aids the working memory with obtaining SA. Schemata represent coherent frameworks for capturing highly complex systems including their components, states, and functioning [6]. Several details of the situation are lost during the capture of a situation to a schema. Still, the schema can serve to capture a coherent picture of a given situation and may be efficiently recalled to aid working memory. Mental models represent a generalization mechanism for generation of general descriptions of the systems (e.g., explanations of functions, goals). An expert operator would have developed a numerous mental model that shift a situation representation to prototypical abstract codes. A mental model can then be understood as a schema for a particular situation. A situation schema can be matched to models in memory, that depict the prototypical situations of the system. These prototypical classifications may be linked to associated goals that dictate decision making and action performance [6]. For example, one can have a mental model of a car engine in general, but the situation schema is the state it is believed to be in now [13].

Elements

It is important to define the elements of SA in a given environment correctly. The process encompasses the identification of environment specific items that operator needs to perceive and understand to reach SA. The items differ across application areas and can not be applied interchangeably. Although both air fighter pilot and captain of an oil supertanker rely on SA, they would require a different set of elements for SA. It is possible to define general elements within a class of application areas or systems. For example, general elements can be defined for aviation, ship transportation, power plant operation, cyber security of concern management. The individual categories for elements may also state a set of requirements on elements (e.g., spatial and geographical awareness required pilots). The identification and definition of such elements allow a to develop a consist and robust description of SA for a given area.

Decision Making

Situation Awareness should be strictly separated from the process of decision making [6, 13]. SA provides an operator's internal model of the situation that serves as input for his/her decision process. Even with a correct SA of the actual state, an operator may come to a wrong decision. The decision process can be involved the operator's personal characteristics, e.g., level of risk aversion, or by momentary operator's conditions, e.g., psychological mood, stamina or health. The distinction of SA from decision process eliminates the aspects of the decision process itself. Thus, the focus remains on reaching SA and is not widen by additional assumptions and conditions.

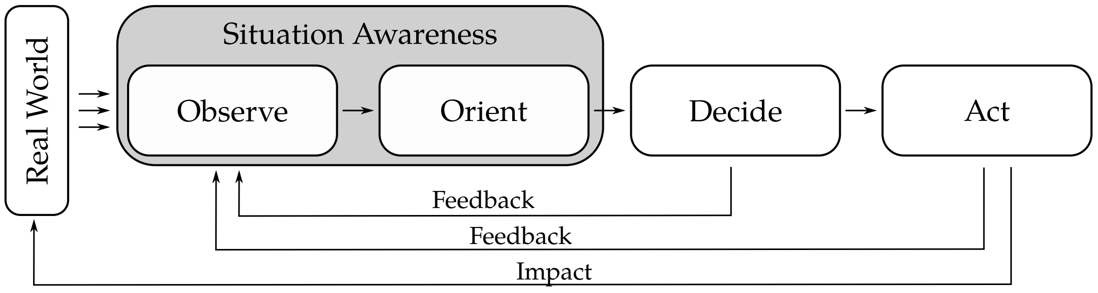


Figure 8: Situation Awareness in OODA loop.

The separation from the decision process also complies with the *Observe-Orient-Decide-Act loop* (OODA), a simple model for reaction to events. The model defined by Boyd [17] initially served as an approach to comprehension, shaping and adaptation to and unfolding evolving reality that is uncertain, ever-changing, and interchangeable. The model is depicted in Figure 8. Outside information and unfolding circumstances are obtained in Observe phase. Also, the feedback from previous acts and decisions are considered. The Orient phase includes the analysis and comprehension of the information provided by Observe phase. Boyd states that orientation is influenced by several aspects, such as cultural traditions, genetic heritage, and previous experiences. Based on the orientation in the situation, the decision process comes in turn. The Decision phase comprised the selection of a specific action forms a set of available measures. The size of the set of available measures and decision maker's characteristic are the most influencing artifacts during this phase. The last phase of the OODA loop, the Act phase, represents the actual locomotion and action according to the decision process. The results of the actions affect the external environment, and so the results of the action enter the Observe phase, and the loop is closed. The part of the loop is also the feedback from the Act stage back to the observation stage.

The SA model fits the first two phases of the OODA loop. Mainly, the three-level model can be intuitively fit into this approach. Perception level is analogous to the Observe phase. The Comprehension level naturally addresses the Orient phase of the OODA loop. Although providing a model of the future state of the environment, the Projection level can still be mapped to Orient level. The projection of the future enables an operator to orient in the situation, and the future projection of various situations can be taken into account during selection of appropriate action. Some may argue, that SA, understood as a product, cannot be part of the OODA loop that represents a continuous process. On the contrary, SA as a product of situation assessment represents a knowledge of the system in a given iteration of the OODA loop. In the next OODA loop, a situation is re-assessed, and a new SA is reached. In this manner, a static product approach to SA is integrated into dynamic OODA loop.

2.2 Cyber Situation Awareness

We think of cyber situation awareness (CSA) as a subset of the general Situation Awareness, particularly the subset concerning the *cyber environment*. Using Endsley's three-level definition, cyber situation awareness is defined as ...

> *... the perception of the elements in the* **cyber** *environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future.*

In this subsection, we provide specifics that are associated with the application of the general SA concept into the cyber domain. We present viewpoints of renowned researchers of the CSA research field, discuss the specifics of a cyber environment through comparison with the original SA application in military, define entities and goals of CSA and provide a formal definition of the CSA. Next, we provide a literature overview to present the latest trends and identify relevant issues for application in network security.

2.2.1 Viewpoints to Cyber Situation Awareness

We present viewpoints of the renowned researchers of CSA field to demonstrate different approaches to CSA and areas of interest of contemporary research. The full summary of viewpoints are published by Barford et al. [18] and by Kott, Wang, and Erbacher [14]. We provide a comprehensive overview of the viewpoints that highlights the area of interest in the CSA.

Cliff Wang observes that information security is art rather than to be practiced as a science. Wang suggests that a new CSA technology should move the security to a more scientific approach. It should allow analysts to obtain a more concrete and complete comprehension of a situation. The new technology must deal with sophisticated adversaries via multidisciplinary research incorporating advances in areas such as machine learning or uncertainty management.

Analogically, CSA should provide a capability to "connecting the dots" according to *Sushil Jajodia*. He identifies the main issue of current CSA as a lack of the big picture. Current tools provide only a small picture They give a few clues about how attackers might exploit combinations of vulnerabilities. Even for a skilled analyst, it is difficult to combine such information to get a big picture. An approach to gain a big picture could be a tool that can process, analyze, and correlate information from various sources. Machine learning methods that can integrate and fuse a broad array of contextual information need to be developed to acquire such a tool. The requirement for inter-contextual, adaptive machine learning techniques for CSA has also been identified by *Thomas G. Diettrich*.

Jason Li comments that CSA can be incredibly overwhelming for an analyst due to complexity, scalability and uncertainty issues. CSA should provide a transformation of low-level data to meaningful higher-level information relevant for the analysts. Li believes that such a transformation can be achieved via enhanced correlation, machine learning, and vulnerability analysis. The CSA should also reflect a need of the human operators to a better understanding of the expert operator mental processes and models. Similarly, *Alexaner Kott* states that a challenging task for CSA to identify not only mental models of the operator but also the mental models of an adversary. The challenge to understand an adversary is multiplied by the problematic identification of the adversary the confusing and non-standardisable context of regular user's activities.

As *George Tadda* states, there is a lack of trust of operators in an automated decision process, especially if they do not know, how the decision was reached. The decision process should be deterministic according to Tadda's view. *Sommesh Jha* and *Matt Fredrikson* suggest to establish and formalize the entities and principles essential to CSA. They put the stress of provenance of the conclusions and ability to verify CSA conclusions. The successful solution of the latter issue would also lead to a solution of the Tadda's concern.

2.2.2 Specifics of Cyber Situation Awareness

Although a general definition of SA can also be applied to the CSA, there are several specifics of the cyber environment that need to be considered. These specifics have a significant effect on SA and, to some extent, shape the approach to SA and the research efforts for CSA. One of the original application of the SA is in the area of conventional military conflict. Therefore, we demonstrate the specifics of the CSA with a comparison to the SA for military purposes. Although we highlight differences between cyber and military SA, these two types are not contradictory. On the contrary, CSA complements SA for a military operation as both virtual and conventional battlefields are encompassed in the current Information Age conflicts [14]. The specifics of CSA are presented in the following paragraphs.

Cyber Environment

There are almost limitless possibilities in the cyber environment. The cyber environment has no borders. This dynamic world is highly malleable and potentially scale-free. The dynamics and limitless of such an environment is a challenge for situation assessment compared to the physical world of conventional military conflicts, where the environment is immutable and govern by the law of physics. The conflicts are potentially scale-free [19], an attack can come out of the blue with no warning, and the costs of joining a conflict are low. Spatial properties of the cyber environment are global [20], which makes the determination of the SA boundaries problematic if we do not want to use specification "everything/everywhere". Therefore, the location of own network/system is usually used as a spatial boundary for CSA [21].

Perception

Information for military SA can be captured both via specifics hardware sensors and by direct, physical observation. The hardware sensors and signal processing techniques play an important but not an essential role; a direct observation can be used in cases hardware sensors are not available. In the CSA, the information is gained solely by sensors. It is not possible to observe the information directly. Each sensor is a complex system that can be misused or deceived to provide false information. The inability to confirm information from a sensor by direct observation limits chances to see through the deception. Similarly to information, anniversaries cannot be directly observed. They can be detected only by an analysis of information captured by sensors, which allows the anniversaries to stay hidden in a network.

Performance

The required resources for launching an attack in a cyber environment are relatively small. Compared to a state or organization required in a conventional conflict, the resources needed for starting a cyber conflict may scale down to an individual with a few prerequisites. Another factor to consider concerning the performance is the speed of the events. The speed of the events is orders of magnitude quicker than in case of physical conflicts. The resources needed for processing of such large volumes of information are significantly higher in the case of CSA.

Attacker Takes the Advantage

According to traditional military doctrine, a defender gains numerous advantages, e.g., defensive fortification, information asymmetry) [14]. A cyber-attacker overtakes all advantages in the case of cyber conflicts. The advantages of a cyber-attacker include, but are not limited to anonymity (it can hide across national sovereignty boundaries), global reach to probe weaknesses, social engineering to exploit human weaknesses, and a possibility to pick a time, place and tools for an attack.

2.2.3 Entities

Entities relevant to CSA are part of a *cyber environment* as stated in CSA definition. Cyber environment refers to *cyberspace* in the context of CSA. The term "cyberspace" became known when it was used by Gibson in a science-fiction novel Neuromancer [22]. In the novel, cyberspace represents a *"graphic representation of data abstracted from the banks of every computer in the human system."* A cyberspace was defined from several perspectives as described by Strate in [23]. Relevant for the scope of this thesis are definitions by Nguyen and Alexander in [24] and Gozzi in [25]. The authors of [24] describe the cyberspace as "the totality of events involving relationships between humans and computers, between humans though computer and between computer themselves". The authors of latter definition [25] define physical cyberspace as "material base of computers (monitors, disk drives, modems, wires, and so forth), and their users". The common aspect of both definitions is the fact, the cyberspace relates to both machines and humans, that fits the general notion of SA. Applying these definitions of cyberspace to cybersecurity, it enables us to identify relevant entities for CSA.

We identify following major types of entities in the context of CSA based on above described definitions - *physical*, *immaterial*, and *human* entities. For each entity, we identify their properties and roles. Individual entities interact with each other and together influence the creation of CSA. It is important to note, that an element that forms the CSA does not necessarily have access to all information on entities.

A group of **physical entities** comprises the physical base of the cyberspace. This group includes computers and their peripherals, network infrastructure formed by wires, switches, routers, and other networking devices. Each device can be described by its properties. The properties of the computers include number and frequency of computer processing units, random access memory size or volume of the disk space. The network wires and lines can be characterized by its type, throughput or packet loss. The properties can also cover a general property, e.g., the criticality of computer or line. Beside the characteristics, each element plays a specific role. A computer can be, e.g., a workstation or server. A router can play a role of either a central, vital element of infrastructure or be a low-importance device in the last mile of some insignificant local network.

Next, we introduce a group of **immaterial entities** to capture the virtual essence of the cyberspace. The immaterial entities serve as a "new edge" for communication between humans and computers or between computer themselves. The immaterial entities cover computer programs, services provided in a network, and so forth. Each of the entities can be assigned with a set of properties, e.g., service can be characterized by an availability, set of functionalities it offers, confidentiality, and so forth. The immaterial entities had originally a strong relationship with a physical entity (a service was hosted on a given physical server). With advances in cloud computing and virtualization, the relation with a physical entity loosens. The services, programs, and even various network functions are operated in a virtual environment (or cloud) and are not linked to a physical machine.

Last group, **human entities**, represents people interacting with the computers. This group of entities can be described by various properties. Among the properties relevant for CSA belong to experience, attention, determination, perceptual skills, short/long term memory capabilities, and analytic skill [6]. A human entity is also characterized by a role. A role assigned to a human partially determines the goals of the human and consequently the situation assessment process. Each role would require slightly different information, comprehension and projection to reach CSA. Roles that we consider in this thesis relevant for CSA divide into two main group - attacker and defender role. Attacker role represents an adversary that intends to do deliberate harm to the object of interest. A defender protects an entrusted asset against the attacker. The defender roles can be, e.g., security analysts (analyze and assess existing vulnerabilities in IT infrastructure), security architects and engineers (design and utilize technologies to enhance security), or administrators (manage organization-wide security systems) [26].

Although all mentioned roles constitute the cyber environment, only human entities usually reach CSA. All three entities form cyberspace, and therefore they are part of the CSA, i.e., are part of the knowledge of the situation. They form situation and influence its state. However, the actual knowledge of a situation (CSA) is associated with human entities only, in the context of this thesis. The human entities can both be part of a cyber environment and reach CSA of the environment at the same time.

2.2.4 Literature Review

An overview of relevant literature is provided to present current state-of-the-art in the field of CSA. First, we demonstrate the increasing importance and demand for CSA. Next, we present an overview of results of queries for keywords related to cyber situation awareness over relevant scientific databases[2] to capture a full range of literature. Further, we identify and present leading

2. The overview queried following databases: IEEE Explore, Scopus, and Web of Science

researchers from the surveyed papers and describe their work. Last, but not least, we mention other works relevant to CSA.

The CSA has been included in the cybersecurity strategies of many nations, which indicates the need and importance of CSA even at national level. The Australian Cyber Security Operations Centre (CSOC) "provides the Australian Government with all-source cyber situation awareness and an enhanced ability to facilitate operational responses to cybersecurity events of national importance" [27]. The United States of America defines their role in cyberspace's future defense as "steady progress towards shared situation awareness of network vulnerabilities and risks among public and private sector networks" [28]. The German national cybersecurity strategy [29] establishes National Cyber Response Center to "directly inform the crisis management staff headed by the responsible State Secretary at the Federal Ministry of the Interior if the cybersecurity situation reaches the level of an imminent or already occurred crisis". The United Kingdom aims to "enhance cyber threat awareness, detection, and reaction functions, through the development of a Cyber Security Operations Centre (CSOC) that uses state-of-the-art defensive cyber capabilities to protect the cyberspace and deal with threats [30]. The Canadian Cyber Incident Response Centre serves "to be the focal point for monitoring and providing advice on mitigating cyber threats, and directing the national response to any cybersecurity incident" as described in Canada's Cyber Security Strategy [31]. The analysis and monitoring the treats and risks on a continuous basis in national critical information infrastructures is one of the leading goals also in National Cyber Security Strategy of the Czech Republic for the period from 2015 to 2020 [32]. The cybersecurity strategies usually target the critical infrastructure of a nation, government systems in the majority of the cases. The common goal is to provide both financial as well as jurisdictional support for cyber defense. An important role in the nation's cyber defense play cybersecurity teams established at the national level. The teams serve as a central element of the cyber defense. The research in the area of threat detection or network monitoring is accented in many strategies [30, 32]. Although mentioned strategies do not use the term Cyber Situational Awareness explicitly, the goals and actions described in the strategies follow CSA principles and aim to reach CSA.

The need and relevance of research on CSA are demonstrated by the increasing numbers of research published on this topic. We surveyed four major scientific databases for keywords cyber, situation(al), awareness to show the coverage of the topic among research community. The time range of the search was set from since January 2003 up to July of 2018. The numbers of publication on CSA published thorough years are presented in Figure 9. There is an identifiable hype of interest in the CSA starting in the year 2012 and culminating in 2015. The relevant papers are mostly indexed by Web of Science core collection followed by IEEE and Scopus collections. The Springer database contains significantly fewer records except the year 2017 when a collection of papers on CSA was published in the title Theory and Models for Cyber Situation Awareness [33]. According to survey by Franke and Brynielsson [34] conducted in 2014, the majority of publications covers the design of CSA and attack detection & analysis while application area, thread description, and workflows are covered rather sporadically.

Major publications on CSA are these two following collections: *Cyber Situational Awareness: Issues and Research* edited by Jajodia et al. (2010) [35] and *Cyber Defense and Situational Awareness* by Kott, Wang, and Erbacher (2014) [14]. These collections provide a comprehensive introduction to the topic, explanation of basic concepts and discuss prevailing open issues of the topic. A summary of recent research advances in CSA are summarized in collection *Theory and Models for Cyber Situation Awareness* by Liu, Jajodia, and Wang (2017) [33]. An exhausting overview of the literature is provided by Franke and Brynielsson [34]. The authors provide a literature overview for specifics areas of CSA such as industrial control systems, emergency management, and military. They also list design papers on tools and visualizations for CSA. We were not able to find a similar survey covering CSA area published since then. A very brief survey covering

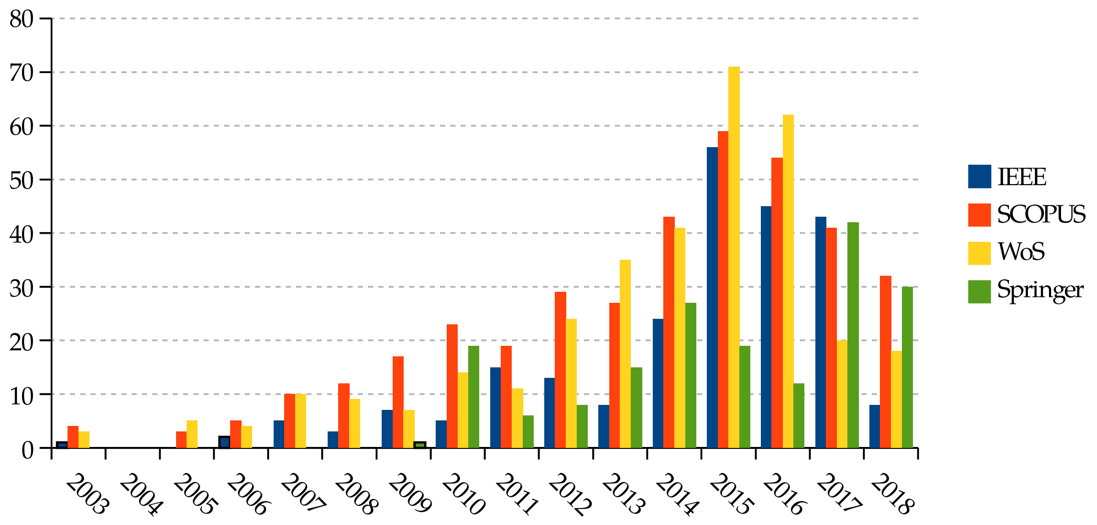


Figure 9: Number of published papers on cyber situation awareness.

the anomaly detection area of CSA was published in 2015 by Friedberg, Skopik, and Fiedler [36]. The latest review of current CSA models and definitions was published by Cooke et al. in [37].

Apart from fundamental publications, our literature survey also focuses on the leading research groups in the CSA field. A strong research group has been established at Center for Secure Information Systems at George Mason University[3]. The director of this group, prof. Sushil Jajodia, Ph.D. gathered a strong team of scientists and started to research the CSA in the project Computer-aided Human Centric Cyber Situation Awareness. The team's members worth mentioning are Massimiliano Albanese, Ph.D. who is interested in modeling network attack and network hardening, and Steven Noel, Ph.D. who's main research area is attack modeling, graph analytics, and visualizations for information security. The group significantly contributes to the formation of CSA [18, 26], to models underlying CSA [38–40], and to CSA measurement [41]. Further, the group investigated concept of attack graphs and their application in CSA [42–44], network hardening [45] along with relevant strategies [46], and cyber deception [47–49]. The group has also developed a framework for cyber situation awareness that integrate an array of techniques and automated tools [50, 51]. The framework is able among others to represent dependencies in a network or model attack scenarios using attack graphs.

Another research group is lead by Alexander Kott at US Army Research Laboratory (ARL) [4]. They also contributed to the fundamentals of CSA by the formalization of cybersecurity problems [52, 53] and editing a collection that provide an introduction to the basics principles of CSA [14]. The ARL is interested in industrial control system protection; risk modeling and intrusion detection are investigated in [54], metrics of SCADA security in [55], and simulation of the cyber defense in [56]. Other topics under the research focus of the ARL related to CSA are resilience of the cyber systems and networks [57, 58], modeling of cyber intrusions [56, 59, 60], and attack prediction [61].

Next research group investigating CSA has formed in Sweden at Swedish Defense Research Agency[5] and RISE SICS[6]. The group is represented mainly by ass. Prof. Joel Brynielsson, Ulrik Franke, Ph.D., and their students. A significant contribution to CSA is a review on CSA literature [34] that provides a systematic overview of research areas in CSA and number of published papers on these areas. Further, the group researches CSA testing [21], modeling an attacker via

3. `http://csis.gmu.edu/`
4. `https://www.arl.army.mil/`
5. `https://www.foi.se/fusion/`
6. `https://www.sics.se/`

attack persona concept [62], and differences in understanding of CSA between normal and IT skilled employers [63].

Prof. Shanchieh Jay Yang, Ph.D. formed a group interested in some aspects of CSA at Department of Computer Engineering at Rochester Institute of Technology[7]. The group has contributed to the major publications on CSA with situation assessment research [64–66]. This group specializes in the area of attack modeling and prediction. They investigate different approaches to attack prediction [67] including Bayesian networks [68], time series forecasting [69], and likelihood estimation using rare-event simulation [70]. They are especially interested in multi-stage cyber attacks. Their research in this area includes characterization of multistage cyber attacks [71] and creation of simulation system for multistage attack emulation that fuses concepts from computer networks, system vulnerabilities, attack behaviors, and scenarios [72].

Last, but not least research group is formed around dr. Florian Skopik at Center for Digital Safety and Security of AIT Austrian Institute of Technology[8]. The researchers at AIT investigates the decision support models in CSA used in cyber operation/security centers [73, 74], CSA in smart grids [75], and network anomaly detection [36]. The design of a system for national situation awareness [76] is also investigated by this group.

2.2.5 Challenges

This section introduces challenges for cyber situation awareness extracted from the surveyed literature sources mentioned in the previous section. The challenges of CSA were mainly discussed by Kott, Wang, and Erbacher in [14], further by Holsopple, Sudit, and Yang in [65], and last, but not least by Dressler et al. in [77]. We cluster the discussed challenges into three separate categories according to the target area. The first category represents the challenges resulting from the specific properties of cyberspace. The second category focuses on the challenges associated with the data processed to obtain CSA. The last category discusses the practical application of CSA principles and related challenges to toolset and automation of the CSA. Following paragraphs present each of the categories in detail.

Cyberspace-related

Adaptation of Situation Awareness to cyber domain is not straightforward due to specifics of the cyberspace (see Sec. 2.2.2). The challenges for CSA resulting from the specifics of cyberspace are following: *complexity*, *speed of events*, *dynamics*, and *rapid evolution*.

- **Complexity** – The cyberspace, especially the computer networks, are complex, large environments. Nodes in a computer network are interconnected, depend on each other, are organized into various structures, and topologies. Systems running in these networks are complex, nested, and often hard to understand. These facts make the developing and maintaining a mental model of a complex network with information of all components, their properties, and inter-connections a challenging task. People's ability to create a mental model is often rapidly exceeded, which influences the SA comprehension phase and consequently also projection phase [14].

- **Dynamics** – The cyberspace is not static; it changes all the time. New nodes are added or removed, systems are updated, new technologies are introduced, and entities are entering or leaving with mobile technologies. Analogous to complexity challenge, it is hard for an operator to capture the dynamics and keep track of all changes in the cyberspace.

7. `http://www.rit.edu/kgcoe/computerengineering/`
8. `https://www.ait.ac.at/en/about-the-ait/center/center-for-digital-safety-security/`

- **Speed of events** – Events can occur at near-light speed in cyberspace. Cyber attacks occur frequently and happen in a fraction of a second. Since a human is not able to handle events at such speed, finely tuned automated tools are required to assist the human (e.g., network visibility tools, attack detection tools, and so forth). These automated tools must be able to process and analyze all incoming events swiftly and provide a human with necessary summarized information in time with high precision. Moreover, if an attack is new or at least unknown to an operator, the operator does not have any tools. He/She needs to analyze attack manually which further delays process of understanding an attacker's intentions needed for network defense.

- **Rapid evolution** – The cyberspace is characteristic by its rapid technology change. New hardware and software are introduced almost on daily basis [14]. Each new technology has its specifics, new properties, and functions. An operator is forced to keep up with the latest trends to maintain an up-to-date model of the cyberspace, its vulnerabilities, characteristics, and behavior. However, keeping up with the latest trend in a whole cyber domain is beyond human abilities. According to Symantec 2017 Internet Security Threat Report, 375 M of new malware variants was discovered in 2017 [78]. Such a large number of malware means that understanding the malware and its effects is impossible for a person. Automated tools and techniques need to be employed.

Some of the cyberspace-related challenges of CSA are being already responded to. Computer security vendors are introducing new tools for network visibility, attack detection, and cyber defense to ease complex network comprehension and to reduce a workload on a human operator. Nevertheless, other challenges still prevail. The *Speed of events* challenge is still topical as current tools may introduce a delay into the analysis process. The *Complexity* and *Dynamics* challenges are not still completely solved too. For example, IP flow based monitoring, used widely to network monitoring, does not provide a detailed host view straightforward, visualization techniques of network topology in IPv4 and IPv6 address space are still under research.

Data-related

Data processed in CSA has all characteristics of *big data*. Big data is defined by four key characteristics - *volume*, *velocity*, *variety* and *value* [79, 80]. To demonstrate the *big data* characteristics of data processed in CSA, we highlight selected conclusions of Cisco report on IP traffic forecast [81]. According to the report, the volume of global IP traffic was 96 EB per month in 2016 and is expected to rise to 278 EB per month by 2021. The velocity of the network traffic is forecast to nearly double by 2021. The variety of the network traffic will increase too as new applications and protocols are continuously developed (e.g., the number of applications in Apple App Store increased from 585 thousands in 2012 to 2.2 millions in 2017 [82]). The value of the information in network traffic can be demonstrated by the market value of the network security segment that is estimated to reach 11.669 millions of USD at the end of 2018 [83].

However, each of the demonstrated *big data* characteristics of CSA data opens new challenges for CSA. We present challenges of CSA linked to big data key characteristics below:

- **Volume** – The volume of the data captured and produced by sensors in cyberspace provides an operator with a massive amount of available information. However, information and data are usually in a raw form that brings a little understanding to an operator. An operator is overloaded with data with no meaning to him/she. This challenge can be appropriately summarized as *"Data overload - meaning underload"* as mentioned in [14].

- **Velocity** – The velocity of incoming data combined with a demand for real-time analyses and results delivery puts challenging requirements on data processing. Tools for data processing and analysis need to provide sufficient throughput to process all data at high-speeds. Moreover, the real-time data processing accents time ordering of data as events occur at millisecond rates in cyberspace. Correct time ordering of the events is required for data analyses where causality is of interest (e.g., root-cause analyses, advanced persistent threat detection).

- **Variety** – The variety of data opens challenges mainly in the data processing area. The current trend is to process all data from different sources and of a different type in a centralized manner - e.g., in a data cloud. With central processing, determining the proper metrics and alert thresholds is essential [77]. Moreover, data refinement and normalization is necessary for data synthesis [77]. Data is collected in different formats based on source tools. For central processing, data is required to be transformed into a common format while keeping the information carried in original data. A central data processing introduces further issues, such as data duplication, unreliable data sources, errors in data.
- **Value** – The value of the data is determined by the *value of information*[9] carried in data for an operator of CSA. The information carried in data is, however, devaluated by the presence of a high noise to signal ratio. Anomalous events are common in a cyber world; systems might not work properly, users behave unexpectedly, protocols are not used according to standard, and so forth. Such disruptions to a "normal" behavior complicate relevant information retrieval and introduce a noise into data, which reduces its value.

Toolset-related

A toolset used by an operator significantly influences his/her level of CSA. An operator observes cyberspace using different monitoring tools, comprehends data via tools for data analysis, and presents results utilizing various visualization tools. The properties, available functions, and performance of available toolset determines an understanding of the current situation and its assessment. Available tools are under rapid development [83]. Nevertheless, the toolset is still facing several challenges, that can be clustered into the following three areas: *performance*, *heterogeneity*, and *visualization*. Performance challenge relates to processing big data and discuss mainly scalability and throughput issues. Heterogeneity challenge addresses the disperse tool landscape and resulting issues from the tool heterogeneity. Visualization challenge represents the issues linked to data presentation. Individual challenge areas are discussed in detail below.

- **Performance** – The performance challenge is closely related to data challenges described above, namely the big data characteristics of data processed in CSA. The volume and speed of the processed data impose a demanding requirement for data processing and computational power of current CSA tools. Namely, scalability and throughput of the tools are currently trending challenges for CSA toolset. The scalability challenge is being met by the development of cloud-based distributed solutions. The throughput is increased by an extensive parallelization of computations and data processing tasks. A novel approach to scalability and throughput challenges that is gaining attention is distributed data stream processing. Other challenges of CSA related to performance are the reduction of analysis time and response time. The operator needs new information as soon as possible to be able to react in time. The approaches to the analysis of CSA data are subjects to delay, partially by their design and by the big data processed. The data processing workflows and analysis methods need to be improved and optimized to reduce delays caused by data analysis.

9. Value of information description including economics factors are defined in [84]

- **Heterogeneity** – A CSA operator needs various information from different data sources (network traffic, logs, ADS and IDS systems, antivirus tools, and so forth). A special tool is used to process data from each different source, and an operator is forced to switch between the tools and utilize a number of different analysis workflows to retrieve needed information. These facts create a highly manually intensive process that hampers current cyber operations. A tool integration and unified approach to CSA data analytics are demanded.

- **Visualization** – A visualization of data plays an important role in situation comprehension. The visualization approaches for cybersecurity are discussed in literature [85–87]. Nevertheless, several issues are still left open for research, such as big data visualization, SDN networks, and human-centered evaluation [86]. An open issue related directly to CSA is a visualization of large-scale complex and dynamically changing networks [14]. The visualization of the networks should be able to scale across levels of detail and across time so that an operator has an instant approach to both overview and detailed information.

2.3 Summary

This chapter introduces background information for cyber situation awareness necessary for understanding the context of our research. We provide an overview of the cyber situation awareness domain, introduce relevant terminology, survey relevant literature, and present the major findings in this area.

First, we introduce a reader to a general concept of Situation Awareness and present key approaches to Situation Awareness. General aspects of Situation Awareness are discussed as well. Next, we narrow our focus to the cyber situation awareness, one of two central research area of this thesis[10]. We demonstrate an adaptation of a general concept of Situation Awareness into cyber domain including the specifics of the adaptation and entities relevant to cyber situation awareness. Further, we provide the literature review for cyber situation awareness research area. We describe significant publications in the area and identify significant research groups. For each research group, we present their research interests, relevant publications and major findings on their research topic. Last, we provide a coherent overview of the current challenges for cyber situation awareness research. The presented challenges relate to the research topic of this thesis, and we aim to respond to these challenges to some extent.

The main contributions of this chapter are:

- coherent description of main approaches to Situation Awareness,
- overview of cyber situation awareness including its specifics,
- literature review of cyber situation awareness,
- description and categorization of the open issues of cyber situation awareness.

10. The other research area *Network traffic monitoring* is described in Chapter 3

3 IP Flow Network Traffic Monitoring

A computer network is a center of cyberspace where all communication takes place. Network visibility is an essential requirement for understanding processes in computer networks. The understanding of a network enables us to manage the network effectively, defense it from adversaries, and improve the quality of network service, and so forth. IP flow monitoring and analysis is one of the major approaches that provide network visibility, even in high-speed, large-scale networks.

This chapter aims to provide a detailed introduction to the concept of IP flow network traffic monitoring and to outline the application of the IP flow network traffic monitoring to cyber situation awareness domain. We present the origins and evolution of the IP flow concept to show the original purpose of the IP flows and to highlight current limitations of the concept resulting from the original purpose. Since term "flow" is massively overused in the area of network security, we provide a coherent definition of the IP flow to avoid any misinterpretations. Further, we describe both IP flow monitoring and analysis workflows. We focus on the explanation of IP flow record creation, processing, and analysis. The structure of the description is inspired by an excellent survey by Hofstede et al. [88], which we recommend for further reading. The information from the survey is updated with current up-to-date findings, especially when describing the current trends and open issues of IP flow monitoring.

We also study the workflow from the temporal point of view, which is, to the best of our knowledge, the first attempt that focuses on the time aspects of the IP flows workflows. This chapter concludes with the application of the described IP flow monitoring and analysis workflows to cyber situation awareness domain using Endsley's Three-level model described in the previous chapter. We revise the specifics and challenges of cyber situation awareness in the light of IP flow network traffic monitoring and identify prevailing challenges.

This chapter is structured as follows:

- *Section 3.1 places the network traffic monitoring in the broader context.*
- *Section 3.2 presents the history, concept and definitions of IP flows along with contemporary trends.*
- *Section 3.3 describes the IP flow monitoring workflow including prevailing open issues.*
- *Section 3.4 describes the IP flow analysis workflow including current open issues.*
- *Section 3.5 presents the IP flow monitoring in the context of cyber situation awareness.*

3.1 Network Traffic Monitoring

Network traffic monitoring has become one of the constituent parts of network security. Network security is a sub-field of *information security* and applies general approaches of information security to computer networks. The goal of the network security is to protect *confidentiality*, *integrity*, and *availability* of a network. The network monitoring process contributes to the network security and consist of collecting and recording data from the relevant network elements in operation [89]. The research fields related to IP flow network traffic monitoring, as understood in this thesis, are depicted in Figure 10. IP flow monitoring is one of the methods for network traffic monitoring. Network traffic monitoring along with monitoring of network infrastructure represent network monitoring research field. The network monitoring area is part of a network security research field. As stated before, network security is a sub-field of general information security. In the next paragraphs, we will present main approaches to network monitoring, describe methods for traffic access and detail main approaches to traffic network monitoring to provide a broader context for the other main scope of this thesis - network IP flow monitoring.

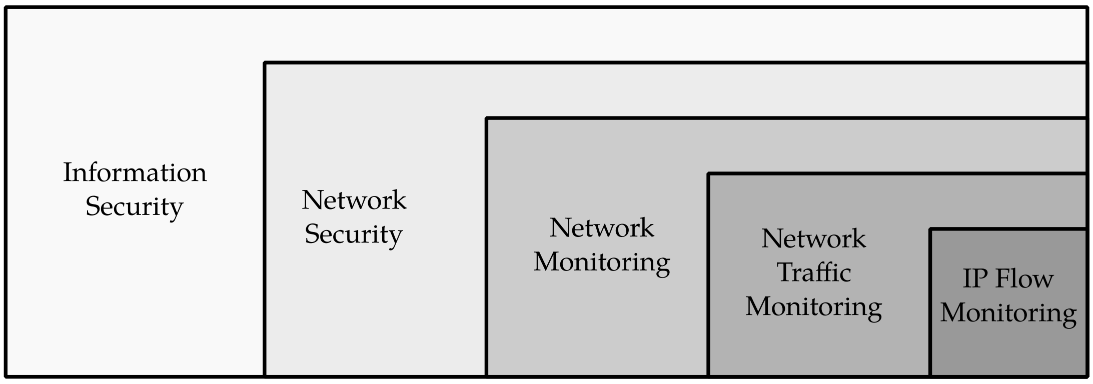


Figure 10: Network traffic monitoring research field.

Two main mechanisms apply for network monitoring: active and passive [88, 89]. The active network monitoring actively injects testing traffic into a network. Based on network response, it infers network status. Active approaches are used for network topology discovery, quality of service measurement, availability checking, or determination of connection properties. Active network monitoring is implemented in tools like *Ping*, *Traceroute*, *Nmap*[1] or *ZMap*[2]. Passive approaches to network monitoring observe existing network traffic as it passes via an observation point in a network. The network traffic is observed using traffic monitoring access methods described later in this section. The passive approach to network observation is mainly used for network traffic monitoring. Nevertheless, the network traffic monitoring is often generally understood in the broader context than observing packets in a network; it also includes network traffic collection, storage, and analysis for various purposes (e.g., threat/anomaly detection, outage detection, or network accounting).

Traffic monitoring access methods are used to observe network traffic for network traffic monitoring purposes. The access methods provide a transparent way to observe packets in a network without any negative effect on network functionality and network traffic delivery. There are two generally accepted types of traffic monitoring access methods: *in-line* and *mirroring* methods [88]. We distinguish a third type of access method - the *build-in* method that emerges due to improved capabilities of network switching and routing devices. All three types of access monitoring methods are described below.

1. `https://nmap.org/`
2. `https://zmap.io/`

- **In-line** traffic access methods observe network traffic using a capture device placed to a monitored line between two network elements. The capture devices are additional hardware installed to a line, such as bridging hosts or network traffic access points[3] (TAP) [90]. Network TAP is a device designed to provide a permanent access port for passive monitoring of all traffic without data stream inference [90]. The TAP duplicates network traffic using splitting or regeneration technology. The duplicated network traffic is sent to the connected monitoring device for further processing. In a copper network, TAP duplicates a signal sent over a line, which requires a constant connection to power supply. A TAP for a fiber line does not necessarily need the power supply as a light signal is split by a prism installed in the TAP. The split ratio representing the amount of light that is redirected from network to monitor ports is ranging from 30:70 to 50:50 in the majority of installations.
- **Mirroring** traffic access methods leverage a feature present in most of the packet forwarding devices – packets mirroring from one or more ports to another port. This feature is referred to as *port mirroring* or *Switched Port ANalyzer (SPAN) session*. SPAN port and mirrored ports are defined in a packet forwarding device's configuration. A network traffic monitoring device is then connected to the SPAN port. The port mirroring modifies monitored traffic, unlike TAPs where monitored traffic remains intact. Port mirroring introduces additional timing difference artifacts, changes traffic stream and reorders packets [91]. Despite mentioned issues, mirroring traffic access method remains a widely used and popular method for network traffic monitoring due to its wide availability and low cost.
- **Built-in** traffic access methods represent a network monitoring software implemented directly to network switching devices. With the recent rise of computational resources in network switching devices and the introduction of new networking concepts such as software-defined networks (SDN), devices with implemented network monitoring software emerges, e.g., *Open vSwitch*[4] supports NetFlow export. It should be noted, that this traffic access method is not suitable for high-speed networks. The network monitoring software is usually only a complementary feature to the primary purpose of the devices - network switching. In the case of high traffic loads, a device allocates resources to fulfill its main function and network monitoring resources are reduced. Such behavior is undesirable in case of distributed denial of service (DDoS) attack, for example.

The main approaches to network traffic monitoring can be categorized by level of abstraction of information collected from network traffic to following three categories: *volumetric statistics*, *deep packet inspection (DPI)*, and *IP flow monitoring*. The level of description is determined by a volume of information from the packet that is processed during a monitoring process. The level of abstractions also affects other characteristics of the approaches to network traffic monitoring, mainly their throughput and resource requirements. The comparison of the approaches to network traffic monitoring is depicted in Figure 11.

- **Volumetric statistics** approaches do not require to analyze packet at all. Instead, these statistics are usually collected directly from network devices using simple network management protocol (SNMP) [92] or remote network monitoring RMON system [93]. The collected information provides only volumetric information on the network traffic including the number of transmitted or received packets, bytes, number of errors. The volumetric statistics provide insufficient information on the nature and content of network traffic and usually serves mainly for basic network accounting.

3. Also referred to as *Test Access Point*.
4. http://www.openvswitch.org/

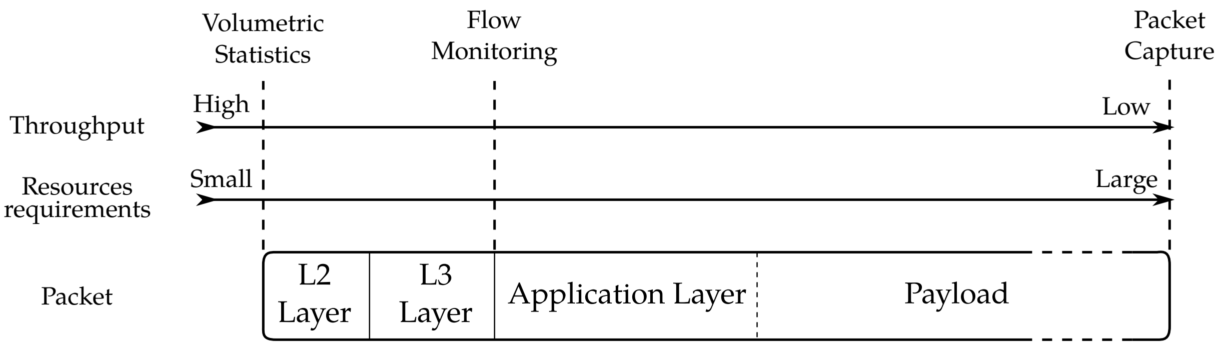


Figure 11: Comparison of approaches to network traffic monitoring.

- **Deep packet inspection** analyses the entire packet, i.e., both header and data part of the packet. The advantage of this approach is that the volume of information extracted from network traffic is the highest from the mentioned approaches. However, such high visibility is outweighed by a lower throughput of the DPI approach and it's high resource requirements. Hence, DPI is suitable rather for analysis of targeted network traffic than overall monitoring of the whole network. Moreover, the volume of information available via DPI is, and will continuously be reduced by an increasing portion of encrypted network traffic [94].

- **IP flow monitoring** is a balanced trade-off between a volume of information available and scale of analysis. In general, IP flow monitoring analyses only information from packet headers, which allows for high throughput and relatively low resource requirements on the one hand. On the other hand, the volume of information is reduced to basic metadata about a connection in a network. The IP flow monitoring process for large-scale Internet measurement has originally been described in [95]. The IP flows and associated monitoring process are discussed in detail in the following sections.

3.2 IP Flows

The purpose of this section is to introduce the concept of IP flows. The following paragraphs briefly outline the history of IP flows that helps to understand the origins of IP flows and associated consequences. Next, we present and explain the definition of an IP flow. Since the term *flow* is overused in the network monitoring research field, we aim to provide a precise definition that will be used consistently throughout this thesis.

3.2.1 History and Evolution

There are two main driving forces present throughout IP flow history: the *Internet Engineering Task Force* (IETF), an open community of network designers, operators, vendors and researchers concerned with the evolution of the Internet, and *Cisco Systems, Inc.* (Cisco), one of the largest network companies in the world. The origins of flow monitoring concept are linked to IETF. IETF, however, did not manage to assert a wider industry adoption of the concept. It required a Cisco's native implementation and market position to make a flow monitoring a widely accepted and used concept. The overview of the history of flow export is displayed in Figure 12. In the following paragraphs, we describe evolution related to IETF and Cisco separately.

The first published mention of a concept resembling IP flow dates back to 1991. In November 1991, *Internet Accounting (IA) Working Group (WG)* of IETF issued request for comments no. 1272 (RFC 1272) describing background information for the "Internet Accounting Architecture" [96].

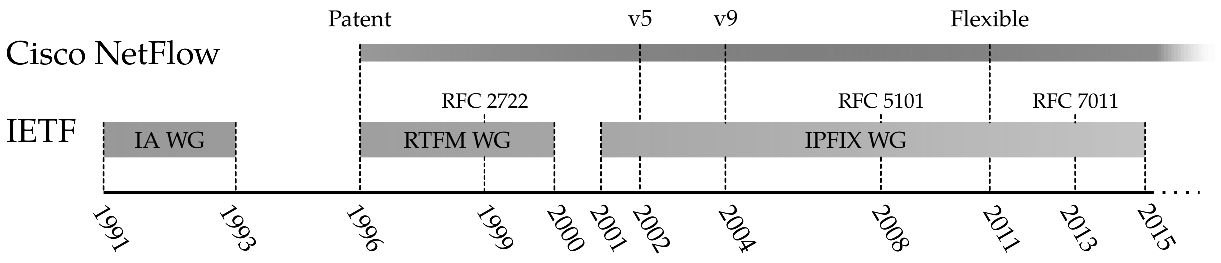


Figure 12: The history of IP flow export.

RFC 1272 purposed a model for internet accounting with a motivation to understand a subscriber's behavior, to monitor compliance with stated policy, and to allocate costs rationally. The model for Internet accounting resembles contemporary models for network traffic monitoring. It comprises of meters used for examining a stream of packets and collectors used for data storage. The model further describes the meter's types and structure and discusses issues linked to the collection. A flow description is included in a meter's definition: *"... The meter records aggregate counts of packets belonging to FLOWs between communicating entities... ...The assignment of packets to flows may be done by executing a series of rules.."* [96]. The IA working group has concluded its charter in 1993. The workgroup did not achieve any wider adoption of the proposed concept, mainly due to lack of vendor's interest and a common attitude that the Internet should be free, i.e., no monitoring should take place [88].

IETF reincarnated the flow related efforts in 1996, based on impulse from a published paper by Claffy, Braun, and Polyzos [97] in 1995. *Real-time Traffic Flow Measurement* (RTFM) WG was established with main objectives to consider security&privacy issues in traffic flow measurement, to produce an improved traffic flow model with a simpler flow specification and a strong focus on data reduction, and to develop the RTFM architecture as a standards document for IETF. The RTFM WG revised the work by IA WG and proposed a Traffic Flow Measurement Architecture in RFC 2722 [98]. The proposed architecture applied to any protocol or multiple network stacks, and it enabled user-defined traffic flow measurement requirements. The data reduction effort was accomplished by placing a flow producer as near as possible to network point. A flow was defined in the context of RFC 2722 as *"... an artificial logical equivalent to a call or connection. A flow is a portion of traffic, delimited by a start and stop time. Attribute values associated with a flow are aggregate quantities reflecting events which take place between the start and stop times..."* [98]. The RTFM WG concluded its charter in 2000, still without any standard for flow export issued, though.

Next effort of IETF to standardize a flow export protocol took place a year later in 2001. *IP Flow Information Export Work Group* (IPFIX WG) was started with a goal to define a standard for flow information export. After four years of analysis, the IPFIX WG published an informational RFC 3197 [99] in 2004 where requirements for IP flow information export were stated. The RFC 3197 provides a base of IP flow definition, which is with small modifications used nowadays[5]. Besides the definition, RFC 3197 united terminology used in IP flow network traffic monitoring and identified applications suitable IP flow information export – usage-based accounting, traffic profiling, traffic engineering, Quality of Service (QoS) monitoring, and attack/intrusion detection. Next, IPFIX WG evaluated candidate protocols for information export (RFC 3955) stating that Cisco's NetFlow v9 is a best starting point for a new protocol specification [100]. The requirements and evaluation were forged into a proposed standard RFC 5101 [101]. The standard provided the specification of the IP flow information export protocol (IPFIX)[6] for the exchange of IP traffic flow information. The specification defined IPFIX message format including header

5. The definition itself is omitted here, as the IP flow definition will be discussed in detail in the following subsection.
6. Please note that abbreviation *IPFIX* refers to the protocol for IP flow information export while *IPFIX WG* refers to IP Flow Information Export Working Group.

and record format. It introduced a concept of template record formats, which provided flexibility in records transferred and allowed for a user-specified format. The proposed internet standard RFC 5101 was turned into internet standard in RFC 7011 [102]. Apart from protocol specification, the IPFIX WG worked on other aspects of information export such as information models [103], flow selection techniques [104], flow aggregations [105], or textual representation of abstract data types [106]. The IPFIX WG concluded in 2015. Since then, the IPFIX is further developed by the community.

In parallel to IETF, the IP flow information export technology was developed by a world-leading network company Cisco Systems, Inc. The Cisco's approach to information export origins in switching approach implemented in Cisco's routers and switches. The flow-based switching maintains all information on connections in a flow cache. Forwarding decisions are made in a control plane only for the first packet, and all subsequent packets are switched in the data plane. The discovery of the value of information stored in a flow cache led to the development of flow export technology. A protocol for flow export, called NetFlow, was patented by Cisco in 1996 [88]. Since Cisco made the protocol and corresponding data format freely available and the technology was naturally present in Cisco's devices, the technology offered a high potential to be widely used. In 2002, NetFlow v5 met the potential and became the first widely adopted protocol for network flow export, even though no official documentation on this protocol version has ever been published. The NetFlow v5 has a fixed format and therefore can carry only limited information (e.g., is limited to IPv4). In 2004, NetFlow v5 was replaced by a more flexible format in NetFlow v9. NetFlow v9 introduced support for flexible data formats, IPv6, virtual local area networks (VLANs), and multiprotocol label switching (MPLS) [88]. NetFlow v9 was described in informational RFC 3954 [107] and served as a basis for IPFIX specification as mentioned before. In 2011, Cisco introduced a NetFlow-lite technology built upon the Flexible NetFlow. Flexible NetFlow is an evolution of NetFlow based on NetFlow v9 and IPFIX that introduces a new concept of immediate cache enabling immediate export of network flows. Flexible NetFlow also provides an ability to export additional information from a packet, such as a packet section. NetFlow-lite leverages the advantages of Flexible NetFlow and, thanks to packet sampling, provides visibility similar to NetFlow at a lower price point without the use of expensive customer application-specific integrated circuits (ASIC) needed otherwise. The low price makes and its flexibility makes NetFlow-Lite a technology suitable for providing visibility in data centers [108].

The benefits and key customer applications are listed by Cisco as follows [109]: network monitoring (on a network-wide basis), application monitoring and profiling, customers monitoring and profiling, network planning, security analysis, and accounting. A flow for NetFlow v5 is identified as *... a unidirectional stream of packets between a given source and destination – both defined by a network-layer IP address and transport-layer source and destination port numbers.* The NetFlow-Lite provides a more general flow definition: *... a unidirectional stream of packets that arrives on a source interface and has the same values for the keys. A key is an identified value for a field within the packet. You create a flow using a flow record to define the unique keys for your flow.* [110].

Apart from Cisco, other vendors introduce alternative technologies for flow information export. Juniper Networks [7] provides *Jflow* [111]. *sFlow* is an industry standard for export of truncated packets creating records similar to IP flows described in RFC 3176 [112]. The current version sFlow v5 is supported by vendors such as Hewlett-Packard, Huawei, IBM, and others. Another relevant technology linked with the upswing of software-defined network (SDN) is *OpenFlow* [113]. In SDN, a control and data plane are separated. OpenFlow serves mainly as a flow-based configuration technology for the control data plane that enables sophisticated and remote traffic management. Apart from the configuration, OpenFlow can export flow level infor-

7. `https://www.juniper.net`

mation from the control plane (e.g., byte and packets counters). This property has been utilized for network's volumetric statistics measurement in [114, 115].

The history review of IP flows has shown, that the original purpose of IP flow was mainly network accounting and profiling. The accounting function of IP flow has determined the design of IP flow monitoring infrastructure. IP flows and associated IP flow monitoring infrastructures are designed to holistic monitoring of network with the focus on regular network reporting. The original design and purpose of IP flow, however, is being continuously transformed from the original to a modern one due to technological advances in networking area (faster networks, SDN), the shift in protocol usage (rise of encrypted traffic), and new possibilities and paradigms for data processing in general. The IP flow definitions, monitoring infrastructure, and related toolset, however, remain with the original purpose, which introduces discrepancies in IP flow applications and other issues in flow processing and analyses.

3.2.2 IP Flow Definition

Throughout time, the definition of IP flow has evolved from a notion resembling a communication in network traffic to a flow definition specified in an internet standard. The historical version of IP flow definition is described in the previous subsection. Here, we present the latest standardized definition of *IP flow* presented in RFC 7011[8] [102].

> *A Flow is defined as a set of packets or frames passing an observation point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties. Each property is defined as the result of applying a function to the values of:*
>
> 1. *one or more packet header fields (e.g., destination IP address), transport header fields (e.g., destination port number), or application header fields (e.g., RTP header fields).*
> 2. *one or more characteristics of the packet itself (e.g., number of MPLS labels, etc.).*
> 3. *one or more of the fields derived from Packet Treatment (e.g., next-hop IP address, the output interface, etc.).*
>
> *A packet is defined as belonging to a Flow if it completely satisfies all the defined properties of the Flow.*
>
> RFC 7011 [102]

An observation point is a location in a network, where packets are observed [102]. The traffic access methods for packet observation are described in Section 3.1. A location of observation point determines the information available in monitored data. An observation point located at connection point to internet service provider provides information on all outgoing/incoming network traffic from/into a network but does not give any information on traffic among elements within the monitored network itself. To observe network traffic inside a network, an observation point needs to be located within a network. For the measurement process, observation points can be aggregated into *observation domains*, e.g., several interfaces of a router line card (observation points) are grouped into the observation domain (a router).

All properties common to all packets in an IP flow are called *flow keys*. RFC 7011 defines flow keys as follows:

> *Each of the fields that:*

8. We use term *IP flow* that refers to RFC 7011 flow definition to clearly distinguish from common usage of the term flow (e.g., traffic flow, information flow).

> - *belong to the packet header (e.g., destination IP address), or*
> - *are a property of the packet itself (e.g., packet length), or*
> - *are derived from Packet Treatment (e.g., Autonomous System (AS) number),*
>
> *and that are used to define a Flow (i.e., are the properties common to all packets in the Flow) are termed Flow Keys.*
>
> RFC 7011 [102]

The most common flow key used to identify an IP flow is the five-tuple of the source and destination IP address, source and destination transport port, and transport protocol. This five-tuple represents an abstraction of a unidirectional communication. It is important to note, that a definition does not explicitly specify, what a packet header is. We believe that packet headers mainly refer to packet headers of L2 – L4 layer of OSI model[9].

Information on IP flow is exported in the form of a IP flow record. The IP flow record is defined as follows:

> *A Flow Record contains information about a specific Flow that was observed at an Observation Point. A Flow Record contains measured properties of the Flow and usually contains characteristic properties of the Flow.*
>
> RFC 7011 [102]

The measured properties are, for example, the total number of bytes and packets in an IP flow, byte and packet rates per second. The characteristics properties of the IP flows mentioned in the IP flow record definition refer to IP flow keys, e.g., the above mentioned traditional five-tuple.

In general, an IP flow is an abstract representation of a set of packets that describes their common properties. An IP flow record is then an IP flow enriched by its measured properties. The processes of IP flow and IP flow record creation are described in Section 3.3.

3.2.3 Current Trends

The current trends influencing network IP flow monitoring originate mainly in achieved technological advances, and in novel approaches to the network usage. We discuss following trends that affect IP flow monitoring domain - an increased computational power of hardware, increasing volume of network traffic, rise of encrypted traffic, focus on privacy, and shift to the cloud environment. For each mentioned trend, we present an impact on the IP flow.

- **Increased computational power** – An advantage of IP flow over DPI network monitoring is the high throughput and the volume of network traffic that can be monitored. Since DPI analyzes a whole packet, its throughput is considerably lower than the throughput of IP flows that process only L2 – L4 packet headers. Nevertheless, the improved computational performance of current hardware and utilization of field programmable gate arrays (FPGA) in network cards enable new possibilities for an IP flow export. The current IP flow exporting tools can process network traffic at the rate over 100Gbps [116], currently targeting to break 200Gbps throughput on commodity hardware. Apart from the increased throughput, the improved computational performance enabled flow exporters to process more information than L2 – L4 packet headers only. Export of information from

9. The Open Systems Interconnection (OSI) model describes and standardizes the communication functions of computing system regardless an underlying technology. Communication is partitioned into abstraction layers, starting with the physical layer (L1), ranging to the application layer (L7).

application protocol (L7 layer) is gaining attention. *Next-generation IP flows* includes information from most common protocols, e.g., HTTP, DNS, and SNMP. Generally speaking, a volume of information exported in IP flow is shifting towards a volume of information exported by DPI.

- **Increased volume of data** – The volume of network traffic is continuously increasing [81]. Combined with the increased computational power of hardware, the volume of IP flow exported, and information contained in the IP flows are enormous (a mid-size network of 15 000 active IP addresses can produce 10 GB of IP flows records a day at an average observation rate 8 000 IP flows/second). Along with this data boost, a demand for information mining from the exported data increases as well. Next generation databases for large-scale storage of IP flows are under research [117] and novel and optimized methods for large volume IP flow analysis are developed and improved [118, 119]. The big data trend might even change the IP flow record definition itself in the future. Novel flow records are tested as part of a storage optimization efforts. The novel records share a common part for each connection, i.e., a traditional five-tuple and the common part is supplemented with an "*event part*" that represents individual connections on the application layer. A flow record size is reduced as a common part of all connections is stored only once.

- **Privacy issues** – With an increasing volume of information exported in IP flow, the privacy issues need to be considered. The efforts for providing a legal framework that would protect the privacy of monitored network users has emerged. General Data Protection Regulation (GDPR), EU Regulation 2016/679, intents to unify and improve data protection for EU citizens via regulation of personal data storage and processing. The Federal Trade Commission Act (15 U.S.C. §§41-58) (FTC Act) is a federal consumer protection law in the United States of America that prohibits unfair or deceptive practices and has been applied to offline and online privacy and data security policies. Another US privacy acts refer directly to electronic communications – The Electronic Communications Privacy Act (18 U.S.C. §2510) and the Computer Fraud and Abuse Act (18 U.S.C. §1030) regulate the interception of electronic communications and computer tampering, respectively [120]. IP flows belongs within personal data affected by these regulations. Security and privacy of stored IP flows shall be improved significantly. A pseudonymization process shall be applied to IP flows. There currently exist anonymization techniques for IP flow. Whether they provide sufficient anonymization level for increased privacy requirements, that is a question.

- **Encrypted traffic** – The increased computational power supports a trend to analyze additional application headers from packets. However, the encryption of network traffic hiders the retrieval of additional information form a packet as information form application headers become unavailable due to encryption. A continuous increase of encrypted traffic share [94] might result in the fact, that encrypted traffic will become a majority in the near future. The IP flows paradigms is expected to return to its origins – only information from traditional five-tuples flow keys will be processed.

- **Cloud sevices** – According to Oracle's cloud prediction [121], about 60 % of IT organizations will have moved their systems management to the cloud. The development and testing workloads are predicted to be completely migrated to cloud by 2025. Small businesses are also expected to move their agenda to a public cloud to reduce their costs and keep up with the latest technological trends. In general, a cloud will contain a significant portion of services in the future. The migration to the cloud also affects network monitoring and IP flows. Thanks to the virtual cloud environment, a novel network traffic access methods are available, and new information can be harvested (e.g., data from cloud hypervisors, host-based traffic access points). Apart from innovations linked to the cloud environment, we expect also cost/effective optimizations of IP flow concept. As business pay for each operation and data stored in the cloud, the demand for cost-effective IP flow monitoring will increase. Types and volume of collected information will be optimized (i.e., reduced while keeping original information value as much as possible).

3.3 IP Flow Monitoring

This section aims to explain how IP flow records are created and handled. The understanding of IP flow record creation process eases understanding of specifics and nuances of IP flow analysis that are discussed later in this thesis. Organization of this chapter is inspired by a article by Hofstede et al. [88] and by RFC 7011 [102].

A general IP flow monitoring workflow is proposed in RFC 5470 [122] and consists of processes depicted in Figure 13. The packets are observed at one or more observation points. A device called *exporter* handles the creation and export of IP flow records. A *collector* receives the IP flow records from one or more collectors and stores them for further analysis. Exporters and collectors are often dedicated devices to achieve high performance. Nevertheless, these two devices can be combined in a single device to save hardware costs. The provided schema represents a general workflow for the majority of IP flow monitoring installations. Following subsections describe each process linked to IP flow creation and storage separately with an accent to information relevant to this thesis. The data analysis process is inspected in a Section 3.4.

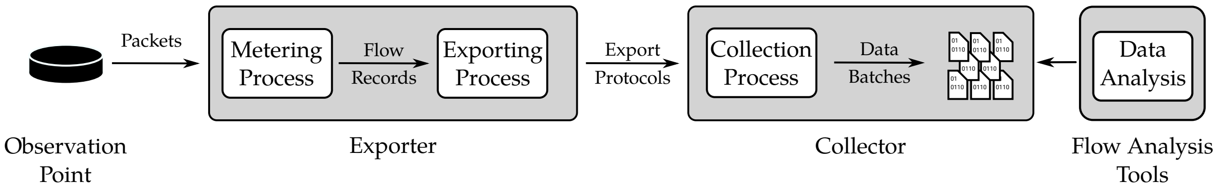


Figure 13: IP flow monitoring workflow.

3.3.1 Packet Observation

Packets are observed at observation points via traffic access methods described in Section 3.1. The packet observation process comprises of two main phases – packet capture and timestamping. These two phases are executed for each observed packet. Optional phases, such as packet truncation and packet sampling, are included mainly to optimize packet capture performance.

First, a packet is captured from a line by Network Interface Card (NIC). Several checks are performed (e.g., checksum error) and the packet is passed via driver to a standard TCP/IP stack in kernel space. From the kernel space, the packet is served to userspace to libraries and application programming interfaces (APIs) designated for capturing network traffic. *Libpcap* or *libtrace* are available libraries for Linux operating systems, *WinPcap* for Windows. These libraries provide both capture and filtering functionality and serve the packets for a Metering Process to create IP flow records. The described packet capture process utilizes general-purpose networking. Therefore, it suffers from suboptimal performance unsuitable for high-speed networks [88]. Optimization techniques reduce costs of handling a packet from NIC to the Metering Process by, e.g., bypassing a kernel space, or leveraging of the hardware-acceleration cards that use FPGAs to reduce resources consumption during the packet capture. More optimization techniques for the high-speed packet capture are described in [88].

The second phase is timestamping. During the timestamping, each packet is assigned with an accurate timestamp of observation. The accurate packet timestamping is essential for all subsequent packet processing including Metering Process and IP flow analysis. Inaccurate timestamping results into a creation of incorrect IP flow records or biased results of IP flow analyses. The timestamping is performed either in hardware or software. The hardware-based timestamping is more accurate than the software-based one, as it does not suffer from hardware-to-software forwarding latency. The hardware solutions massively rely on GPS time synchronization and provide timestamps with an accuracy below one micro-second [123]. However, the hardware solutions are available mostly on specials NICs that are expensive for common usage. Due to costs of the hardware, software-based timestamping prevails. The software-based timestamping solutions implement time format with a microsecond resolution, so the accuracy of timestamps cannot go beyond this resolution. [123]. Hofstede et al. [88] reports the accuracy in the order of 100 microseconds when using a Network Time Protocol (NTP) for clock synchronization.

The optional phases that are used to reduce a workload for further workflow processes are packet truncation, sampling, and filtering. During the packet truncation process, only those bytes needed for further IP flow processing are kept, and the others are truncated. As IP flows are created from packet headers, usually the whole data part of a packet is dropped, which significantly reduces data volume to process and increases the throughput of the workflow.

The packet sampling for network monitoring purposes is described in RFC 5475 [124]. According to the RFC, packet sampling aims to select a representative subset of packets that allow accurate estimates of properties of the unsampled traffic. As only a representative subset of the packets is selected, the packet sampling reduces the volume of data and is used whenever it is expected that the volume of incoming packet will overload the Metering Process. Two main types of packet sampling are distinguished based on the method of packet selection process [124]: *systematic* and *random*.

Systematic packet sampling selects packets according to a deterministic function. The deterministic functions are either time-based or event-based. The time-based selection function selects packets every t seconds, while the event-based function select ever $n - th$ packet. As the selection function is deterministic, using the function involves a risk of obtaining a biased subset of network traffic. A subset created by time-based selection function will be likely biased toward regular network traffic. Random packet sampling selects the packets by a random process. The packets are selected using following two main random functions: *n-out-of-N* and *probabilistic* [124]. The n-out-of-N function splits the network traffic into subsets of N packets. From each subset n packets are randomly chosen. The probabilistic function selects each packet with probability p based on predefined statistical distribution. The probability p is usually based either on uniform distribution (each packet is selected with the same probability), non-uniform distribution (packets are selected with different probabilities), or non-uniform flow state dependent (existing flows are taken into account when selecting packets). The random sampling is preferred to the systematic sampling as random sampling does introduce less bias to the selected traffic subset than the latter one.

Packet filtering is used to select network traffic of interest. According to RFC 5475 [124], the packet filtering separates all the packets having a particular property from those not having it. The packet can be filtered either by property match filtering (e.g., a defined IP address range, protocol type, network traffic direction) or by hash-based filtering (packets with a certain hash are selected). The hash-based filtering is used, e.g., when tracking a packet path across different observation points.

3.3.2 Metering Process

Metering Process is responsible for aggregation of individual packets into IP flows and for the creation of IP flow records. A cornerstone of the Metering Process is a *flow cache*. A flow cache is a table, where all active IP flows are stored. The flow cache tables contain entries composed from the key and non-key fields. The key fields represent IP flow keys used for aggregation of packets into IP flows. Non-key fields represent additional information in an IP flow record, e.g., the sum of bytes, packets. The key fields are usually the traditional 5-tuple mentioned in IP flow definition in the previous section. The traditional 5-tuple creates unidirectional IP flows. A *bidirectional IP flows* are used in cases, where a direction is not important, and we focus on the connections (source/destination pairs). As the source and destination flow key are still present in a flow cache, a special cache is needed for pairing request and response flows to create a bidirectional IP flow.

The update process of a flow cache process the packets passed from packet observation phase. With each passed packet, the flow cache is searched for entries with the corresponding flow keys. If the corresponding entry is found, the non-key values of the entry are updated. In case, there is no corresponding entry in the flow cache a new entry is created using flow keys from the passed packet. The cache entry expires when a corresponding IP flow is considered to have terminated. We distinguish the following reasons for cache entry expiration [122]:

- **Inactive timeout** – An IP flow is terminated when no packets belonging to the IP flow are coming for a predefined period. The predefined period of IP flow inactivity is referred to as inactive timeout. The inactive timeout usually ranges from 15 seconds to 5 minutes based on default settings [125].

- **Active timeout** – Active timeout for expiration is triggered when an IP flow reaches the maximum predefined lifetime. The active timeout serves to split long-lasting connection into several IP flows to provide up-to-date network visibility. If the active timeout was not present, the long-term connection would be expired after their termination based on the inactive timeout or natural expiration. Information on the IP flow would be available for an operator after an unacceptably long time (some connections may last for several days or weeks). The values of active timeouts usually range from five minutes to thirty minutes[10]. The flow records expired by the active timeout are typically not removed from the flow cache. The record is kept, and non-key values are reset.

- **Natural expiration** – Some connections contain markers of their termination in protocol by default. In a TCP connection, FIN or RST flags are used to signal termination of the connection. A flow cache can use these flags as an IP flow expiration trigger. This type of IP flow expiration introduces a risk of a premature expiration of IP flow. Although FIN and RST flags terminate a connection meaning no more data should be transferred, it often happens that a delayed data packets are sent/received even after FIN and RST flags are observed (e.g., due to network latency, incorrect protocol implementation).

- **Resource constraints** – In the case a flow cache becomes full, selected IP flows are expired immediately to free space for new IP flows. Alternatively, other parameters for IP flow expiration are adjusted to free the flow cache (e.g., inactive timeout is reduced to expire IP flow records earlier).

- **Emergency expiration** – This category covers the unexpected situations that can occur during the Metering Process. In case of such situation, all IP flows are considered expired, and the cache is flushed. These situations are, for example, a significant change in exporter configuration, change of the system time after a time synchronization, or exporter shutdown.

10. 30 minutes is a default active timeout for Cisco's NetFLow [125].

After an IP flow is expired by one of trigger mentioned above, the whole record is removed from a flow cache and handed over for the Exporting Process. A sample of IP flow record for a HTTP communication is presented in Figure 14.

```
Flow start    Duration  Proto     Src IP Addr:Port        Dst IP Addr:Port    Flags  Packets  Bytes
09:41:21.763   0.101    TCP     72.16.96.48:5094 ->  55.85.135.147:80     .AP.SF    4      715
09:41:21.893   0.031    TCP   55.85.135.147:80   ->    72.16.96.48:5094   .AP.SF    4      1594
```

Figure 14: IP flow records representing a HTTP communication.

During a hand over to the Exporting Process, an IP flow record sampling and filtering take place. Analogous to packet sampling and filtering, the goal of the IP flow sampling and filtering is to reduce processing requirements of subsequent phases of IP flow monitoring workload by reducing the number of IP flow record to process. The IP flow sampling aims to create a representative IP flow record sample, that would maintain all main characteristics of the sampled IP flow record set. IP flow sampling and filtering are described in RFC 7014 [104]. The RFC distinguishes two main IP flow sampling techniques (similar to packet sampling): *systematic* and *random* sampling. During the systematic sampling, an IP flow is based on deterministic selection function that periodically selects n-th IP flow record. The random sampling techniques select n-out-of-N elements from the N IP flow to be exported. Another method for random IP flow selection, probabilistic IP flow sampling select each IP flow record based on the predefined probability. The IP flow record sampling functions (both systematic and random) should be selected and used wisely, as they might introduce bias into a sampled data. The sampling method proposed by Estan and Varghese in [126] is biased towards large flows [125]. Duffield, Lund, and Thorup propose *smart sampling* in [127, 128] that takes in account the heavy-tailed distribution of IP flows and selects IP flow with a probability dependent on IP flow size to create a usage-sensitive billing IP flow sample.

The IP flow record filtering uses a *property match* or hash-based filtering function [104]. The IP flow record filtering based on property match works similar to packet filtering with the difference that only IP flow keys are used here for matching, instead of packet fields. The hash-based filtering maps IP flow keys into a predefined hash range. If a hash of IP flow is within a predefined subset (i.e., filtered traffic) of the hash range, the associated IP flow record passed for further processing.

The Metering Process is a cornerstone phase of the IP flow monitoring workflow. The settings of the Metering Process substantially influences further next phases of the IP flow monitoring workflow including subsequent IP flow analysis tasks. The active/inactive timeout settings influence number IP flow created and their properties regarding size, number of packets, and distribution. The flow cache size determines a maximum number of stored IP flow records and consequently, the number of prematurely exited IP flow records. The selection of IP flow keys defines the aggregation process of packets into IP flows. Hence, knowledge of the Metering Process settings is essential for all IP flow data analysts, as the settings are the determinant of observed IP flow data properties.

3.3.3 Exporting Process

Exporting process handles sending IP flow records generated by one or more Metering Processes to one or more Collecting Processes [102]. The IP flow records are aggregated into messages and send via a selected protocol to data collection device. The structure of a message is usually derived from a protocol used for IP flow export. There have emerged several protocols and their versions for IP flow export since the initial idea of flow information export (see Subsection 3.2.1). The most common protocols are NetFlow v.5, v.9, and IPFIX. We describe in greater detail IPFIX-based IP flow export to demonstrate the basic properties of the Exporting Process. We select IPFIX-based export for closer description as IPFIX is considered as current state-of-the-art exporting protocol. Exporting Processes using other protocols mostly share similar properties, only the message structure, and available fields for export differs.

An IPFIX message comprises a message header and message data. The message header has a fixed structure and contains meta information of the message – protocol version number, message length, export time, sequence number, and observation domain ID. The message data contains *sets* of IP flow records. A set contains set ID, length, and one of three following set types: *template set*, *data set*, and *options template set*. A template set contains information on the structure of data records carried in the data set. A data set carries the actual IP flow records according to a template description in the template set. A options template sets are used for additional information export to collectors, e.g., control plane data or flow keys used in a Metering Process. The number of IP flow record in an IPFIX message is usually set to match the Maximum Transmission Unit (MTU) of a line to a collecting device to avoid message fragmentation. The IPFIX message is sent over a network to a collecting device when the message is full of IP flow records. The exporting process aims to send as many IP flow records in a message as possible without fragmentation to achieve optimal utilization of a link. Optionally, a smaller size of IPFIX message can be set to achieve a prompter export of IPFIX messages at the cost of a higher number of packets sent over the network.

The created IPFIX message is transported over the network to a collecting device using transport protocol. IPFIX messages support transport over multiple protocols. The currently supported transport protocols are User Datagram Protocol (UDP), Transmission Control Protocol (TCP), and Stream Control Transmission Protocol (SCTP). Apart from these standard transport protocols, IPFIX supports export of the messages into files (IPFIX File Format [129]) that can be sent using application protocols enabling file transport (e.g., Secure Shell (SSH), Hypertext Transport Protocol (HTTP), or File Transport Protocol (FTP)). The most widely used and implemented protocol is UDP. Since UDP uses a simple connectionless communication, it is easy to implement and deploy. However, the protocol does not provide any guarantee for packet delivery. The unreliable delivery means that some IPFIX messages are not delivered, i.e., a template sets might be lost during the transport or a significant portion of data can be lost during a burst of exported IPFIX messages (e.g., during a (D)DoS attack). TCP provides the desired reliable message transport. The TCP protocol is also relatively easy to implement. It also provides a binding to Transport Layer Security (TLS) protocol to provide encryption of IPFIX messages and privacy of their transport. An often mentioned disadvantages of the TCP protocol are a relatively harder setup compared to UDP protocol and back pressure effect during overload situations. The SCTP is a preferred transport protocol for IPFIX implementations. It should be used in deployments where Exporters and Collectors are communicating over links that are susceptible to congestion [102]. The SCTP provides a congestion-aware packet delivery. The packet boundaries are preserved so a collector can process individual IPFIX messages instead of a stream of bytes when messages are sent over TCP. Despite these advantages, SCTP is the least deployed of the three main protocols mainly due to its difficult implementation [88].

3.3.4 Collection Process

After the export of IPFIX messages, the monitoring workflow enters the collection phase. The exported IP flow records are received by a Collecting Process that handles their pre-processing and storage. The Collecting Process should be able to obtain the IP flow information passing through multiple network elements within the data network. A device hosting one or more Collecting processes and associated storage is called a *collector* [102].

In general, there exist two main types of data storage used for IP flow records - *primary* (temporary) and *secondary* (persistent) data storage. Primary storage, usually with a small capacity, is used for fast data processing or caching. In IP flow monitoring workflow, the primary storage is often used when data needs to be analyzed on-the-fly, e.g., for generation of time-series used for visualization purposes [88].

The secondary memory is used for long-term data storage and is significantly slower than primary storage. It is used for IP flows retention[11] and as a data source for further IP flows analyses. We describe four main types of secondary storage used for IP flow monitoring. Three of the types are described in [88]; we add the fourth type that is gaining attention lately and deserves it's own category, in our opinion. The speed of secondary storage and query flexibility are the main discriminants among secondary storage types.

- **Flat-files** are very fast for writing and reading, on the one hand. On the other hand, they provide limited querying possibilities, and a specialized tool is usually needed for querying different flat-file types. The representative of flat-file storage of IP flow record is *nfdump toolset*[12], namely *nfcapd*, that stores IP flow records into a binary files.

- **Row-oriented databases** represent the main-stream databases used in Database Management Systems (DBMS) such as *MySQL*[13], *PostgreSQL*[14]. The data are stored in rows in tables and accessing a data means reading all row. These databases offer full query flexibility. Their performance is, however, significantly lower both regarding data insertion or query response times compared to other described storage types. Nevertheless, being the mainstream databases, their performance is being constantly improved. Row-oriented databases are being used by some of IP flow monitoring vendors as a supporting database to flat-files, e.g., *nProbe*[15] [130].

- **Column-oriented databases** store the data into columns. During a query, only required columns are accessed which decreases the query response time. The column databases seem suitable for IP flow record storage as individual record's keys can be naturally stored in separate columns. During a query, only relevant keys are read instead of the whole IP flow record. Indexes over each column can be pre-computed, which further increases the querying speed. *FastBit*[16] is a representative of column-oriented databases used for storing IP flow records.

- **Next-generation databases** have emerged recently. Their expansion is associated with the advances in distributed cloud computing and big data processing. The next-generation databases abandon the traditional Relational Database Management Schema (RDBMS) and introduce NoSQL (Not only SQL) approach. The databases are schema-free, non-relational, distributed, scalable, and support other then SQL-like query languages. These databases are created for storage of large volume of data. An example of a database that can be used as IP host relations storage is a *neo4j*[17] graph database. Among databases aiming at scalability, distribution, and speed, belong *Elasticsearch*[18] or *Hadoop*[19].

11. Data retention, mainly for internet service providers, is recently being required by law in increasing number of countries.
12. `https://github.com/phaag/nfdump`
13. `https://www.mysql.com/`
14. `https://www.postgresql.org/`
15. `https://www.ntop.org/products/netflow/nprobe/`
16. `https://sdm.lbl.gov/fastbit/`
17. `https://neo4j.com/`
18. `https://www.elastic.co/`
19. `http://hadoop.apache.org/`

The performance of the different storage types and their suitability for IP flow monitoring have already been discussed in the literature. Hofstede et al. evaluated the performance of *nfdump tool* compared to MySQL management system in [131]. The *nfdump tool* outperformed the MySQL in all comparisons covering different database operation's response times (listing, filtering, grouping). Velan evaluated available query tools for their suitability for IPFIX flow collector in [132]. Among others, performance of query tools *SiLK*[20], *nfdump* and *fbitdump*[21] were compared proving the advantages of column databases used by fbitdump. Experiences with distributed big data frameworks (*Hadoop*, *Vertica*[22], and *Elasticsearch*) are described by Zadnik, Krobot, and Wrona in [133].

When storing IP flows, a privacy aspect needs to be considered. The requirement for keeping the privacy of end users in a network is already implemented in legal and regulatory actions in several states. Worth mentioning is the General Data Protection Regulation (GDPR), EU Regulation 2016/679 that directly affects the IP flow record storage via regulation of personal data storage and processing. The IP flows do not contain as much privacy-sensitive information compared to packet traces as only packet headers, not the communication content, are collected. However, the privacy exposure of end users has increased recently with the introduction of next-generation IP flows that contain information from application layer headers. Next-generation IP flows enable to track IP activity in World Wide Web using headers from HTTP of DNS network traffic[23]. Technically, it is not always possible to unambiguously link an identity directly to each IP flow, as only IP addresses are collected. Nevertheless, IP addresses are still considered by the regulations as an identifier of an individual. Therefore, the privacy protections apply also to IP flows, mainly to IP addresses anonymization.

Summary of the best practices for network traffic anonymization is provided by the Center for Applied Internet Data Analysis (CAIDA) in [134]. The summary is accompanied by a bibliography on network traffic anonymization that covers years 1992-2013 [135]. The anonymization of IP addresses is a trade-off between the level of privacy and data utilization. No anonymization technique would ensure complete privacy and keep the information value of the data intact. In the wild, Crypto-PAn algorithm [136] is used for IP address anonymization. The Crypto-PAn algorithm preserves the IP addresses prefix and creates anonymized networks. This approach allows for IP flow based analyses at the cost of restricting the analysis to an anonymized network [88]. Implications of network traffic anonymization are also investigated. Burkhart et al. investigated the attacks on traffic anonymization by injecting artificial network traces that help to retrieve original traces from the anonymized ones [137]. Coull et al. inspects the web browsing privacy in IP flow traces [138]. They develop new approaches to identify target web pages within anonymized IP flow traces. The challenges for IP anonymization are summarized in [139].

3.3.5 Timeline

This section investigates delays that emerge in the IP flow monitoring workflow. The delay is a natural component of the workflow and it is present at different stages of the workflow. However, the delay limits the performance and applicability of IP flow monitoring in several areas. In the security area, the delay increases the speed of response to a potential attack. Due to the delay introduced in IP flow measurement, an attack detection method receives data late and the attack is detected with a delay of, e.g., several minutes. Given the speed of attacks in orders of several seconds, the attack may be even over, and it is not possible to capture live attack traffic identified by detection anymore. The delay is an issue also in a performance monitoring area. For example,

20. `https://tools.netsa.cert.org/silk/index.html`
21. `https://github.com/CESNET/ipfixcol/tree/master/tools/fbitdump`
22. `https://www.vertica.com/`
23. Host identification in network traffic is part of our research and is discussed in detail in Chapters 4 and 5 of this thesis.

a late identification of a rapid performance decrease of an e-shop application might result in the financial loss as customers are not served.

Significant events occurring in the IP flow monitoring workflow are depicted in Figure 15. Each event is associated with a relevant timestamp to be able to illustrate the delays present in the monitoring process. We assume that t_0 is a timestamp when an event occurs, i.e., when we can observe an IP flow. We deliberately omit the fact, that the actual event occurs earlier, as it takes time for a packet to reach an observation point of IP flow measurement. This delay is out of the scope of IP flow measurement as it cannot be influenced by the IP flow measuring processes[24].

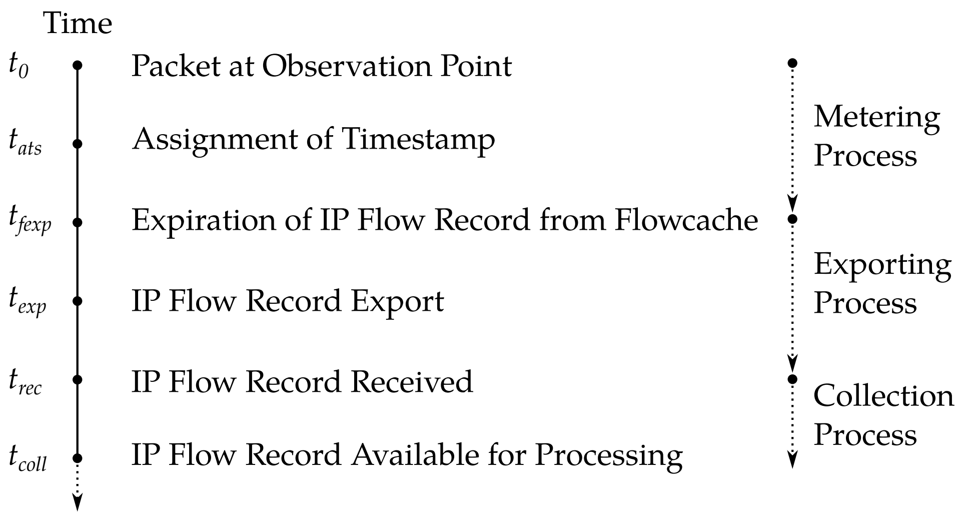


Figure 15: Timeline of IP flow monitoring workflow.

The timestamp $t_{ats} = t_0 + \Delta_{ats}$ represents a point in time when a timestamp is assigned with a timestamp in NIC and is sent to a flow cache. The time delay Δ_{ats} introduced in this step origins in hardware-to-software forwarding latency. The delay ranges from orders of micro to nanoseconds [123]. This delay is for our work negligible, as it is below the human cognition abilities.

The timestamp $t_{fexp} = t_{ats} + \Delta_{fexp}$ represents time when an IP flow record is exited from a flow cache. At t_{fexp}, the metering process is complete and the exporting process starts. The Δ_{fexp} is time needed for creating IP flow record and is dependent on a flow duration t_{FD}, active timeout t_{AT}, and inactive timeout t_{IT} settings as follows:

$$\Delta_{fexp} = f(t_{FD}, t_{AT}, t_{IT}) = \begin{cases} t_{AT}, & \text{if } t_{FD} \geq t_{AT} \\ t_{FD} + t_{IT}, & \text{if } t_{FD} < t_{AT} \end{cases}$$

where $t_{AT} \geq t_{IT}$. This equation works for IP flow record expiration triggered by inactive or active timeouts. When other flow cache expiration types mentioned in Section 3.3 are applied, we assume $t_{IT} = 0$. Based on the equation, we can derive theoretical maximum and minimum values of Δ_{fexp}:

$$\min(\Delta_{fexp}) = \min(f(t_{FD}, t_{AT}, t_{IT})) = t_{FD}$$
$$\max(\Delta_{fexp}) = \max(f(t_{FD}, t_{AT}, t_{IT})) < t_{AT} + t_{IT}$$

In other words, the minimum delay of metering process is the duration of IP flow. The minimum delay is achieved when other than active and inactive flow cache expiration types are applied

24. Only way, how to reduce the delay is to move the observation point closer to the event origin location.

(i.e. $t_{IT} = 0$) and IP flow duration is close to 0. The maximum possible delay is upper bounded by $t_{AT} + t_{IT}$, i.e. the IP flow is over right before active timeout and inactive timeout is then applied.

The timestamp $t_{exp} = t_{fexp} + \Delta_{exp}$ represents a point in time when a IP flow message of IP flow records is exported from a flow probe to a collector. The time needed to create an IP flow message Δ_{exp} is dependent on Maximum Transition Unit (MTU) of a network, rate of expired IP flow records per second FR, and average IP flow record size SR. The time needed to create an IP flow message can be then described as

$$\Delta_{exp} = f(MTU, SR, FR) = \frac{MTU}{SR * FR}$$

We do not difference individual IP flow record sizes for the sake of simplicity and take an average IP flow record size instead. The higher rate of expired IP flow records leads to lower Δ_{exp}. Analogous, the higher is the average IP flow record size (e.g., more IP flow keys is exported in a record), the lower time is needed to export an IP flow message.

The timestamp $t_{rec} = t_{exp} + \Delta_{rec}$ represents a point when an IP flow record in an IP flow message is received by a collector. At this point, an exporting process is complete and a collection process of IP flow monitoring workflow begins. The time needed to deliver an IP flow message from a flow probe to a collector Δ_{rec} is equal to a packet deliver time and is defined as follows:

$$\Delta_{rec} = \frac{PS}{BR} + \frac{D}{S}$$

where PS is a packet size, BR is a bit rate, D is a distance line between a probe and collector, and S is the propagation speed of the link. The higher is the bit rate and propagation speed of the link the lower time is required to deliver an IP flow message from a probe to a collector. The lower packet size and shorter distance speed up the delivery of IP flow message as well.

The timestamp $t_{coll} = t_{rec} + \Delta_{coll}$ represents a point in time when an IP flow record is available for processing and analysis. At this point, the collection process is finished, which also completes the IP flow monitoring process. The duration of an IP flow collection process Δ_{coll} is influenced by collector database implementation. Regarding the collector with a row-oriented database, the IP flow record is available nearly instantly, and the Δ_{coll} represents the only time needed to insert the IP flow record into the database. The insert time depends on the hardware configuration of a collector (e.g., type of hard drive) and the rate of IP flow records to insert. Regarding the collectors using flat files as data storage, the information is available as soon as the file is complete. For example, *nfcapd* tool stores the IP flow records into files containing five minutes of IP flow records. In the worst case, the Δ_{coll} is equal to five minutes given this setting. The column-oriented databases and next generation show similar delay as row-oriented databases as an IP flow record is available instantly after the insertion into the database.

Combining the above-described timestamps, we obtain the total delay of the IP flow monitoring workflow $\Delta_{monitoring}$ as follows:

$$\begin{aligned}
\Delta_{monitoring} &= t_{coll} - t_0 \\
&= \Delta_{coll} + t_{rec} - t_0 \\
&= \Delta_{coll} + \Delta_{rec} + t_{exp} - t_0 \\
&\vdots \\
&= \Delta_{coll} + \Delta_{rec} + \Delta_{exp} + \Delta_{fexp} + \Delta_{ats} + t_0 - t_0 \\
&= \Delta_{coll} + \Delta_{rec} + \Delta_{exp} + \Delta_{fexp} + \Delta_{ats}
\end{aligned}$$

The total delay of the workflow is the sum of delays of different stages of the workflow. The differentiation of the workflow to the phases and definition of the delays enable us to focus on the phases individually and minimize the delays phase by phase. The Table 1 shows the contributions of individual delays to the total delay. We can observe, that the most significant contribution makes the time needed to create an IP flow Δ_{fexp} followed by the collection process duration Δ_{coll}. In a real-world deployment, the maximum possible delays are around 10 minutes, as active timeouts are usually set to 5 minutes and the frequently used flat-file collectors are set to collect IP flow records into five-minute files. A combination of these two settings leads to the mentioned 10 minutes delays.

Delay	Duration of a delay (orders)
Δ_{ats}	< microseconds
Δ_{fexp}	minutes
Δ_{exp}	milliseconds
Δ_{rec}	milliseconds
Δ_{coll}	from milliseconds to minutes

Table 1: Overview of the delay sizes.

3.3.6 Open Issues

We identify the following open issues of IP flow monitoring workflow: *performance*, *scalability*, *response time*, and *changing IP flow paradigm*. The open issues derived from the current trends in IP flows (see Subsection 3.2.3) or they are mentioned in literature [88, 140]. Some open issues are everlasting, such as performance issues, other issues are relatively new, driven by the latest advances, such as changing paradigm of IP flow usage. Next paragraphs discuss the open issues in more detail.

- **Performance** – The performance open issues are associated with the increasing volume of network traffic to monitor and with increasing volume of information analyzed from a packet. Each phase of the IP flow monitoring workflow needs to be optimized to enable us to process the large volume of network traffic and data. Enhanced network probes with high-performance flow caches and sampling techniques are demanded to prevent from flow exporter overload. Similarly, the efficiency of the IP flow transport protocol can be improved to prevent transport overload caused by a high volume of exported IP flows. The collector performance is also the prevailing open issue a large-size IP flow records are required to be stored and searched efficiently. Current solutions should to adapt to new advances in IP flow export and should enable us to store new information contained in next-generation IP flow records. The IP flow monitoring workflow also contains artifacts, that come at the expense of accuracy of IP flow measurement [88]. The IP flow artifacts are timing artifacts resulting in incorrect timestamps, data loss (e.g., due to flow cache overload), or invalid counters [141].

- **Scalability** – Demand for a scalable solution emerged when network monitoring community realized, that the volume of network traffic will continuously increase in the future (possibly in at exponential rate [81]). Distributed monitoring infrastructures containing distributed IP flow collectors are currently under research. The distributed collectors are designed to scale with the increasing performance demands while maintaining the original functionality.

- **Response times** – There are several time delays during the IP flow monitoring workflow on the one hand. On the other hand, industry and business deployments demands for near-real-time accounting and network visibility. The minimization of these delays, e.g., immediate IP flow export and continuous updates of the IP flow records, could provide a near-real-time IP flow monitoring workflow.

- **Changing paradigm** – The technological advances in networking and the shift from in-house applications and services to cloud impose demands on IP flow monitoring workflow. IP flow monitoring needs to be able to capture information from a network that uses a variety of networking mechanisms, such as Virtual Private Networks (VPN), generic routing encapsulation (GRE tunnels), IPv4 encapsulation of IPv6 traffic, and so forth. The shift to cloud services modifies the monitoring workflow, e.g., a suitable location of observation point needs to be discussed. The shift to the cloud also opens cost-performance optimization efforts. A balance between the volume of monitored data, available information, and costs of monitoring is searched to reach a suitable solution for different types of business.

3.4 IP Flow Record Analysis

IP flow record analysis aims to retrieve information from data captured in IP flow records created during IP flow monitoring. Hofstede et al. [88] identifies three main areas where IP flow record analysis is employed: analysis and reporting, performance monitoring, thread and anomaly detection. Li et al. [142] presents different, yet analogous, perspectives on network flow applications: network measurement and analysis, network application classification, user identity inferring, security awareness and intrusion detection, and issues of data error. In this section, we first focus on the analysis process itself. We describe main approaches to IP flow record analysis and their specifics. Next, we provide a brief overview of each of the area of IP flow record application based on Hofstede et al. [88] categorization.

3.4.1 Workflow

The IP record analysis workflow is rather specific for each analysis use-case. Nevertheless, it is possible to derive the main types of analysis workflow general enough to be applied to all mentioned analysis use-cases. The general workflows can be divided into categories by regularity of querying and by approach to data processing. The regularity-derived workflow categories are *regular* and *on-demand*. The workflow categories related to data processing approaches are *batch-based* and *stream-based* workflows.

Regular analysis workflow serves for regular, repeating analyses. Such analyses are usually automated and are used for reporting, anomaly detection, or statistics pre-computation, for example. The queries to data store are known in advance. Data to analyze are predefined as well. Given these facts, the stored data can be pre-processed to lower query response time. Moreover, the queries can be scheduled to spread the workload in time. The analysis is usually initiated by a time job scheduler, e.g., cron in Unix-like systems. The analysis is executed on relevant data and results are stored or handled over for visualization to an operator.

On-demand analysis workflow differs from the regular one by the initiation of the analysis. The analysis is initiated by a user and it can happen at any time. Analysis query is not predefined. Data used for analysis can cover an arbitrary time span. The on-demand analysis serves mainly for additional exploration of captured data. This analysis is usually used in particular cases when regular analyses do not offer sufficient information. The on-demand analyses are usually more computationally demanding than the regular ones, as raw data need to accessed to find an answer to a query.

Batch-based and stream-based workflows differ substantially in the way how data is processed. The batch-based workflow processes data in batches. The size of a batch is defined by the way, how data is stored in a collector. Collectors usually store data in time-based batches. For example, collectors with flat-file storage, such as *nfcapd*, stores data in five minutes batches[25]. Once a batch is complete, it enters an analysis. The results of the queries are not available continuously. Instead, they are available at a certain intervals given by batch size.

In the stream-based workflow, the data is analyzed in so-called *data streams*. An IP flow record is processed immediately after their creation. Such an approach reduces the time of analysis as we do not need to wait for a data batch completion. The stream-based workflow is one of our contribution to IP flow monitoring, and it is discussed in detail in Chapter 6.

3.4.2 Reporting and Analysis

Reporting and analysis belong to the original purpose of IP flow monitoring. Possibility to collect information about network centrally are highly valued advantages of IP flow monitoring. Reports created from IP flow records can provide information on the network as a whole. For example, common report type is bandwidth reporting. Such a report can contain a sum of flows and data volume transferred by/to a customer, number of connected devices. A list of sample statistics and metrics to derive from IP flow records is provided in [143]. Reports and analyses use following query types and their combinations to search through the IP flow records:

- **Filter** is used to select traffic of interest. A commonly applied filter is a time filter that selects relevant time interval of a data[26]. Other, frequently employed filters are IP address or range filters that enable us to focus on a specific network subnet, L3 protocol filters that turn attention to, e.g., only TCP traffic, or port filters that address an application using a specific port. The filters can be combined using standard logical operators AND, NOT, and OR.

- **Aggregation** serves to get information grouped by a given key. The IP flow records are usually grouped by IP address or network range to obtain all data about a host, network range, or communicating pair. IP flow records can be, in general, aggregated by any flow key. The measured IP flow properties are either summarized (numbers of flows, packets, bytes) or averaged (rates flows/s, packet/s, bytes/s) in the grouping.

- **Sort** operation is usually applied to the measured IP flow properties. For example, a descending sort is applied to the number of packet values to obtain an IP flow with the highest number of packets transferred. Besides the measured properties, the sorting algorithm is used for time sorting of the IP flows. The IP flows are stored in a collector depending on the export time. They are not ordered by the start time which is a natural sorting used for traditional IP flow record analyses.

- **Top N statistics** represents a combination of above three query types. "Find 3 IP addresses that transferred the most bytes during last 5 minutes" is a typical query for this statistics. The statistics first filters the given period of data (e.g., last five minutes), then aggregates the filtered IP flow to get summary statistics (e.g., by source IP address), and sorts them according to a given metric and statistics (sort descending by the number of bytes). First, *N* records of the sorted lists are returned as the Top N query result. This type of query is frequently used for identification of *top-talkers*[27].

These queries are used for browsing and filtering data, providing statistic overview, and creating content for reports. Besides the reporting and statistics computation, these queries are also

25. The size of a batch is usually aligned with the value of the active timeout.
26. This fact does apply in batch-based processing. Stream processing notion of time filter is slightly different. See Chapter 6.
27. Top-talkers are hosts that create the majority network traffic transferred via the source observation points.

used for alerting. Alerting can be used for identification of user exceeding his/her allowed bandwidth, an outage of network link detection, malfunction of an application alerting, or misconfiguration of a service identification, for example. Li et al. discriminates following categories of network monitoring and analysis [142]: network monitoring, application monitoring, host monitoring. Network monitoring covers analysis information about routers, application monitoring covers analysis of application usage and is used for planning, and host monitoring provides knowledge about a hosts utilization and behavior that can be used for, e.g., detection of security policy violation.

3.4.3 Performance Monitoring

The goal of performance monitoring is to watch the status of the services in a network. The IP flow records contain information that can be used to assess performance of the services. Frequently used information for performance assessment are, for example, Round-Trip-Time (RTT), delay, response time, or packet loss. The performance of services can be measured by a client-based service monitoring. The client-based measurement offers more possibilities than IP flow-based performance monitoring in terms of available metrics and precision on the one hand. On the other hand, the client-based monitoring requires an installation of agents in the monitored devices, which requires direct access to all monitored devices, impedes the setup of the measurement infrastructure, and results in a need for sophisticated management of agents infrastructure. The IP flow-based performance monitoring enables central, one point, performance monitoring without the necessity to access the devices directly. The cost for the remoteness is the reduced number of information available for performance assessment.

Commonly available IP flow records enable us to monitor primary performance metrics such as service availability and RTT. The availability of the service can be determined based on the TCP flags. The RTT can be derived from the bidirectional IP flows by computing a difference between the starting times of request and response IP flows. The performance metrics computed directly from common IP flows are instantly available and easy to deploy. However, the reported metrics can suffer from flaws resulting from IP flow implementation. For example, measured RTT does not represent the performance of service solely. The measured performance includes the performance of a hosting device and network links from a host to an observation point. An increased RTT then can represent flaws in network links or swapping host instead of the decreased performance of service itself.

The common IP flow using traditional flow keys provides a limited view into monitored service only. For example, we can monitor only the web server as a whole, but we have no information about the performance of individual web pages on the web server. Next-generation IP flows can provide enhanced visibility to application performance. IP flows containing information from HTTP headers provide statistics also for individual web pages. There emerge IP flow probes optimized for performance monitoring, e.g., *nProbe*[28], or *Flowmon probe*[29]. These probes leverage DPI techniques to retrieve additional information from network traffic to determine a service's performance.

3.4.4 Attack and Anomaly Detection

IP flow records can be naturally used for network forensics, i.e., for investigating with which host is a target host communicating. The observation of communication pairs in combination with IP reputation lists or blacklists brings a possibility to detect infected devices in a network. IP reputation lists and blacklists contain a list of sources of malicious activities, e.g., command and control servers of botnets, machines with malware, spam machines, or servers hosting phishing

28. `https://www.ntop.org/products/netflow/nprobe/`
29. `https://www.flowmon.com/en/products/flowmon/probe`

websites. A communication with an IP address that is listed in a blacklist means that the communicating host is likely to be infected. There exist approaches that enhance the blacklists quality. For example, Moura, Sadre, and Pras provide a proof of existing internet bad neighborhoods in [144], which can be used to widen the blacklist coverage. Similarly to communication with blacklisted IP addresses, communication with honeypots can be monitored to detect adversaries sweeping protected network.

The difference between anomaly and attack detection lies in the information that is available at the beginning of an analysis. Among IP flow community, an attack detection is understood as a search for a known attack pattern in a data, i.e., an analyst knows what to look for precisely. The anomaly is *"something"* that deviates from what is standard, normal, or expected. In the case of anomaly detection, it is not precisely defined, what to search for in data. Moreover, an anomaly is not necessarily an attack. An anomaly can be caused by, for example, an unexpected, benign behavior of a user. In general, flow-based attack detection can be exact and can provide a high detection rate. The anomaly detection methods serve mainly for identification of suspicious events.

Information carried in IP flows records can be used to detect specific network attacks. Since IP flow record contains reduced information from network traffic (only information form packet headers are present), IP flow-based detection can detect only a subset of attacks. Sperotto et al. provided an analysis of attacks which can be detected using IP flow records in [125]. The attack classes, that can be detected using IP flows, are *(Distributed) Denial of Service* ((D)DoS), *scans*, *worms*, and *botnets*. Vykopal, Drasar, and Winter have investigated the possibilities of flow-based detection of brute-force attacks in [145]. The techniques and challenges for flow-based intrusion detection are surveyed in [140].

A fundamental, comprehensive and detailed overview of network anomaly detection methods is provided by Bhuyan, Bhattacharyya, and Kalita in [146]. Figure 16 depicts main categories of anomaly detection methods. Statistical methods are suitable for alert generation and can be used without previous knowledge of the normal behavior of a network. However, they can be poisoned by an attacker, i.e., an attacker can slowly train the statistical method to mark the attacker's traffic as normal. Classification methods serve for classifying unknown network traffic into given classes. The classified classes can be binary (normal vs. anomalous traffic), or the classes can represent different types of traffic (e.g., Voice over IP (VOIP), machine to machine, gaming, video, tunneled traffic). Clustering techniques aim to divide analyzed traffic into clusters that contain as much similar traffic as possible. Clustering techniques are used, when no categories are known, or for categories identification. Soft computing methods use techniques such as neural networks, genetic algorithms, and fuzzy sets to identify an anomaly in network traffic. The disadvantages of soft computing methods are a need for annotated traffic sample and black-box-like algorithms. Knowledge-based methods use expert knowledge and rule description to define and detect anomalies. The survey also includes a list of available datasets for network anomaly detection including the dataset evaluation.

Further, Patcha and Park provide an overview of anomaly detection methods in [147]. The overview includes a description of different types of intrusion detection systems, anomaly detection techniques classification, and open challenges. Another survey on anomaly detection is provided by Chandola, Banerjee, and Kumar [148]. The authors focus primarily on the underlying approaches and assumptions of each of the described anomaly detection techniques. Machine learning approaches for network traffic classification are surveyed by Nguyen and Armitage in [149]. The survey contains a table of reviewed literature including the description of machine learning algorithms, its features, data traces used, and classification level. Data mining and machine learning methods used for cyber intrusion detection are listed in a survey by Buczak and Guven [150]. Apart from the explanation of machine learning and data mining

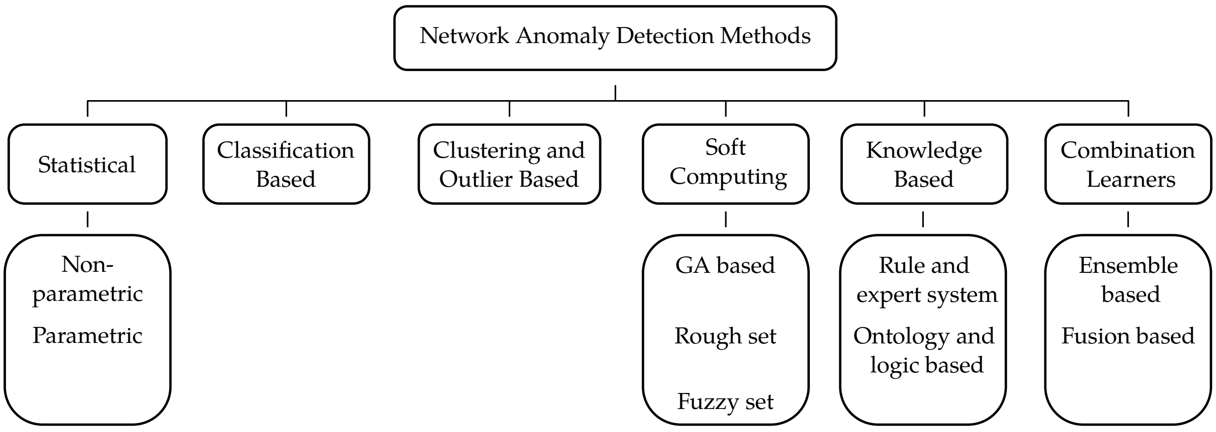


Figure 16: Anomaly detection methods classification (*adapted from [146]).*

methods, this survey also discusses the major steps of the data analysis process. Issues and challenges of network traffic classification are summarized in [151].

3.4.5 Timeline

Analogous to Section 3.3.5, we describe the timeline of IP flow analysis and aim to investigate the delays introduced into analysis workflow by its design. We define significant points of the IP flow record analysis process, which enables us to define delays between these points. We want to keep the description of the timeline as general as possible, so we do not take into account nor the type nor the purpose of the analysis. In general, the IP flow analysis comprises steps depicted in Figure 17.

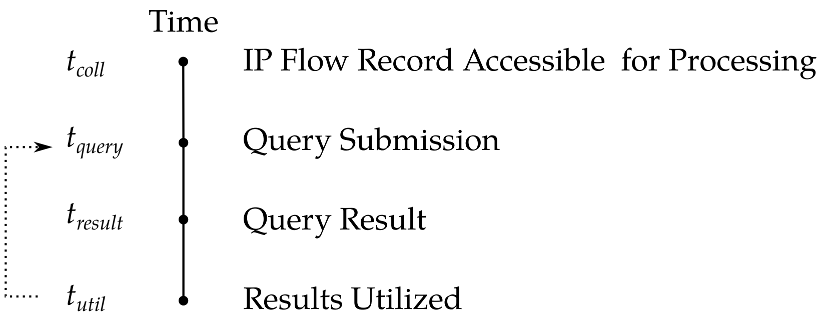


Figure 17: Timeline of IP flow analysis.

The analysis workflow begins when IP flow records are available for processing at a collector. Next step in the IP flow record analysis workflow is the formulation of a query and its submission to the collector. The collector then processes the data and returns the query results. The result is received, analyzed and utilized by a query submitter, either a human operator or an automated process. The result can raise additional questions, new queries are formulated, and the analysis process starts from beginning again.

As mentioned above, the timestamp $t_{query} = t_{coll} + \Delta_{query}$ represents a point in time, when a query is submitted to a collector. We differentiate two types of queries in this context. The first type of query is a *new query*. This type is a previously unknown query, and it needs to be formulated and submitted by a human operator. The operator needs to define IP flow records of interest, describe the analysis tasks and format of the result. These facts need to be then translated into the query language of the collector and submitted for processing. The time needed for these tasks, denoted as Δ_{query}, is in orders of tens of seconds at the minimum as it is physically impossible for a human operator to submit a query faster. The other type of query is an *already known query*. This type represents a query that has already been defined and now is only resubmitted. The submission of this query can be raised by a human operator or, more frequently, by an automated process. In this case, the time needed to submit the query Δ_{query} is in orders of milliseconds as no query formulation is needed.

The timestamp $t_{result} = t_{query} + \Delta_{result}$ represents a point in time, when a collector delivers a result of a submitted query. The time needed a query execution, denoted as Δ_{result} mainly depends on following parameters: volume of processed data, the complexity of the query (including the complexity of the analysis algorithm), the hardware configuration of the collector, and type of the database used in a collector. The larger volume of data to process, higher query complexity (e.g., data mining algorithms) increases the query execution time. Regarding the mentioned query types for IP flow reporting and analysis, the most complex query having the highest Δ_{result} is the Top N statistics. The evaluation of different query types and their response times are described by Hofstede et al. [131] and Velan [132]. According to the author's evaluation, a query over a one day network traffic ranges from twelve to twenty minutes.

The timestamp $t_{util} = t_{result} + \Delta_{util}$ represents a point in time, when a results from a query is utilized for further usage. This step is not usually mentioned in the context of IP flow record analysis. We mention this point specifically as it fits the context of the cyber situation awareness, where the purpose of an analysis is to be used for forming a decision. There emerges a delay, denoted Δ_{util}, between the point a query result is available and the point a result is utilized for further action. In case the whole analysis and decision process is automated, the delay is negligible. However, when the results of the analyses are processed by a human operator, the delay increases. Results of automated queries that need to be processed manually may wait for their utilization in orders of minutes, hours or even days. For example, results of queries being part of a monthly reports generation can wait for their utilization even one month.

3.4.6 Open Issues

The open issues of IP flow record analysis are linked to current trends in IP flow approaches (see Section 3.2.3). We identify following groups of open issues from literature. Issues related to analysis evaluation, complexity issues, and privacy issues. Besides the open issues mentioned in literature, we identify issues related to a host-based view of IP flow records.

- **Analysis Evaluation** – Evaluation of analysis results is a major and prevailing issue of IP flow record analysis [140, 146, 152–155]. There is a lack of updated public IP flow-based dataset that could serve for rigorous analysis evaluation. The currently popular dataset are outdated, such as KDD from 1999, or DARPA from 2000, or does not contain IP flow records, such as Caida datasets[30]. Several works are discussing the issues related to the evaluation methodology. Gharib et al. [153] present a formula to estimate the quality of datasets for evaluation purposes. Tavallaee et al. [155] survey the evaluation of anomaly detection systems performed by researchers. In their survey, most of the recent works are found lacking reliability and validity of results. Further, researchers use different IP flow records, select various IP flow keys, and usually do not provide an exact specification of the IP flow monitoring process due to page limit for papers. These facts further hinder re-evaluation of the presented analyses.

- **Complexity** – The design of an IP flow record analysis method needs to take into an account a large volume of data to be processed and associated computational complexity. Modern approaches to data analysis, such as machine learning and data mining, are not usually designed for large-scale online analysis of IP flow records. In default settings, the

30. `https://www.caida.org/data/`

algorithms are computationally complex and require extensive resources. According to Buczak and Guven [150], algorithms for online analysis should be of complexity $\mathcal{O}(n)$ and $\mathcal{O}(n \log n)$. $\mathcal{O}(n^2)$ are acceptable for most of the practices, and $\mathcal{O}(n^3)$ and higher should be used mainly for offline analyses. Moreover, the mechanism of the analysis itself should not be too complex. An operator should be able to understand the analysis mechanism to be able to interpret the analysis result correctly. The black-box analyses, such as neural networks, does not enable the operator to interpret the results of the analysis properly. The reduction of both analysis and computational complexity and related increase of analysis system performance are research problems of current interest.

- **Privacy** – User privacy should be preserved even during an analysis. There exists a trade-off between a degree of anonymization and available information in a data for analysis. Various anonymization techniques that would keep information in data and enable for advanced IP flow analysis are explored. The challenges of network data anonymization are summarized in [139]. Impact of anonymization on network security analysis is assessed in a master thesis by Bose [156].
- **Host-based view** – An IP flow record represents a connection. It is stored in a collector in such a manner. A transformation of data from a connection-based paradigm to a host-based paradigm is time-consuming. The transformation includes finding all relevant IP flow records for a host in time, aggregate them for each time frame, and compute wanted characteristics. Such operations need to be done for all hosts in a monitored network. This approach is computationally demanding, and an operator does not have instant access to a hosts information at hand. A possible solution, pre-computation of characteristics for all hosts in advance, is a computationally demanding task, though.

3.5 IP Flow Monitoring and Cyber Situation Awareness

Having described the cyber situation awareness and IP flow monitoring, we show an application of IP flow monitoring in cyber situation awareness context. In this section, we use Ensley's three-level model of SA and apply it to IP flow monitoring workflow. Further, we discuss the specifics of CSA described in Section 2.2.2 in the light of IP flow monitoring. We also revise the challenges of CSA with respect to IP flow monitoring. This section aims to demonstrate a role of IP flow monitoring in CSA and possibilities and limitations of the usage of IP flow monitoring for CSA.

To recall, Endsley's Three-level model describes the SA as *"the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in the near future"*. Since IP flow monitoring is part of network traffic monitoring research field (see Figure 10), the term *environment* in Endsley's definition does not apply to the whole cyberspace, as in the general CSA, but is limited to a computer network instead. The limitation to the computer network is not strict, in any case. The computer network is used for communication between computers. Information from observed IP flow records can be used to derive partial information about a communicating computer (e.g., application performance monitoring). Still, the derived information is limited and incomparable with information provided by an agent deployed directly on the computer. For example, exact information on running processes, CPU utilization, used drive space are not directly observable from a computer network. Hence, IP flow monitoring can provide a CSA over a computer network, but not over the whole cyberspace.

Further, it is essential to understand, that IP flows are not designed for providing information *about* network itself. The primary purpose of IP flows is to represent information *from* network, i.e., network traffic. IP flow records can be used for network topology discovery, but the topology is usually not complete as only network traffic passing observation points is collected and some computers do not communicate over the observation points (or does not communicate at all). The information from a network is the primary information perceived during IP flow monitoring.

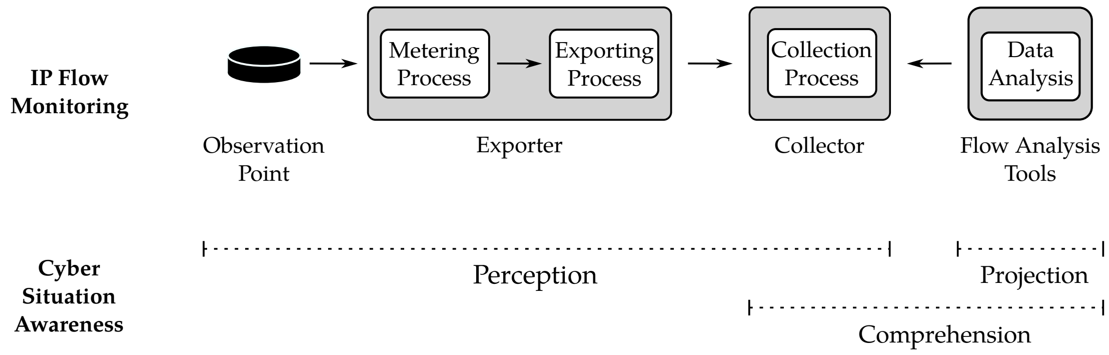


Figure 18: IP flow monitoring in the context of cyber situation awareness.

The three levels mentioned in Endsley's model are the Perception level, the Comprehension level, and the Projection level. The Perception level covers data observation and collection, the Comprehension level aims at understanding the collected data, and the Projection level serves for prediction of a future state of the environment based on collected data and their comprehension (see Figure 6). We show, how an IP flow monitoring workflow fits the three levels of the CSA model in Figure 18.

The Perception level covers Metering, Exporting and Collection processes. Assigning these processes into a Perception level is straightforward: during these processes, information from a network in the form of IP flows records are observed, created and stored, which represents a perception of information from a network. The Comprehension level comprises of collecting process and data analysis phase of the IP flow monitoring workflow. The data analysis phase serves for gaining Comprehension of the data. Various analyses, visualizations, and dashboards are used to ease an understanding of network traffic. The collecting process also overlaps into the Comprehension level, as during collection process basic statistics might be computed. Moreover, the collection process coalesces with the analysis in stream-based architectures for IP flow processing as we will show later in this thesis. An alternative approach to CSA can include the whole IP flow monitoring workflow into the Comprehension level as an operator should be aware of settings of the IP flow monitoring process to be able to comprehend the data carried in IP flow correctly. Nevertheless, we stand for Comprehension level that comprises from collecting and analysis processes as we believe that during the Comprehension level we aim to comprehend information from data collected during the Perception level, and that the understanding of Perception level should not be part of Comprehension level. The Projection level is represented solely by data analysis phase. As mentioned before, the main purpose of IP flows is to provide visibility into a network. The prediction of future network state is not the primary goal of IP flow monitoring. Nevertheless, the information obtained via IP flow monitoring can be used to predict future network state. The prediction quality depends on the algorithms IP flow records analysis and prediction.

3.5.1 Specifics of IP Flow Cyber Situation Awareness

There are specifics of application of CSA to IP flow monitoring domain that need to be discussed. We similarly divide the specifics as in Section 2.2.2 into the cyber environment, perception, and performance specifics. The attacker-related specifics are not discussed here as these specifics applies fully to IP flow monitoring domain. As discussed before, the cyber environment related to IP flow monitoring is a computer network. The computer network is a highly dynamic and complex environment. The computer networks are usually hierarchical structures with autonomous layers, so it is hard to keep information on the network centrally. The lack of information on a network often hinders the achievement of CSA for an operator. The boundaries of CSA can be stated more precisely in the IP flow CSA than in general CSA. The boundaries are usually set to the network range of the monitored network. The current trends in IP flow, mainly the shift to cloud services, change the boundaries of the IP flow CSA, though. Regarding the cloud service, the boundaries of IP flow CSA vanish. An operator needs to take into account, that there is no control and no access to underlying physical infrastructure, and actions linked to CSA are limited. The information considered for IP flow CSA are reduced to information that can be obtained from IP flow records.

The perception specifics of CSA resulting from application to IP flow monitoring domain are visibility limited to observation points and probe-based network perception. Similarly to a general CSA, information from a network can be observed only via sensors. The traffic access methods are described in Section 3.1. The information that can be used for IP flow CSA is only that observed at observation points. The location of observation points then plays an essential role in achieving CSA. For example, observation points located only on network perimeter does not provide any information on intranet traffic. We can observe only ingress and egress network traffic. To achieve a complex CSA, the observation points should be able to cover all, or at least the majority of network traffic. In an ideal case, observation points should be located at all routing devices in a network. This state is achievable as many of these devices support the IP flow export by default. Besides the location of the observation points, the parameters of the IP flow capture plays a vital role in CSA. An operator should be aware of the optional settings of the IP flow monitoring process as these options influence a perceived data (e.g., packet/IP flow sampling, active and inactive timeout, the transport protocol for IP flow records).

The performance specifics of CSA related to IP flow monitoring domain are the reduction of event speed and the large volume of data in computer networks. An IP flow record represents an abstraction of network connection. The number of events to process is reduced from a packet rate to a connection rate. Therefore, the speed of events that need to be processed by an operator is reduced compared to packet observation techniques. Further, the boundaries of IP flow CSA are limited to a chosen network range, which further reduces information to process in IP flow CSA compared to a general CSA. Nevertheless, the number of observed events (IP flow records) is still too high to be processed by a human. Hence, automated analyses are employed, and an operator receives only predefined alerts and statistics. The resources needed to launch an attack remains relatively low in IP flow CSA. The resources for defending a network decreases when considering IP flow CSA. The IP flow metering can be run using commodity hardware. However, IP flow record analysis methods require significantly higher computation resources to process the large volume of IP flow records. Nevertheless, the resources needed for obtaining IP flow CSA are lower compared to a general CSA.

The set of entities relevant to IP flow CSA is naturally smaller than a set for a general CSA. Since the IP flow monitoring focuses on observing information from a network and not on the information about a network, the physical entities are covered by IP flow CSA only partially. As mentioned before, some information about physical infrastructure can be derived from the IP flow records. However, the volume of available information is significantly lower than in other approaches. The immaterial entities can be observed from IP flow records as well. It is possible to determine services running in a network to some extent and even determine their performance via application performance monitoring approaches. The human entities considered in IP flow CSA are twofold: operators administrating and defending a network, and humans interacting with computers in a network. The operators use information from IP flow record to achieve the CSA to administrate the network and protect it from adversaries. The human's interactions with computers create the network traffic. The interactions are captured in the IP flow records.

It is important to note, that it is a challenging task to identify and follow a human identity in an IP flow records as only IP addresses are captured. In majority cases, it is not possible to link a human to individual events in a network. Nevertheless, there are indirect ways to link a person to computer even using IP flows (e.g., behavior profile, frequently visited pages, time of communication).

3.5.2 Challenges

This section discusses the challenges of CSA presented in Section 2.2.5 with respect to IP flow monitoring. For the identified CSA challenges, we present a contribution of IP flow monitoring to these challenges. We identify challenges that are still open and highlight challenges that are addressed in this thesis.

Cyberspace-related

The CSA cyberspace in the context of IP flow monitoring is limited to a computer network, data transferred via the network, and human entities interacting with the network. We present the contribution of IP flow monitoring to the cyber-space related challenges below.

- **Complexity** – The understanding and creating a mental model of a cyber environment is a challenging task for a human. IP flow monitoring enables a central collection of IP flow records, which can be used for a comprehension both of processes in a network and even network topology to some extent. Using IP flow records, it is possible to identify main network traffic flows and traffic distribution among individual hosts in a coherent manner. This information can be passed to an operator and assist him/she in creating a mental model of a computer network. In this thesis, we investigate the possibilities of identification of a network host in network traffic since the host identification is one of the main requirements for understanding the complexity of a computer network.

- **Dynamics** – The dynamics of a network is hard to track. The passive approach of the IP flow monitoring enables an operator to continuously monitor and keep track of the changes in the network (both topology and content of network traffic) straightforward. The passive measurement observes events in a network and changes are projected directly to observed network traffic. No scheduled updates for a collection of up-to-date information is needed as in the case of active monitoring. Nevertheless, the payoff for the continuous observation of the passive approach is less detailed information that is possible to derive from the data compared to active approaches. Optimization or reduction of the trade-off between the volume of the information available and the possibility of continuous observation remains the challenge, though.

- **Speed of events** – The speed of events in cyberspace is a challenge for a cyber operator. The IP flow monitoring enables to process information from a computer network at high speeds. However, the IP flow monitoring introduces several delays during the metering and analysis processes as shown in the previous sections. Due to these delays, a cybersecurity operator does not operate with up-to-date data, which may lead to incorrect decisions. In this thesis, we investigate a possibility to reduce the delays in IP flow monitoring workflow by introducing stream-based IP flow monitoring (see Chapter 6).

- **Rapid evolution** – The IP flow monitoring does not influence an operator's ability to keep up with new trends directly. IP flow traffic does not provide any report on new trends, properties, and functions. However, as soon as these new technological advances are implemented and deployed, they can be observed in network traffic. So that, an operator can indirectly learn the latest trends and their real-world demonstrations from network traffic observation. New trends that are possible to observe in network traffic are, for example, deployment of a new application, behavior habits of the computer users, new services and protocols present in a network.

Data-related

The IP flow monitoring is limited only to observation of network traffic; no host-originating information, such as logs, CPU usage, disk usage, power consumption, are available. Even though the data used in IP flow CSA are reduced only to network traffic, it still shows all characteristics of *big data*. For example, in a medium-sized network of 24,000 active IP addresses, it is possible to observe 12 000 flows/s. An education network produces flows at a rate 110 000 flows/s [A3]. These values are expected to grow in the future [81].

- **Volume** – The *"data overload, meaning underload"* holds in case of IP flows, as well. The volume of raw IP flow records generated from network traffic is too big to be processed by a human operator. Although an IP flow record contains an aggregated information from a set of packets, it still contains relatively low-level information. An operator needs higher level information from IP flows to be able to use the IP flow records efficiently. There exist approaches to provide an aggregated information from IP flow records (see description in Section 3.4). Nevertheless, the selection of relevant aggregation approaches, appropriate scope, and suitable information presentation (e.g., lack of host-based view in IP flow monitoring) are still open challenges for IP flow CSA. In this thesis, we focus on usability IP flow aggregations and creation of the aggregated host-based view.
- **Velocity** – The velocity of the IP flows in network traffic is high; the rates of IP flows reach over 110 000 flows/s. The vendors of the IP flow monitoring hardware can keep up with the rising speed of network traffic. They deliver solutions even for backbone lines. Currently, FPGA network cards capable of monitoring and creating IP flow from lines with rates 100 Gbps are available [116], and the 200 Gbps rate is expected to break in a near future. While we can create IP flows from high-speed traffic, the velocity remains a challenge in the area of IP flow record analysis. The current tools do not provide sufficient throughput for IP flow analysis, especially when advanced analytic methods, such as Neural Networks, are employed. This thesis focuses on the throughput evaluation of IP flow monitoring workflow when next-generation IP flows are monitored. Further, we describe a design of scalable, high-throughput IP flow monitoring workflow that can serve for IP flow CSA.
- **Variety** – Considering the IP flow monitoring for CSA solely, the variety of data is reduced to IP flows and IP flow records. The structure of IP flow records is given by the IP flow exporter and by the protocol used for IP flow record transport. Nevertheless, even IP flow records can contain a variety of information. The IP flow keys can be chosen at will, and additional private information elements can be added to IPFIX protocol to be able to transport a custom data of interest. Standardized IPFIX information elements and their numbers are maintained by Internet Assigned Numbers Authority (IANA). The information elements in the list are the basic elements, and other private elements are frequently added by various vendors, which increases the variety of information transferred in IP flow records. Nevertheless, the variety of data in IP flow records is negligible compared to the variety of data in the general CSA.
- **Value** – The value of information carried in the IP flow can be determined by a value of that information to an operator. The exact determination of the value remains a challenge

also in the IP flow CSA. The noise to signal ratio is lower compared to packet monitoring due to aggregation introduced. Nevertheless, it remains high, and several techniques for anomaly detection and *"normal traffic"* identification are under research [146]. In this thesis, we show information available from a general Top N statistics, and we investigate possible applications of this characteristic that might an operator benefit from (Chapter 5).

Toolset-related

The toolset for IP flow monitoring can be divided into two main categories - IP flow metering tools and tools for IP flow records analysis. The IP flow metering tools range from IP flow export software integrated in network routing and switching devices (e.g., Cisco Catalyst 3750X-12S-E Switch[31]) to a dedicated hardware for IP flow metering (e.g. Flowmon probe[32]). The landscape of tools for IP flow record analysis is also diverse. IP flow record analysis is part of various systems and tools: Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), or Anomaly Detection Systems (ADS), network monitoring tools, and so forth. Individual CSA challenges concerning IP flow monitoring toolset are presented below.

- **Performance** – The performance of the tools for IP flow CSA is an issue mainly for IP flow analysis tools, as discussed above. The data-related challenges imply the high requirements on the performance of the tools for IP flow record processing in terms of processing power, throughput, and so forth. Besides the performance issues of the IP flow analysis tools, the reduction of the time delay introduced in the IP flow monitoring workflow is a frequently mentioned challenge related to IP flow monitoring. In this thesis, we discuss the possibilities of the reduction of the time delay and study the performance of IP flow metering tools concerning the volume of information collected (Chapter 4 and 6).
- **Heterogeneity** – The heterogeneity challenges remains in the context of IP flow monitoring. There exist various types of IP flow probes, collectors, and analysis tools. The collectors differ in the query languages based on the storage approach; for example, *nfdump* query language differs from structured query languages (SQL) in relational databases. The IP flow probes are represented by different switching and routing devices from different vendors exporting IP flows in various formats. The workflow of an IP flow analysis is also vendor specific. An operator is usually forced to use different tools and approaches to gain a CSA. To achieve homogeneity in approaches, we present a general workflow for next-generation IP flow monitoring in Chapter 6.
- **Visualization** – Visualization challenges of CSA are linked to issues of big data visualization. Open issues for IP flow CSA are the visualization of a large network, relations between hosts in a network derived from IP flow records at a different timescale and level of details.

The overlap of the open issues of cyber situation awareness and IP flow monitoring is displayed in Table 2. The performance issue of the IP flow monitoring naturally implies the performance issue of CSA toolset as IP flow monitoring is used to achieve CSA. The complexity issue of CSA relates to the new trends in IP flow monitoring, namely encrypted traffic, and cloud services, as these trends add new layers to the network and reduce information available for analysis. The complexity of the IP flow analysis and efforts for its reduction also decrease the complexity of the CSA as the easier understanding of the IP flow analysis results in better understanding of the cyber space. The problems related to the speed of events and dynamic of

31. https://www.cisco.com/c/en/us/support/switches/catalyst-3750x-12s-e-switch/model.html
32. https://www.flowmon.com/en/products/flowmon/probe

the cyberspace can be partially addressed by the solving problems related to the response time reduction in the IP flow monitoring.

			IP Flow Traffic Monitoring							
			IP Flow Monitoring				IP Flow Analysis			
			Performance	Scalability	Response Time	Changing Paradigm	Analysis Evaluation	Complexity	Privacy	Host-based View
cyber situation awareness	Cyber	Complexity				✓	✓	✓		✓
		Dynamics			✓					
		Speed of Events			✓					
		Rapid Evolution				✓				
	Data	Volume	✓	✓				✓		
		Velocity	✓	✓	✓					
		Variety				✓		✓		
		Value							✓	✓
	Toolset	Performance	✓	✓	✓			✓		
		Heterogeneity				✓				
		Visualization		✓		✓				✓

Table 2: Overlap of IP flow and CSA challenges.

3.6 Summary

This chapter provides an introduction to the field of network traffic monitoring. The goal of this chapter is to present to a reader the basic concepts of IP flow monitoring, put the IP flow network monitoring into a context of cyber situation awareness, and discuss current open issues and challenges, that are addressed in this thesis.

First, we explain the position of IP flow monitoring in the broader research fields, compare it briefly with other approaches to network traffic monitoring, and discuss the traffic access methods used for IP flow measurements. In the next section, we describe the concept of IP flows. We focus on the history and evolution of the IP flow concept as knowledge of the origins of IP flows enables to understand the purpose and design of IP flow monitoring workflow. We provide the IP flow definitions, and we present current trends that influence the IP flows and need to be taken into account. The trends worth mention are concerns regarding privacy issues, transformation to a cloud environment, and the increasing volume of network traffic.

The third section explains the IP flow monitoring workflow. We discuss each phase of the monitoring workflow including relevant literature. The timeline of the workflow is defined to show a delay introduced during the monitoring workflow. We discuss the timeline later in this thesis and present a way how to reduce the described delays. Open issues of the IP flow monitoring, such as scalability, response times, and performance, concludes this section.

The next section describes the IP flow record analysis. We mention the general workflow of the analysis and describe the main use-cases of the analyses - reporting and analysis, performance monitoring, and attack and anomaly detection. For each use-case, we mention relevant literature sources. Similarly to the previous section, the timeline of the analysis workflow is described.

The fifth section puts the IP flow monitoring into the context of cyber situation awareness. We utilize Endsley's Three-level model to describe the IP flow workflow, discuss the specifics of IP flow CSA resulting from IP flow monitoring and analysis workflows and revise the challenges of CSA from Section 2.2.5 to reflect IP flow approach.

The main contributions of this chapter are:

- description of IP flow concept and its history,
- overview of the IP flow monitoring and analysis workflow,
- list of open issues in the IP flow monitoring and analysis domain,
- analysis of timeline of the IP flow monitoring and analysis workflow,
- description of IP flow monitoring in the context of CSA.

4 Data Perception

Data perception level links to IP flow monitoring processes in the context of Endsley's Three-level model applied to IP flow monitoring domain. In this chapter, we present our contributions to the data perception level. We contribute to the analysis of next-generation IP flows export and to improvement of data collection for client identification in encrypted network traffic. Our contributions address the performance open issue of the IP flow monitoring process. Specifically, we focus on the performance of IP flow probes capable of next-generation IP flow export. Addressing performance open issue, we directly respond to the toolset-related performance challenge of CSA and indirectly respond to the volume challenge of data-related CSA challenge category. Apart from the performance of the IP flow probes, we investigate a novel approach to the collection of IP flows that can serve for a client identification even in encrypted network traffic. This research effort responds to the current trend of increasing volume of encrypted network traffic and changing paradigm trend in IP flow network monitoring. We believe that the presented results contribute to the solution of complexity and dynamic challenges of CSA.

The first section investigates how the addition of application-level information into IP flow influences the performance of IP flow probes. We focus on HTTP protocol being the widely used protocol by a variety of applications. We investigate the parsing mechanisms of the state-of-the-art IP flow probes and propose an improved version of HTTP protocol parser. We provide a performance comparison of the other approaches with our approach and with application-less IP flow probes to demonstrate twofold results: over 50 % decreased performance when application-level information is parsed, and the 300 % better throughput of the optimized strcmp parsing algorithm compared to the others. Further, we show that the addition of an extra HTTP header information for parsing, once we are already parsing the HTTP protocol, does not significantly decrease the performance of the IP flow probe.

Next, we focus on the improvement of data collection for client identification from encrypted network traffic. We investigate the characteristics of SSL/TLS traffic to identify information that can be used for client identification. Based on the analysis, we propose a dictionary that serves for identification of client's browser or operating system even from encrypted network traffic. The dictionary is created by combining information from HTTP (User-Agent values) and HTTPS (Cipher Suites) IP flows. We execute series of experiments to examine the share and properties of SSL/TLS network traffic, create the dictionary from real-world traffic, evaluate the coverage of the dictionary, and compare IP flow- and host-based approaches. Based on the experiments, we highlight the dictionary's properties and propose several improvements that would lead to increased accuracy and quality of the dictionary.

This chapter is organized as a collection of following revised peer-reviewed publications that summarize our research efforts in the area of data perception: [A4–A6].

This chapter is structured as follows:

- *Section 4.1 investigates a performance of HTTP protocol parsers and demonstrate the trade-off between a volume of perceived information and performance of the IP flow probes.*
- *Section 4.2 proposes a novel approach to client identification in encrypted traffic by creation a dictionary that links information from HTTP and HTTPS IP flows.*

4.1 HTTP Protocol Parsers for IP Flow Information Export

As explained in previous chapters, IP flow records are based on IP headers (network and transport layer), and it did not originally include any payload information. On the other hand, we observe that HTTP protocol [157] became a *"new Transmission Control Protocol"* (TCP). More and more applications rely on HTTP protocol, e.g. Web 2.0 content, audio and video streaming, instant messaging etc. HTTP traffic (TCP port 80) can usually pass through most firewalls and therefore presents a standard way of transporting/tunneling data. The versatility, ubiquity and amount of HTTP traffic makes it easy for an attacker to hide malicious activities. Missing application layer visibility renders standard NetFlow and IPFIX to be ineffective for HTTP monitoring and limits its utilization for cyber situation awareness.

Network and security devices use application layer analysis to provide application visibility, monitoring and traffic control. For example, *Cisco Application Visibility and Control* (AVC) [158] solution uses next-generation deep packet inspection (NBAR2) and flexible NetFlow to identify, classify and report on over 1,000 applications. HTTP information elements are supported by *YAF* [159] and *nProbe* [130] IP flow meters and are exported in IPFIX format. Most intrusion detection systems extract application layer data for in-depth analysis.

We aim to research the impacts of application layer analysis of HTTP protocol on IP flow monitoring workflow. The contribution of our work is threefold: *(i)* We design and evaluate several HTTP protocol parsers representing current state-of-the-art approaches used in today's IP flow meters. *(ii)* We introduce a new *flex*-based HTTP parser. *(iii)* We report on the throughput decrease (performance implications of application parser), which is of the utmost importance for high-speed deployments.

This research addresses the following challenges of IP flow CSA. We address the data value challenge as we intend to enrich IP flow records with information from the HTTP protocol. The introduction of *flex*-based HTTP parser for high-speed network addresses the data velocity challenge. The performance challenge is investigated by evaluation of throughput performance of the tools when adding HTTP application parser into an IP flow probe.

4.1.1 State-of-the-art

Application layer protocol parsers are an integral part of many network monitoring tools. We explored the source code of the following frameworks to see how the HTTP parsing is implemented. *nProbe* [130] uses standard *glibc* [160] functions like *strncmp* (compare two strings) and *strstr* (locate a substring). *YAF* [159] uses Perl Compatible Regular Expressions (PCRE) [161] to examine HTTP traffic. *Suricata* [162] and *Snort* [163] are both written in C. *Suricata* uses *LibHTP* [164] library which does HTTP parsing using custom string functions while *Snort* does its parsing using *glibc* functions. *httpry* [165] is another HTTP logging and information retrieval tool which is also written in C and uses its own built-in string functions. These HTTP parsers are hand-written.

Another approach is taken by *Bro* [166] authors. They use *binpac* [167], a declarative language and compiler designed to simplify the task of constructing robust and efficient semantic analyzers for complex network protocols. They replaced some of *Bro* existing analyzers (handcrafted in C++) and demonstrated that the generated parsers are as efficient as carefully hand-written ones.

We try to determine whether these approaches to HTTP parsing can handle large traffic volumes common in CSA. Besides the above approaches, we propose to use the *Fast Lexical Analyzer* (*flex*) [168] to design a new HTTP parser. *Flex* converts expressions into a lexical analyzer that is essentially a deterministic finite automaton that recognizes any of the patterns. The algorithm that converts a regular expression directly to deterministic finite automaton is described in [169] and [170].

There are other works that inspect the HTTP protocol headers. Authors of [171] use statistical IP flow analysis to differentiate traditional HTTP traffic and Web 2.0 applications. In [172] the authors identify HTTP sessions based on IP flow information. In both cases, a ground truth sample is needed, which is a topic addressed by [173]. In [174] and [175] the authors use DPI to obtain information from the HTTP headers. Our approach is orthogonal to these works since we are interested in extending IP flow records with HTTP data.

4.1.2 Parser Design

HTTP protocol [157] has a number of properties that can be monitored and exported together with IP flow data. The most commonly monitored ones are present in almost every HTTP request or response header. Based on the properties monitored by the state-of-the-art DPI tools we selected the following ones for our parsers: *HTTP method, status code, host, request URI, content type, user agent and referer*. Keeping track of every bidirectional HTTP connection is too resource consuming on high-speed networks. Thus we focus on the evaluation of each packet. This approach is more common for IP flow meters since it is more resistant to resource depletion attacks.

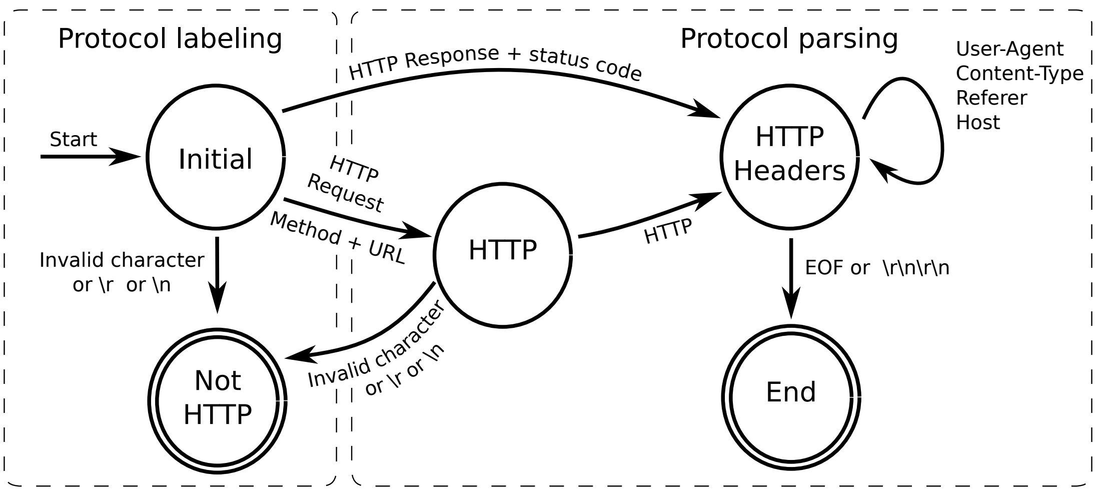


Figure 19: *flex* algorithm schema.

We implemented and evaluated three different types of parsing algorithms. The first algorithm (*strcmp* approach) loops the HTTP header line by line and searches each line for given fields. It uses standard glibc string functions like *memchr*, *memmem* and *strncmp*. The simplified pseudocode is shown in Algorithm 1. The second algorithm (*pcre* approach) uses several regular expressions taken from *YAF* to search the packet for specific patterns indicating HTTP header fields. The pseudocode for the *pcre* algorithm is shown in Algorithm 2. We designed the third algorithm (*flex* approach) to handle each packet as a long string. It uses finite automaton to find required HTTP fields, and the Flex lexer is used to process the packets. The automaton design is shown in Figure 19.

Algorithm 1 *strcmp*

```
1: if first line contains "HTTP" then
2:     while not end of HTTP header do
3:         for every parsed HTTP field do
4:             if field matches the line then
5:                 store the value of the line
6:             end if
7:         end for
8:         move to the next line
9:     end while
10:    return HTTP packet
11: else
12:    return not HTTP packet
13: end if
```

Algorithm 2 *pcre*

```
1: if first line contains "HTTP/x.y" then
2:     for all PCRE rules do
3:         if rule matches then
4:             store the matched value
5:         end if
6:     end for
7:     return HTTP packet
8: else
9:     return not HTTP packet
10: end if
```

Since the Flex is a generic tool, its initialization before scanning each packet is quite an expensive operation. Therefore we decided to remove all unnecessary dynamic memory allocations and costly initializations to see whether the performance can be increased. We named the new version *optimized flex*. The disadvantage of *flex* is that it has to keep the data in its own writable buffer. Therefore the received data must be copied to such a buffer, which adds to the processing costs significantly. The advantage of the *flex* parser is its simple maintenance and extension possibilities. The framework can be modified to parse any other application layer protocol just by changing the set of regular expression rules. The *strcmp* parser would have to be rewritten from scratch.

The *strcmp* implementation also offers a space for further improvement. Algorithm 3 shows an *optimized strcmp* version of the code that features a better processing logic. The optimized version searches for specific strings by comparing several bytes at once, which is done by casting the character pointer to integer pointer. The number that is compared to the string is computed from ASCII codes of the characters and converted to network byte order. The size of the used integer depends on the length of the string; longer integers offer better performance.

To focus only on the HTTP parsing algorithms, we decided to let the FlowMon exporter [176] handle the packet preprocessing. We used a benchmarking (input) plugin that reads packets from PCAP file to memory at start-up. Then it supplies the same data continuously for further processing. This approach allows us to focus on benchmarking the algorithms without the necessity of considering the disk I/O operations. We provide the source code of implemented algorithms and used packet traces at the paper homepage [177].

4.1.3 Evaluation Methodology

We define a methodology for HTTP protocol parsers evaluation in this subsection. We focus on parsing performance (number of processed packets per second) of the algorithms described in Section 4.1.2 from several different perspectives.

The first perspective focuses on the performance comparison with respect to analyzed traffic structure. The second perspective covers the impact of the number of HTTP fields supported by a parser. The third perspective describes the effect of a Carriage Return (CR or `'\r'`) and a Line Feed (LF or `'\n'`) control characters distribution in the packet payload.

A common technique for increasing of network data processing performance is processing only the relevant part of each packet. Therefore, we perform each of the tests in two configurations. In the first configuration, the parsers are given the whole packets. We set the limit of

Algorithm 3 *optimized strcmp*

```
1: if payload begins with "HTTP" then
2:    store status code
3:    while not end of HTTP header do
4:       for every parsed response HTTP field do
5:          if line starts with field name then
6:             store the value of the line
7:          end if
8:       end for
9:       move to the next line
10:   end while
11:   return HTTP response packet
12: end if
13: if payload begins with one of GET, HEAD, POST, PUT,
       DELETE, TRACE, CONNECT then
14:   store request URI
15:   while not end of HTTP header do
16:      for every parsed request HTTP field do
17:         if line starts with field name then
18:            store the value of the line
19:         end if
20:      end for
21:      move to the next line
22:   end while
23:   return HTTP request packet
24: end if
25: return not HTTP packet
```

packet size to 1500 bytes, which is the most common maximum transmission unit value on most Ethernet networks. In the second configuration, the parsers are provided with truncated packets of length 384 bytes, which is the minimum packet length recommended for DPI by authors of the *YAF* exporter [159].

To test the performance of the parsers, we created an HTTP traffic trace (testing dataset). Our requirements on the dataset were as follows: preserve the characteristics of HTTP protocol, reflect various HTTP traffic structures, and have no side effects on the IP flow meter. In order to meet these requirements, we decided to create a synthetic trace.

The HTTP protocol is a request/response protocol. To preserve the characteristics of HTTP protocol during the testing, random request, response, and binary payload packets were captured from the network. To omit the undesirable bias of the measurement only these three packets were used to synthesize test trace. The final test trace consists of 200 packets. In order to reflect various traffic structures, we suggested the following ratio:

$$r = \frac{\text{\#request packets} + \text{\#response packets}}{\text{\#all packets}} * 100 \tag{4.1}$$

where $r \in [0, 100]$ and created a test set for each integer ratio from the interval. Further, we created two packets with modified payload. One packet contained the CR and LF control characters only at the very *beginning* of the packet payload, the other one only at the *end*. For both of the modified packets and for the *unchanged* packet the test trace for each integer ratio was created.

Having defined the test trace, we propose the following case studies to cover all evaluation perspectives. The case studies are carried out for both full and truncated packets. Moreover, we measure the performance of the flow meter without an HTTP parser (*no HTTP* parser). This way we can estimate the performance decline caused by increased application layer visibility.

- **Performance Comparison**: This case study compares the parsing performance of implemented parsers. Moreover, we report on the IP flow meter performance without an HTTP parser (*no HTTP* parser).
- **Parsed HTTP Fields Impact**: This case study shows a parser performance with respect to the number of supported HTTP fields. We incrementally add support for new HTTP fields and observe the impact on the parser performance.
- **Packet Content Effect**: The result of this study presents the influence of the CR and LF control characters position in a packet payload on the parser performance. The test traces containing modified payload packets are used to perform the measurement.

The performance evaluation process employs the benchmarking input plugin (see plugin description in Section 4.1.2) to obtain the number of processed packets per second. In order to avoid influencing the results, the plugin uses a separate thread and CPU core for the accounting. The plugin counts the number of the processed packets in a ten-second interval and then computes the packets per second rate. We have operated the benchmark plugin for fifty seconds for each test trace and computed a number of packets processed and a standard error of the measurement. The parsed HTTP header fields impact and packet content effect were assessed similarly. All measurements were conducted on a server with the following configuration: Intel Xeon E5410 CPU at 2.33 GHz, 12 GB 667 MHz DDR2 RAM and Linux kernel 2.6.32 (64 bit).

4.1.4 Parser Evaluation

This subsection presents the results of the HTTP parser evaluation. First, we describe the parser performance comparison, then we investigate the impact of supported HTTP header fields. Finally, the effect of the packet content on HTTP parsing performance is shown.

Performance Comparison

This case study uses the standard version of each parser that supports seven HTTP fields. The dataset containing the unmodified payload packets is used, and the parsers are tested both on full and truncated packets. Figure 20 shows the result for full packets case study and Figure 21 shows performance evaluation for truncated packets.

First we discuss the Figure 20. The *no HTTP* meter is capable of parsing more than 11 million packets per second. This result is not influenced by the application data carried in the packet since the data is not accessed by the *no HTTP* parser. Employing event the fastest of the HTTP parsing algorithms the performance drops to the nearly one half of parsed packets per second. All of the HTTP parsers show the decrease in the performance as the ratio r increases since the amount of request and response packets, which are more time demanding to parse, grows.

The best performance is achieved by *optimized strcmp* parser, which uses application protocol and code level optimizations. The parser takes into account the HTTP header structure, the difference between HTTP request and response headers and looks only for header fields that can be found in the specific header type. The code level optimizations include converting static strings into integers and matching them against several characters at once, which can be done in one processor instruction. The *strcmp* parser performance is the second best, although the throughput is less than half of the *optimized strcmp* parser.

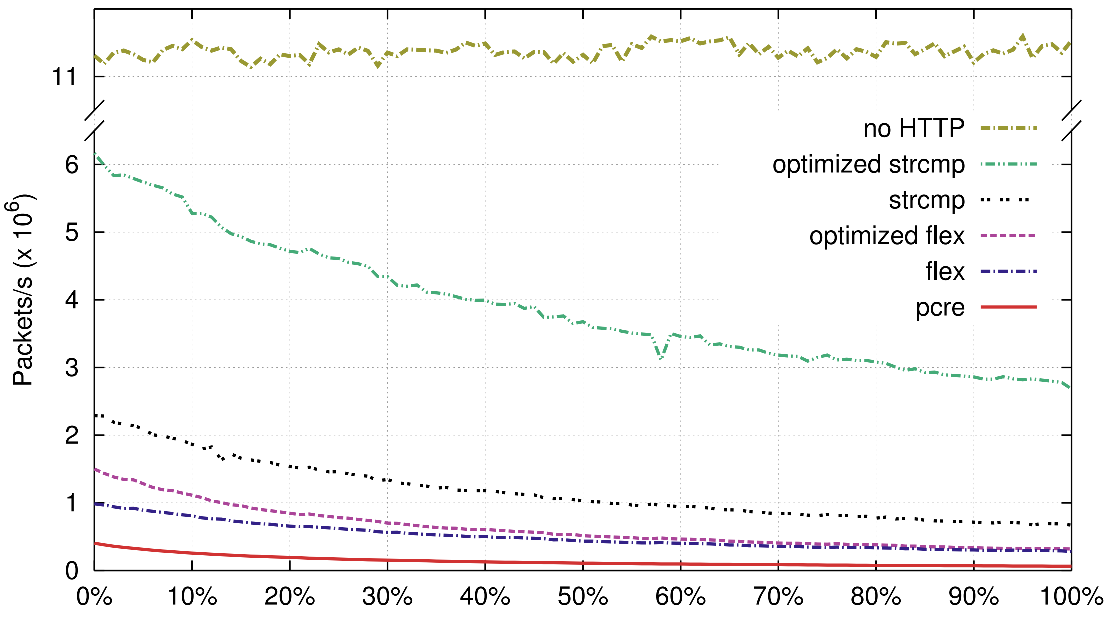


Figure 20: Parser performance comparison with respect to HTTP proportion (0 % - no HTTP, 100 % - only HTTP headers) in the traffic - full packets 1500 B.

The main difference between *flex* and *optimized flex* parsers is in the automaton initialization process. By rewriting the initialization process, we achieved slight performance improvement, which is noticeable mainly in the $\langle 0\,\%, 20\,\% \rangle$ interval, where the actual HTTP parsing time is short. There is one other important factor affecting the *flex* parser performance. The *flex* automaton is designed to work with its own writable buffer since it marks the end of individual parsed tokens directly into the buffer. For this reason, a copy of the packet payload must be created before the actual parsing can start. To measure the impact of the copying, we created another two parser plugins called *empty* and *copy*. First, we measured the IP flow meter throughput with *empty* plugin which performs no data parsing, then with *copy* plugin which only copies packet payload to a static buffer. From the results, we estimate the throughput the *optimized flex* parser would have without the memory copying. The performance of the *optimized flex* parser would be about 2.4 million packets per second for 0 % and 0.33 million packets per second for 100 % HTTP packets. This shows that the actual HTTP parsing when compared to *strcmp* parser, is slightly faster for binary payload packets and slower for HTTP header packets.

The performance of the *pcre* parser is the lowest. The PCRE algorithm converts the regular expression to a tree structure and then performs a depth-first search while reading the input string. In case there is no match in the current tree branch, the algorithm backs up and tries another one. Therefore, for a complex regular expression, the pattern matching is not that fast as simple string search using functions like *strcmp*. Another reason why the *pcre* parser is not fast is that it performs all searches on whole packet payload. The other algorithms are processing the data sequentially.

Figure 21 shows the results for truncated packets. The *optimized strcmp* and *no HTTP* are only slightly faster since the truncating of the packets has a positive impact on CPU data cache utilization. The *strcmp* algorithm is flawed since its throughput on HTTP packets deteriorates rapidly. This shows the disadvantage of hand-written parsers, as they are more error-prone than the generated ones. The *pcre* parser performance is almost doubled, as the repeatedly processed data are truncated. The *flex*-based parser also achieve performance increase, since the memory copying costs are reduced for smaller data.

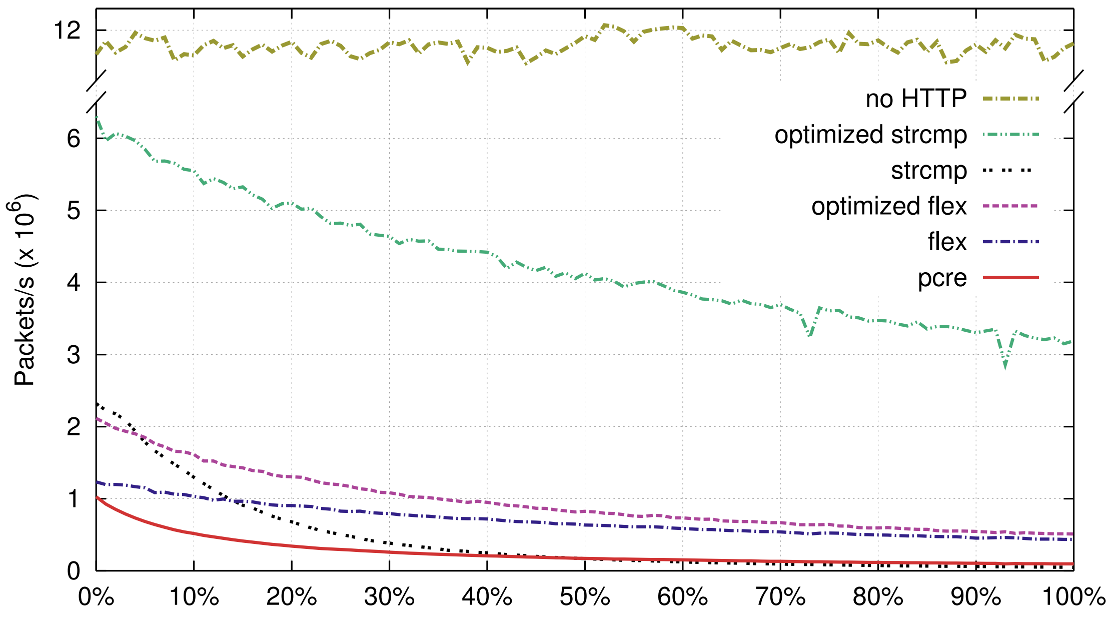


Figure 21: Parser performance comparison with respect to HTTP proportion (0 % - no HTTP, 100 % - only HTTP headers) in the traffic - truncated packets 384 B.

Parsed HTTP Header Fields Impact

This case study was designed to show the impact of a number of parsed HTTP header fields on the parser performance.

When payload packets are detected, they do not have their content parsed for additional HTTP header fields. Therefore, a test set containing only HTTP request and response packets was used. The case study starts with an empty plugin, that does not parse HTTP header fields and just labels the HTTP packets. In the next steps, we cumulatively add header field to parse until we parse all of the seven supported fields. We run the tests for both full and truncated packets. The average performance of the parsers for each of the added field is shown in Figure 22.

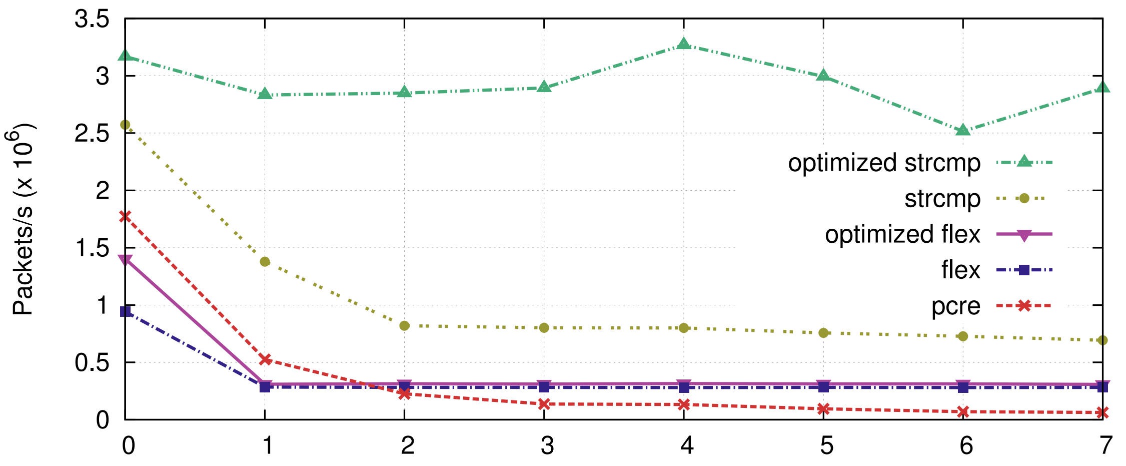


Figure 22: An HTTP parser throughput for 1500 B packets; supported fields - (0) *none* - HTTP protocol labeling, (1) *+host*, (2) *+method*, (3) *+status code*, (4) *+request URI*, (5) *+content type*, (6) *+referer*, (7) *+user agent*.

Only the request and response packets are parsed, thus the values for the seven fields parsed in the Figure 22 correspond to the 100 % packet/s values in the Figure 20 and Figure 21. For the same reason the parsed packets per second numbers are lower in comparison with the Figure 20 and Figure 21. The performance of *strcmp* and *pcre* parsers drops with each additional parsed HTTP header field. The *optimized strcmp* parser implementation details attentional fluctuation effect on performance shown in Figure 22. An example is the performance increase when adding a *(4) request URI* or a *(3) status code*. It is caused by extra code snippet that extracts the URI so that this line is not processed by the more generic code designed for parsing other header fields. Due to the usage of the finite automaton, the data is always processed in one pass by the *flex*-based algorithms. Therefore, they retain the same level of performance for all additional fields. This feature could be used to automatically build powerful parsers when a large number of parsed application fields would make it ineffective to create hand-written parsers.

Same as in the previous case study, the parsers processing truncated packets show better performance than the parsers working on full packets.

Packet Content Effect

This case study investigates the possible effects of the position of the CR and LF control characters in the packet payload on the parser performance. The mentioned ASCII characters represent the end of a line in the HTTP header. Some of the proposed algorithms use these characters as the trigger to stop parsing. Therefore the position of these characters affects the performance of the parser. The packets with the CRLF characters at the very *beginning* should be parsed faster than the packets having the CRLF at the *end* since the algorithm terminates as soon as it identifies the CRLF characters. The test sets with modified binary payload packets (see Section 4.1.3) enables us to compare the algorithms taking into account this perspective.

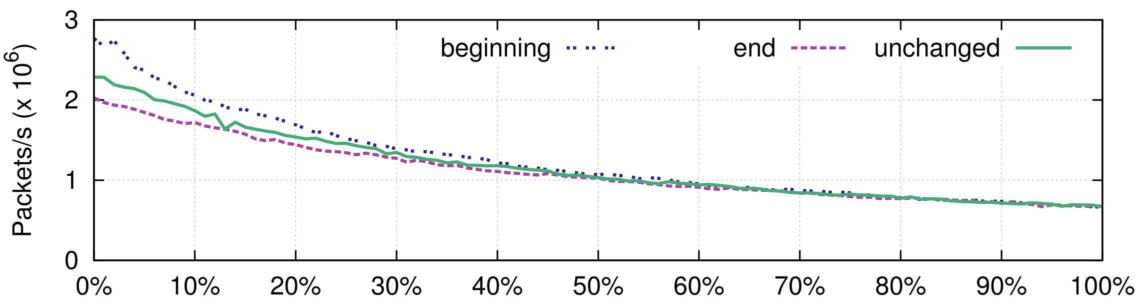


Figure 23: Packet content effect - packet length 1500 B.

We used the modified binary payload packets to test the parsers. The parsing algorithms, except the *strcmp* algorithm, show an insignificant difference in their performance for all variants of the modified packets. The *pcre* and *optimized strcmp* parsers do not search for the end of line characters in order to label the packet, therefore this test does not affect them. The *flex*-based algorithms are not significantly affected since they stop parsing on the first character that is not expected in HTTP header and therefore stop at the first character in any case. The *strcmp* parser depends on the search for the end of line characters, which is confirmed by Figure 23. The sooner the characters are found, the faster the algorithm terminates. The scenario with truncated packets is different, since the performance on *end* dataset is greater than on *unchanged* data. This is caused by removing the end of packet payload together with the end of line character. When the *strcmp* algorithm cannot find the character, it terminates immediately without trying to search the data. Therefore it terminates sooner than on *unchanged* dataset, where the end of line character is found, and the search continues.

4.1.5 Summary

In this section, we have assessed the impacts of HTTP protocol analysis on IP flow monitoring performance. We implemented the state-of-the-art approaches to HTTP protocol parsing. Moreover, the new *flex*-based HTTP parser was designed, and its performance was compared to the other approaches.

The evaluation shows that in our case the hand-written and carefully optimized parser performs significantly better than implementations with automated parsing. It also shows that the new *flex*-based implementations handle the increasing number of parsed HTTP fields without significant performance loss. Truncating the packets before HTTP protocol parsing can increase the parser throughput. The performance comparison of *no HTTP* parser with HTTP parsers shows that providing application visibility is a demanding task. Current approaches to the application protocol parsing may not be effective enough to process high-speed network traffic.

Our *flex*-based parser and evaluation experiments affect the CSA in the following ways. We can provide an operator with additional valuable information without a significant decrease in the application parser's performance. Nevertheless, we have shown that application visibility to network flow is a demanding task. The throughput of the IP flow monitoring tool decreases below 50 % when we turn HTTP parsing on. The trade-off between performance and available information is demonstrated by our performance comparison experiment. Last, but not least, the comparison of the parsers can be used for selection of the optimal HTTP parser for an IP flow CSA framework.

4.2 HTTPS Traffic Analysis for Client Identification

The rising popularity of encrypted network traffic is a double-edged sword. On the one hand, it provides secure data transmission, protects against eavesdropping, and improves the trustworthiness of communicating hosts. On the other hand, it complicates the legitimate monitoring of network traffic, including traffic classification and host identification. Nowadays, we are able to monitor, identify, and classify plain-text network traffic, such as HTTP, but it is hard to analyze encrypted communication. The more secure the connection is, from the point of view of communicating partners, the harder it is to understand the network traffic, identify anomalous and malicious activity, and reach cyber situation awareness.

In this section, we discuss HTTPS - HTTP over SSL/TLS, the most common encrypted network traffic protocols. In a communication encrypted by SSL/TLS, the hosts have to first agree on encryption methods and their parameters. Therefore, the initial packets contain unencrypted messages with information about the client and the server. This information varies among different clients and their versions. Similar client identifier is a User-Agent value in a HTTP header, which is commonly used for identifying the client and classifying traffic. However, only the SSL/TLS handshake can be observed in a HTTPS connection without decrypting the payload. Therefore, we approach the problem of identifying the SSL/TLS client and classifying HTTPS traffic by building up a dictionary of SSL/TLS handshake fingerprints and their corresponding User-Agents.

We set up an experiment, which aims to investigate following three research areas: *(i)* Parameters of a SSL/TLS handshake usable for client identification, *(ii)* pairing selected SSL/TLS handshake parameters and HTTP header fields, and *(iii)* volume of information, i. e., number of known SSL/TLS parameters and pairings to HTTP headers, needed for the analysis of a significant portion of network traffic.

First, we aim to observe real network traffic to gain insight into contemporary SSL/TLS handshakes. We deploy network traffic monitoring, filter HTTPS connections, and create a list of the SSL/TLS handshakes and their fingerprints. We focus on analyzing information provided during the handshake by the client, i.e., the *ClientHello* message containing the protocol version, the list of supported cipher suites, and other data. Apart from the useful information for identifying the client, we are particularly interested in the share of old and vulnerable protocol versions. Recent discoveries of severe vulnerabilities, such as POODLE [178], might have significantly changed the proportion of protocol versions in use.

Second, we correlate selected parts of SSL/TLS handshakes and HTTP headers. We suppose that the list of supported cipher suites (declared by the client in the *ClientHello* message) can be used as an identifier similarly to a User-Agent in a HTTP header. However, it is not possible to get the User-Agent from the HTTPS request without decryption. We use two approaches to obtain pairs of cipher suite lists and corresponding User-Agents. The host-based approach is based on advanced logging on the server side. The novel network-based method is based on simultaneous monitoring of HTTP and HTTPS connections. The information from HTTP headers is obtained using HTTP IP flow probe described in the previous section. We assume that clients mostly communicate on both protocols. Therefore, we look for HTTP and HTTPS connections from the same client over a short time period and pair cipher suites and User-Agents from such connections.

Third, we use the pairs of SSL/TLS fingerprints and User-Agents as a dictionary to assign User-Agents to the HTTPS connections observed during the measurement. We discuss the quality of the obtained pairs with respect to the dictionary size and accuracy of the User-Agent estimation. The goal of this part is to estimate the size and accuracy of a dictionary which could be used for identifying clients on a large-scale and classifying network traffic.

The motivation for our work came from three main areas of applications. Firstly, it is the analysis of encrypted network traffic, including the identification and characterization of encrypted network traffic and classification of communicating clients. From this point of view, we are not interested in individual clients, but rather overall characteristics of network traffic. The second area of interest is the host identification, which aims to obtain information on an individual host. This applies mostly on a web browser fingerprinting. The third area is network security and forensics, where we typically want to detect the activity of a specific network host, detect malicious clients, and evaluate the activity of unknown or unusual clients. Our research address the cyber-space related challenges of CSA, namely the complexity and dynamics challenges. We contribute to the area of a host identification in network traffic, which improves to possibilities of operator's orientation in a network even when only IP flow data are available.

Analysis of Encrypted Network Traffic

We have to understand the network traffic before we can proceed to client identification and detection of suspicious or even malicious activity. Therefore, we have to observe network traffic to get insight into typical patterns. Specifically, we have to retrieve a record of encrypted network traffic containing as many different patterns as possible. To motivate our work, we decided to analyze real network traffic in a campus network instead of generating the traffic patterns in a laboratory environment. Therefore, we can get more interesting results which are not necessarily related to the proposed experiment. These results can later be useful for network administrators, security practitioners, and the scientific community.

We have to identify what are the options of establishing the SSL/TLS communication and which options are used in real traffic. We have to use methods from a survey by Velan et al. [179] as the basis and a real network data to identify these options. Then, we have to find which of the options are varying the most and if the variability of these options indicates different traffic patterns, e. g., different communicating partners or type of traffic.

Client Identification and Browser Fingerprinting

Having insight into the network traffic, we can proceed to the client identification and browser fingerprinting. The client identification and browser fingerprinting contribute significantly to network security and detection of malicious activities, e. g., by outdated systems identification or unusual behavior detection. Identification and fingerprinting are moreover useful for commercial purposes (targeting ads, price discrimination, assessing financial credibility), network accounting and client behavior monitoring [180].

Assuming the clients have unique fingerprints, the fingerprinting can be used for advanced traffic analysis. For example, we can enumerate unique client fingerprints that share the same IP address as depicted in Figure 24. This approach can help in enumerating the number of users of a specific machine, the presence of a NAT mechanism and number of clients behind a NAT, and so forth.

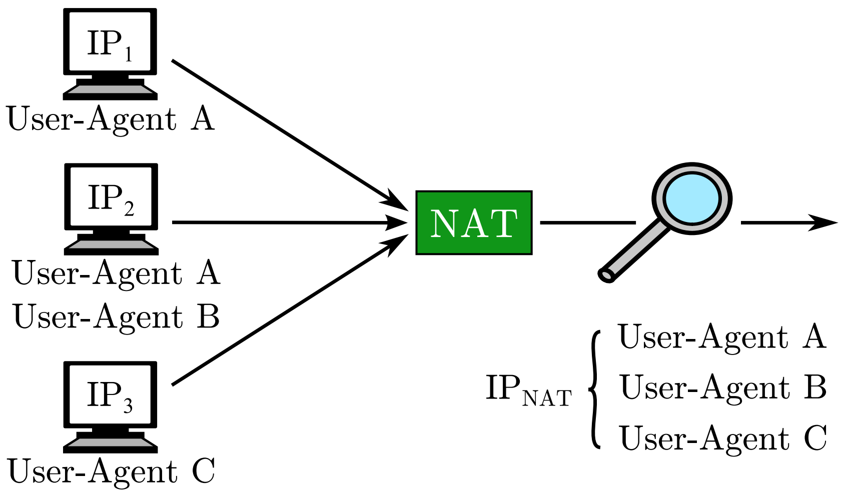


Figure 24: Identification and enumeration of HTTPS clients behind NAT.

A problem of client identification based on HTTP User-Agent is that the User-Agent string can be easily manipulated. For example, illegitimate web crawlers and bots typically spoof the User-Agent string as to be mistaken for legitimate ones such as Googlebot [181]. Manipulation with User-Agent string does not apply only to malicious clients, but also to legitimate clients. Historically, the web browsers were adding identifiers of each other to their User-Agent to resolve compatibility issues with certain web pages. Therefore, the Internet Explorer includes "Mozilla" in its User-Agent string and Android browsers claim themselves to be Safari web browser. Many browsers also offer user-friendly option to completely replace a User-Agent with arbitrary string or identifier of a different browser. To sum it up, there is a constant risk that the User-Agent string is forged and the results of any work based on User-Agent string analysis cannot be trusted.

Network Security and Forensics

Network security and forensics have their unique preferences and points of view in comparison to network traffic analysis and client fingerprinting. Ordinary network traffic is not a primary interest of network security monitoring. Instead, the abnormal and previously unknown traffic patterns are in question. Of course, we have to understand the network traffic first to recognize common traffic patterns. Then it is possible to focus on unusual events and, in our case, fingerprints to detect suspicious or straightly malicious clients and their activity.

The case studies in network security and forensics are dealing with malware and vulnerability exploitation. For example, Win32/Hotbar is a malware, whose activity can be detected by searching for HTTP requests with a specific User-Agent [182]. Another example is related to Shellshock, Bash vulnerability disclosed in 2014. Shellshock can be exploited via HTTP requests containing strings starting with "`() { ;; };`" in values of various headers, which could be processed by some script at a server side. The attack can be detected by checking the characteristic sequence in HTTP request headers. In both cases, we face the problem of malware detection method that is easily performed over HTTP but is hard to accomplish over HTTPS traffic.

The overall trends in network communication affect almost all the communicating hosts, including the malicious ones. The prime examples are botnets which use HTTP(S) for communication between bots and command and control centers (C&C). Modern botnets are switching to HTTPS for similar reasons as legitimate clients. For example, the bots are likely to accept commands only from trusted C&C centers. HTTPS provides a certain level of trust and, also, prevents eavesdropping of the communication, on which the detection mechanisms typically rely [183].

Keeping in mind that almost everything is switching to encrypted communication protocols, we may expect a demand for methods of encrypted traffic analysis for security purposes. Clients fingerprinting seems to be a suitable option as it does not interfere with the encrypted content of a communication and thus cannot be considered as a violation of privacy. The question, however, is if the fingerprinting can distinguish legitimate and suspicious clients. For example, malicious actions executed using common web browser are not any different from legitimate traffic from the fingerprinting perspective. However, the malicious clients are often created for a specific purpose, and their implementation complies to this. Therefore, we may assume that certain malicious clients have unique fingerprints. This applies namely to self-propagating malware and bots communicating within a botnet.

This section is divided into four subsections. Subsection 4.2.1 presents a brief introduction into SSL/TLS protocols and relevant state-of-the-art. The design of the experiment, measurement tools, and measurement environment are described in Subsection 4.2.2. The results are presented in Subsection 4.2.3 and evaluated in Subsection 4.2.4.

4.2.1 State-of-the-art

We describe a brief introduction to SSL/TLS protocols and a survey of network-based SSL/TLS analysis in this subsection. Further, we present a short survey of case studies in network forensics and related fields.

Transport Layer Security (TLS) [184] is a new version of the Secure Sockets Layer version 3 (SSLv3) protocol [185], which is no longer recommended for use due to its security vulnerabilities. It provides confidentiality, data integrity, non-repudiation, replay protection, and authentication through digital certificates directly on top of the TCP protocol. The TLS protocol is currently used for securing the most common network protocols, such as HTTP, FTP, and SMTP, and is part of Voice over Internet Protocol (VoIP) and Virtual Private Network (VPN) protocols. In this paper, we focus on SSL/TLS's use within the HTTP protocol, known as HTTPS [186], which is the most common use of the TLS.

The TLS connection can be divided into two phases: an initial handshake and application data transfer, both depicted in detail in Figure 25. The initial handshake begins with a *ClientHello* message identifying which protocol version is used, the cipher suite list, and extensions. The full list of these identifiers is available on the IANA web page [187]. The following messages of the initial handshake are used for peer authentication using X.509 certificates [188] and shared secret establishment based on agreed parameters. All TLS messages exchanged during the initial handshake are not encrypted until shared keys are established and confirmed by *Finished* messages. The TLS protocol consists of configurable cryptographic algorithms and different sub-protocols which form a layered design [189]. The main part of this design is the Record

Protocol [184], which is used as an envelope for all TLS messages and encrypted application data.

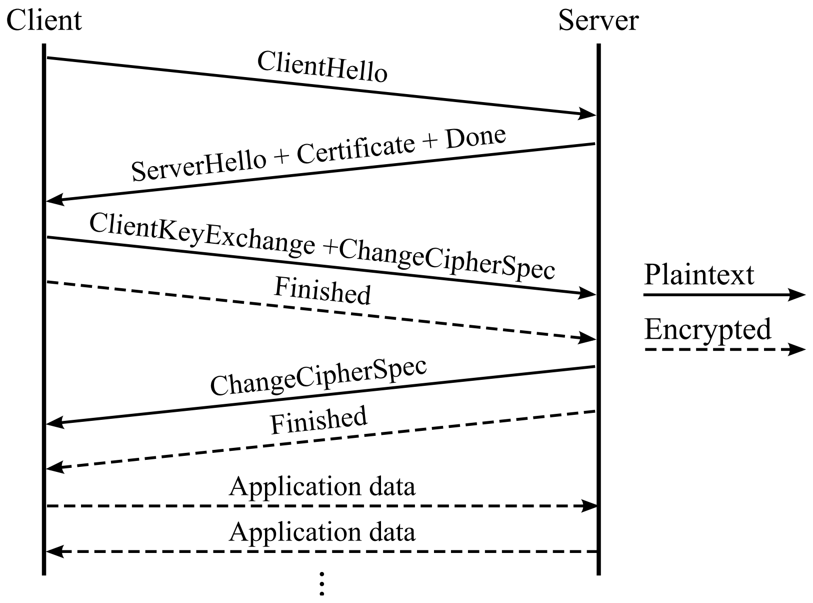


Figure 25: A SSL/TLS handshake.

The long-term SSL/TLS traffic monitoring and measurement on the Internet were presented by Levillain et al. [190] for the period 2010–2011. They observed a lot of servers which were intolerant to some cipher suite lists and detected certificate chains, which did not comply with the standard. Another study of SSL traffic was conducted by Holz et al. [191], who focused on SSL/TLS certificate properties. They revealed a great number of invalid certificates and certificates shared among a large number of hosts. The work of Holz et al. was followed by the work of Durumeric et al. [192] which focused on an assessment of certification authorities.

The SSL/TLS protocol and its applications are analyzed by Qualys SSL Lab [193]. Apart from SSL/TLS applications testing, they presented the idea of HTTP client fingerprinting using an analysis of the SSL/TLS handshake. The idea was implemented in the *SSLhaf* [194] proof-of-concept tool for a simultaneous host-based analysis of HTTP and SSL/TLS connections. A brief analysis of fingerprints of common web browsers based on the tool was published at Internet Storm Center [195]. The idea was also implemented by Majkovski [196] as the *p0f* tool module used for fingerprinting operating systems. Another idea was presented by Bernaille and Teixeira [197], who identified underlying applications in a SSL encrypted connection by the first SSL/TLS packet size.

A survey of web tracking and its mechanisms was proposed by Bujlow et al. [180]. The authors placed a special focus on fingerprinting as it is rich in various methods. The survey proposed a taxonomy of tracking mechanisms including a category of fingerprinting. The approaches relevant to our work are *Device fingerprinting*, *Operating System instance fingerprinting*, and *Browser instance fingerprinting*. The methods are mostly focused on the analysis of HTTP headers. However, TCP/IP headers can also be used, e. g., for OS fingerprinting. OS can be detected using information from network flows (TTL, SYN packet size, TCP window size, User-Agent) [A7], DNS traffic analysis [198] or using a combination of the previous and prebuilt dictionary such as in *p0f* tool [199]. Regarding the client identification, remote psychical client fingerprinting using clock skews was described by Kohno et al. [200]. Graph-based structures for client profiling were introduced by Karagiannis et al. [201]. Manual labeling of fingerprints, a drawback of fingerprint-based methods, was tackled by Abdelnur et al. [202]. The authors intended to solve the problem by automation of fingerprint signature creation.

Regarding the browser fingerprinting, detection using User-Agent field is considered not to be reliable, since the field is set by a browser and can be easily forged. Instead, fingerprinting methods based on CSS and HTML5 fingerprinting were introduced by Unger et al. [203]. The JavaScript engine was used for browser identification by Mulazzani et al. [204]. *Panopticlick project* [205] uses an active approach in collecting fingerprints of a browser. Collected features are browser plugin details, system fonts, cookie settings, screen size, and others. Nevertheless, all of these techniques can be overcome by using a web proxy [206] or TOR [207]. In these cases, detectable fingerprints are removed or replaced by custom ones.

The case studies of network forensic analysis, including analysis based on User-Agent identification, were presented by Raftopoulos and Dimitropoulos [182]. They cited Win32/Hotbar as an example of malware, whose activity can be detected by searching for HTTP requests with a specific User-Agent. One related problem is the identification of network address translation (NAT) in the network. Traffic flow-based methods were proposed by Gokcen et al. [208] and Krmíček et al. [209].

4.2.2 Experiment Design

We designed a three-phase experiment to answer our research questions and verify the idea of using HTTPS client identification with SSL/TLS fingerprinting. In the first phase, we set up a measurement of live network traffic in the campus network of Masaryk University. The monitoring was primarily focused on SSL/TLS connections. In the second phase, we created a dictionary of SSL/TLS fingerprints and HTTP User-Agents, based on an analysis of the captured network traffic. In the third phase, we applied this dictionary to assign User-Agents to the measured traffic and verified the capabilities of HTTPS client identification.

SSL/TLS Traffic Measurement

We measured live network traffic in the campus network of Masaryk University. The network has more than 40,000 users and 15,000 active IP addresses on average per day. We deployed an IP flow probe in a 10 Gbps link that connects the university and the network of CESNET, Czech National Research and Education Network (NREN).Each measured standard IP flow record contained following flow keys:

$$f = (proto, srcIP, dstIP, srcPort, dstPort, tSt),$$

where $proto$, $srcIP$, $dstIP$, $srcPort$ and $dstPort$ are the shared values of IP flow keys and tSt is the timestamp of the IP flow.

Since the standard IP flow record does not contain detailed information about HTTP and HTTPS traffic, we used two extensions for IP flow measurement, which add new elements to the IP flow record. The first extension adds a User-Agent (ua) element to a HTTP IP flow based on work in [A4]. Only HTTP IP flows with destination port 80 were considered. The set of all extended IP flows with assigned User-Agent will be denoted F_{HTTP}.

$$\begin{aligned} F_{HTTP} = \{\, (\, f,\ ua\,) \mid f.dstPort = 80 \\ \wedge\ ua \neq null \,\} \end{aligned}$$

The second IP flow measurement extension adds elements from the *ClientHello* message exchanged during the initial SSL/TLS handshake of the HTTPS connection. We measured only those elements which do not change with each client connection, namely the SSL/TLS protocol version (vr), cipher suite list (cs), compression (cm), and TLS extensions (ex). The set of all extended HTTPS flows will be denoted F_{HTTPS}.

$$\begin{aligned} Hello &= (\, vr,\ cs,\ cm,\ ex\,) \\ F_{HTTPS} &= \{\ (f,\ Hello) \mid f.dstPort = 443 \\ &\qquad \wedge\ Hello \neq null\ \} \end{aligned}$$

The aim of the measurement was to get the base data for the subsequent phases of the experiment. In the next phase, we created a dictionary which allowed us to transform elements of the SSL/TLS fingerprint into HTTP User-Agents. This dictionary was then applied to all the measured data to verify its usability and gain more information about HTTPS clients. The secondary aim of the measurement was to get a closer look at SSL/TLS connections and to obtain basic statistics about the network traffic, primarily focusing on SSL/TLS traffic.

Pairing Cipher Suite Lists and User-Agents

To identify HTTPS clients, it is necessary to create a dictionary containing pairs of SSL/TLS handshake elements and User-Agents. This represents the second phase of our experiment. We decided to use only a cipher suite list from the *ClientHello* message to build up a dictionary. Cipher suite lists are the most varied elements of the SSL/TLS handshake, and we supposed that they are sufficient for identifying clients. Other elements of the handshake only have a few different values. Therefore, we did not plan to include them in the dictionary. However, we assumed they could clarify ambiguous results.

We took two approaches, host-based and IP flow-based, to pair a cipher suite list to a User-Agent. The host-based method uses the information from a single HTTPS connection on the server side, where the unencrypted data including the HTTP header, are available. This method is very accurate, but it requires clients to visit the server where the monitoring is deployed. We set up a HTTPS server running Apache web server and *SSLhaf* plugin [194]. *SSLhaf* enabled us to log the SSL/TLS parameters of a HTTPS connection. We logged SSL/TLS connection parameters, including the cipher suite list in a *ClientHello* message, and the User-Agent from the HTTP header for each incoming connection. The dictionary was created by a simple combination of the cipher suite list and the User-Agent from a single connection.

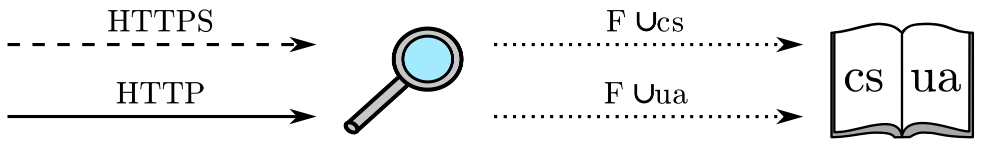


Figure 26: A flow-based cipher suite list and User-Agent pairing.

The IP flow-based method is based on network monitoring, the extraction of cipher suite lists and User-Agents, and the correlation of HTTP and HTTPS connections from a single client, see Figure 26. We assumed that web clients commonly communicate via both HTTP and HTTPS protocols. SSL/TLS connections monitoring, as well as HTTP monitoring, was utilized in this phase of the experiment. The method of pairing cipher suite lists and User-Agents can be described as follows. Let CS denote the set of all possible cipher suite lists and UA the set of all user-agents, then:

$$\begin{aligned} Dict = \{\, (cs,\ ua)\ \in\ CS \times UA \mid \\ &\exists\, f_S \in F_{HTTPS},\ \exists\, f \in F_{HTTP} : \\ &f_S.cs = cs \land f.ua = ua \\ &\land\ f_S.srcIP = f.srcIP \\ &\land\ \forall f' \in F_{HTTP},\ \forall f'_S \in F_{HTTPS} : \\ &|f'.tSt - f_S.tSt| \geq |f.tSt - f_S.tSt| \\ &\land\ |f.tSt - f'_S.tSt| \geq |f.tSt - f_S.tSt|\ \} \end{aligned}$$

We searched for HTTP and HTTPS connections with the same source IP address. We selected a cipher suite list from the HTTPS connections and paired it to the User-Agent from the HTTP connection which was the closest in time. We assumed that the flow-based approach would better reflect the structure of live network traffic and allow us to cover more cipher suite lists observed in the network.

We did not expect that the dictionary would provide an unambiguous translation of one cipher suite list to one User-Agent, but there would be one cipher suite list with more corresponding User-Agents and vice versa. However, we assumed that User-Agents assigned to one cipher suite list have only slight differences, such as the software version. Therefore, it does not affect the identification of general properties, e. g., the operating system or web browser. We also expected there to be some significant deviations caused by ambiguity in the flow-based approach or, for example, by forged connections by a malicious crawler pretending to be a legitimate search engine. In this case, we took only the most similar User-Agents substrings.

Assigning User-Agents to Measured HTTPS IP Flows

The third phase of our experiment was a conjunction of the results from the previous phases as depicted in Figure 27. We combined both types of pairs of cipher suite lists and User-Agents which were generated in the second phase. Then we applied them to the measured data from the first phase. The results contained HTTPS flows extended by information about the corresponding User-Agent or list of User-Agents:

$$\begin{aligned} F'_{HTTPS} = \{\ (f, ua)\ \in F_{HTTPS} \times UA \mid \\ (f.cs, ua) \in Dict\ \} \end{aligned}$$

We planned to validate the results of the assignment and verify our idea of HTTPS client identification using SSL/TLS fingerprinting. We were interested in the share of SSL/TLS traffic for which we were able to assign a correlating User-Agent. The connections containing cipher suite lists for which we failed to assign a User-Agent are potentially relevant from a security perspective. We expected most of the traffic to be initiated by common web browsers, for which we were able to easily get the pair of a cipher suite list and User-Agent using the host-based method. However, we expected that a combination of host-based and flow-based dictionaries was needed to cover the majority of the traffic.

If multiple User-Agents corresponding to a single cipher suite list were found, we planned to evaluate what is the minimal set of information provided by the set of User-Agents. For example, when multiple User-Agents corresponding to the same cipher suite list point to different versions of a client application. Similarities in grouped User-Agents can indicate at least if the client is a web browser, what is its operating system, or if it is a mobile device. Major differences would otherwise indicate an error in the pairing method.

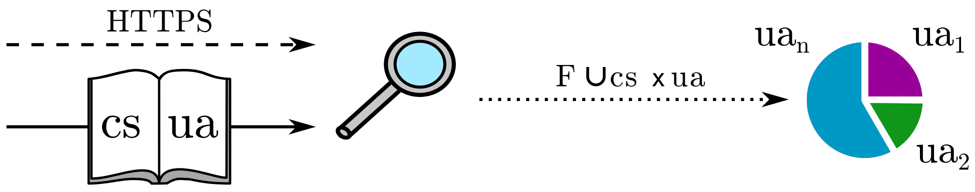


Figure 27: Assigning User-Agents to cipher suite lists.

4.2.3 Experiment Results

In this subsection, we present the results of the experiment. First, we sum up the results of measuring SSL/TLS connections in the campus network. Second, we describe the set of User-Agents and cipher suite list pairs obtained via the host-based and flow-based methods. The section closes with the results of assigning User-Agents to the SSL/TLS connection obtained in the first phase of the experiment.

Measurement

We conducted measurements over a 7 day period in January 2015. The network bandwidth utilization on the monitored network link ranged from 3 to 5 Gbps. We filtered the HTTPS connections, processed the SSL/TLS handshakes, and saved the content of *ClientHello* messages. The SSL/TLS version, cipher suite list, compression, and extensions were recorded for each connection. In total, we processed 85,250,090 HTTPS connections.

Version	Number of Connections
TLS 1.2	49,140,929
TLS 1.0	33,827,182
SSL 3.0	1,365,409
TLS 1.1	913,014
other	3,556

Table 3: Distribution of SSL/TLS versions.

The observed versions are listed in Table 3. Over 57 % of connections used the TLS 1.2 protocol followed by almost 40 % for TLS 1.0. Only 1.6 % of connections used the older and more vulnerable SSL 3.0 protocol. TLS 1.1 represented around 1 % of connections. The remaining connections were unrecognized. However, the number of such connections is insignificant.

We can confirm that only a small number of cipher suites and cipher suite lists cover the majority of live network traffic. As we can see in Figure 28, the top 10 cipher suite lists represented 68.5 % of live network traffic and top 31 (out of 1598) cipher suite lists were enough to cover 90 % of the traffic, and 121 cipher suite lists cover 99 % of the traffic.

Pairing Cipher Suite Lists and User-Agents

First, we created a base set of pairs using the host-based method. We manually contacted the monitoring server with available clients, such as web browsers and tools such as *curl* [210], to create an initial dataset. Therefore, most of the traffic incoming to the monitoring server was artificial. We then made the server publicly accessible and spread the links to lure more clients, such as web crawlers. In total, we obtained 72 unique cipher suite lists and 293 unique User-Agents, forming 307 pairs. Multiple User-Agents, with the same cipher suite list, were similar in most cases. The differences were usually in the version of the client in the User-Agent.

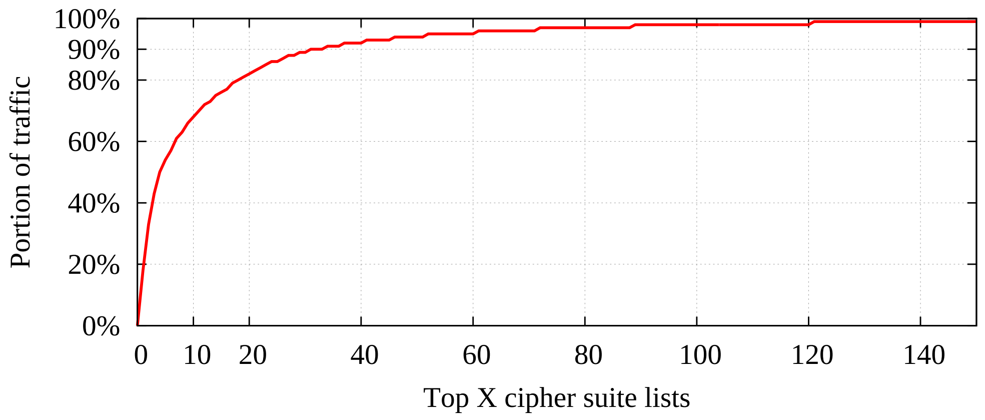


Figure 28: Network traffic represented by Top X cipher suite lists.

Then, we moved on to the flow-based method, i. e., combined monitoring of cipher suite lists from SSL/TLS handshakes in HTTPS connections and the User-Agents from HTTP connections. We analyzed a 1-hour sample of peak network traffic from our campus network and selected the hosts which initiated both HTTP and HTTPS connections. User-Agents from HTTP connections and cipher suite lists (from HTTPS connections) from the same client created a new pair. We observed 10,890 clients communicating on both protocols in a short period of time, 305 unique cipher suite lists, and 5,043 unique User-Agents. In total, we derived 12,832 unique pairs during the measurement.

Following this, we investigated the relationship between cipher suite lists and User-Agents by determining the cardinality of the relationship. Both methods provided more User-Agents which correspond to one cipher suite list, i. e., a 1:n relation. After a manual inspection, we discovered, that these User-Agents differ mostly in the system versions while the information about the client, e. g. browser type, stays more or less constant. Therefore, it is possible to identify a client with high accuracy. The flow-based method also generated a single User-Agent which corresponds to more cipher suite lists. However, this is most likely caused by inaccuracy in the method which cannot distinguish more clients communicating at the same time.

Assigning User-Agents to Measured HTTPS Flows

We used the dictionary provided by the host-based method and then filled in the supplemented results with a dictionary provided by the flow-based method. The host-based dictionary contained only 72 unique cipher suite lists, which represented 4.5 % of all cipher suite lists measured during the first phase. However, we observed that those unique cipher suite lists covered 78.0 % of all the measured HTTPS flows. When we combined the host-based and flow-based dictionaries, we obtained 316 unique cipher suite lists (19.8 % of all) covering 99.6 % of the measured HTTPS connections. Therefore, we assigned a User-Agent to almost all observed HTTPS connections using a combined dictionary based on data from a single server, and the correlations from a 1-hour sample of network traffic.

As we already mentioned, multiple User-Agents were assigned to one cipher suite list, which caused ambiguity in the translation. Even hundreds of different User-Agents were found to correspond to a single cipher suite list. However, we discovered that multiple User-Agents assigned to a single cipher suite list differed in details, such as the version of the used software. Figure 29 shows the results of the User-Agent assignment to the measured HTTPS connections according to the level of certainty. Certainty represents the number of User-Agents per one cipher suite list.

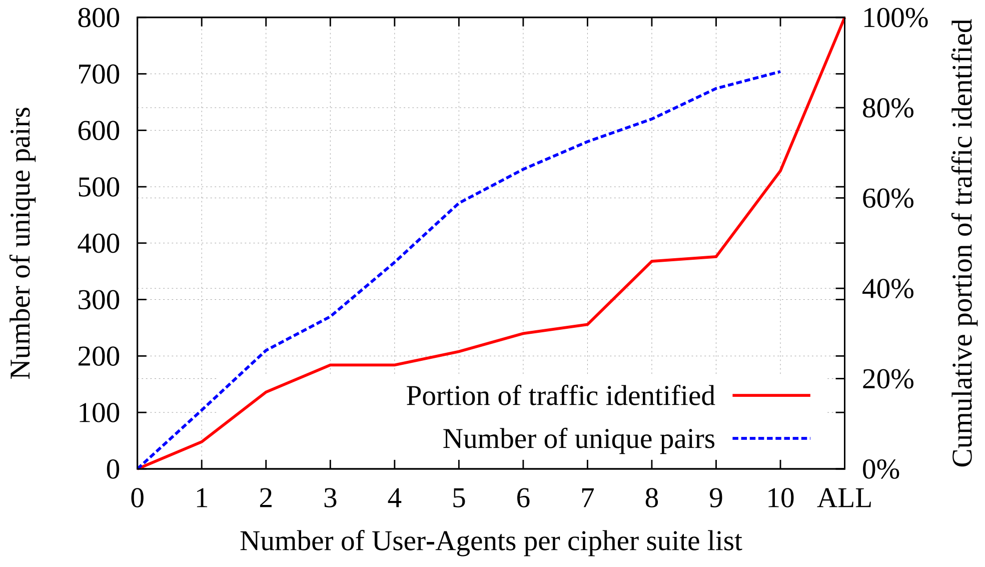


Figure 29: Relation of dictionary size and covered portion of network traffic.

The results show that in the event of unambiguous pairing, i. e., one User-Agent per one cipher suite list, the dictionary contained 104 unique pairs and covered only 6.3 % of all the measured HTTPS flows. If we gradually decreased the level of certainty, we were able to cover more cipher suite lists and a greater proportion of HTTPS flows. If we used up to 10 User-Agents per one cipher suite list, we were able to cover 66.0 % of all HTTP flows using 704 unique pairs with 253 unique cipher suite lists. In this case, User-Agents were relatively different, nevertheless, we were able to derive a general identification from the client, e. g., if it was a web browser, mobile device, or a web crawler.

4.2.4 Experiment Evaluation

In this subsection, we discuss the experiment's circumstances and results. First, we evaluate the measurement phase and shares of protocol versions and cipher suite lists in the observed network traffic. Second, the quality of dictionaries used for client identification is discussed. Methods are proposed which can be used to increase the accuracy of the dictionary. Finally, we evaluate the structure of the network traffic according to the estimated User-Agents and compare our results to the related work to estimate the credibility of our results.

Measurement

The plain measurement results were similar to our expectations. We analyzed the shares of SSL/TLS versions and cipher suite lists in the monitored connections. It is not surprising that the majority of the HTTPS connections used the latest TLS 1.2 protocol. However, high share of TLS 1.0 should not remain unnoticed. We further confirmed that the majority of SSL/TLS connections was represented only by a small number of cipher suites and cipher suite lists.

The interesting discovered fact was the 1.6 % share of SSL 3.0 in the observed HTTPS connections. The SSL 3.0 protocol is no longer considered safe due to serious vulnerabilities discovered in 2014, such as the POODLE attack [178]. We naturally wondered if the discovery of serious vulnerability in a protocol leads to a decrease in its usage. In comparison to earlier results, the share of connections initiated over SSL is decreasing. For example, Levillain et al. [190] reported 5 % share of SSL 3.0 in 2011.

Pairing Cipher Suite Lists and User-Agents

The host-based and IP flow-based method of pairing cipher suite lists and User-Agents provided diverse, but complementary, results. The host-based method was more precise and feasible in a controlled environment. However, the quantity of data depends on the popularity of the server where the monitoring is deployed. It could be interesting to perform host-based monitoring on a popular web server with a variety of clients. However, even the high attractiveness of a server does not guarantee capturing traffic from all the common clients in the network. Client applications like Spotify and Instagram, which communicate only to specific servers, are typical examples.

The IP flow-based method, focusing directly on clients, provided more pairs than the host-based method but at the cost of uncertainty. However, clients like web crawlers, uncommon clients, and even suspicious ones are hard to capture without access to a live network. The 1-hour sample of network traffic was sufficient to capture connections from almost all the different clients observed over a longer period of time.

The combination of both methods provided a usable dictionary which was sufficient for the needs of our experiment. The pairs obtained via the host-based method were also obtained via the IP flow-based method, which suggests that the IP flow-based method can provide acceptable results.

Assigning User-Agents to the Measurement Results

The assignment of User-Agents to the observed HTTPS connections was successful. The size of the dictionary was sufficient to describe almost every cipher suite list observed during the measurement. The number of connections with an unknown cipher suite list was negligible. The accuracy of the assignment can be disputed because more User-Agents corresponded to a single cipher suite list. However, multiple User-Agents with the same cipher suite list were typically similar. For example, similar User-Agents corresponding to a single cipher suite list are presented in Table 4.

To improve the quality and accuracy of the assignment, we have to improve the dictionary. We do not expect to gain more results from the host-based method without distributing the measurement among more attractive HTTPS servers. However, we can improve the quality of results in the IP flow-based method. We identified three approaches to enrich the dictionary and give precision to the data. The options are selecting the most suitable pair to put in the dictionary by manually inspecting the User-Agents or selecting statistically most significant values from repeated measurements and correlation of results to data from other sources.

First, we can manually check the results to find the most suitable pairs to put in the dictionary. This approach can significantly reduce the number of User-Agents assigned to a single cipher suite list which differ only in software version or other detail. It can also improve the quality of client type grouping. For example, User-Agents of mobile devices often include the type of hardware. However, this is laborious and error-prone.

A more convenient way to improve the quality of the results is to repeat the experiment to get statistically significant data. We assumed that the clients communicate on both HTTP and HTTPS protocols in near time. However, the two connections from the same client may be initiated by a different software client. Repeating the experiment in different time windows or even different network settings would provide slightly different results due to random errors and inconsistencies. The resulting dictionary would not be based on a single measurement nor union of all of them. Instead, only the pairs which appear in the results of multiple experiments would be added to the resulting dictionary.

Another approach to improving the results is a correlation to additional data. Considering only the network-based method, we can extend the fingerprinting to TCP/IP. The operating sys-

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.132 Safari/537.36 OPR/21.0.1432.67
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36 OPR/22.0.1471.70
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36 OPR/23.0.1522.77
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.94 Safari/537.36 OPR/24.0.1558.53
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2050.0 Iron/38.0.2150.0 Safari/537.36
Mozilla/5.0 (Windows NT 6.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36 Sleipnir/6.1.2
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.65 Safari/537.36

Table 4: User-Agents corresponding to a single cipher suite list.

tem of a client can be estimated by TTL value, TCP SYN packet size, and TCP Window Size [A7]. The User-Agent strings often include identifiers of operating systems, which may be correlated to the operating systems estimated by the TCP/IP fingerprinting. In addition, web crawlers can be identified by their hostname or WHOIS record. For example, reverse DNS query may validate User-Agent of clients such as Googlebot.

4.2.5 Summary

In this section, we have shown that it is possible to estimate the User-Agent of a client in HTTPS communication. This was done for further identifying the client using network monitoring and fingerprinting the SSL/TLS handshake. We designed an experiment in which we measured HTTPS traffic in a campus network. We processed only the initial SSL/TLS handshake in which the client and server negotiate the parameters of the encryption. Therefore, our approach was lightweight and avoided decrypting traffic. The created dictionary allows for the enhanced perception of encrypted network traffic for cyber situation awareness.

First, we investigated the parameters of the SSL/TLS handshake, which could be used to identify the client. The client identifies itself in a *ClientHello* message during the handshake. The most various part of the *ClientHello* was the list of cipher suites supported by the client. The cipher suite list differed among various client applications and their versions, which made it suitable for further identification. In total, we observed 305 unique cipher suite lists during our measurement. The other parts of the *ClientHello* message, such as the SSL/TLS version, compression, and supported extensions, were interesting for analysis but unusable for client identification due to the limited number of distinct values.

Second, we studied the relationship between cipher suite lists and HTTP User-Agents. The User-Agent is a common client identifier in HTTP. However, in HTTPS, it is not directly accessible without decrypting the transferred data. We deployed two methods for monitoring SS-L/TLS handshakes and HTTP headers simultaneously in order to pair cipher suite lists and

User-Agents. The host-based method, i. e., measurement on the server side, provided accurate results. However, this method was limited by the set of clients accessing the monitoring server, and we obtained a smaller number of pairs. The flow-based method used network monitoring and was not limited to a single server. We were looking for clients communicating on HTTP and HTTPS protocols over a short period of time and paired the observed cipher suite lists and User-Agents from both connections. We gained a large dictionary of more than 12,000 pairs. However, this method was less accurate compared to the host-based method.

Third, we assigned the corresponding User-Agents from the dictionary to the results from monitoring the SSL/TLS connections and discussed the required size and accuracy of the dictionary. We found that we need a dictionary of about 300 cipher suite lists with assigned User-Agents. Therefore, the dictionary which was created using the host-based method was not sufficient to cover all the distinct cipher suite lists which appeared in network traffic. On the other hand, only a 1-hour sample of the HTTPS traffic contained almost all the cipher suite lists which were observed over the week-long measurement. Therefore, we used the dictionary obtained using the flow-based method. However, many cipher suite lists were paired with more than one User-Agent. We were able to assign a User-Agent to almost every observed cipher suite list with a certain level of probability. Fortunately, a lot of User-Agents which corresponded to a single cipher suite list shared the same client identifier and differed only in their version or a similarly attainable value.

In summary, our work enhanced the capabilities of network forensics by introducing the network-based identification of HTTPS clients. Our IP flow-based approach is lightweight, not limited to a single server, and does not approach the encrypted data. Therefore, we can identify clients while preserving the communication's privacy. Our results are applicable for identifying clients in the network and serve for enhancing methods of network perception.

4.3 Summary

This chapter presents our research efforts in the area of CSA perception. Namely, we investigated the performance of IP flow monitoring probes with respect to the volume of collected information and possibilities of retrieving information from encrypted network traffic.

First, we demonstrated how the collection of additional information from HTTP protocol decreases the performance of IP flow monitoring probes. The throughput of the IP flow monitoring probes decreases by 80 % , when HTTP information is retrieved using the available HTTP protocol parsers. We introduced optimized versions of HTTP protocol parsers that show only 50 % performance decrease compared probes that parse no application information. The second major finding was that the throughput of a probe does not decrease with additional HTTP information parsed once some HTTP information is already parsed. This research extends our knowledge of the impact of the volume of information collected on the performance of the IP flow monitoring workflow.

Second, we investigated the possibilities of identification of a client in encrypted network traffic. We focused on HTTPS traffic and identified a data collection approach that enables to identify a client in HTTPS network traffic to some extent. We propose to create a dictionary that links Cipher Suites obtained from HTTPS network traffic with User-Agent obtained from unencrypted HTTP traffic. Both information can be collected using IP flow monitoring. We evaluate the dictionary and present its coverage of the traffic and the uniqueness of the client identification. The dictionary enables to identify a client unambiguously in 6.3 % of encrypted network traffic. The general client identification, i.e., type of device - browser, mobile device, web crawler, is available for all encrypted HTTPS traffic.

The main contribution of this chapter is the joint contribution of the author’s peer-reviewed publications [A4–A6], among others

- creation of optimized HTTP parser,
- evaluation of the performance of IP flow probes capable of exporting information from HTTP protocol,
- identification of SSL/TLS protocol headers suitable for client identification,
- proposal of methodology for client identification in encrypted traffic using a dictionary,
- creation and evaluation of the dictionary for client identification.

5
Data Comprehension

Data comprehension enables us to interpret information contained in the monitored data correctly. The data comprehension in the context of Ensley's CSA model is represented by IP flow record collection and analysis. In this section, we present our research contributions to IP flow record analysis that advance the understanding of the nature of network traffic and consequently improve an operator's CSA. We present results from several experiments that investigate the nature and complexion of different subsets of network traffic. We believe that our measurements and their evaluation bring light into previously not easily visible and comprehensible behavior of network traffic.

We choose to focus our research efforts on the following three areas: evaluation of properties of Top N statistics, analysis of tunneled network traffic, and retrieval of host-related information. The Top N statistics is frequently used in IP flow record analysis. We investigate the nature of this statistics and discuss its properties. Further, we research the statistics information value for host identification in network traffic. The statistics is evaluated concerning its availability, uniqueness and time stability. Our work on the analysis of tunneled traffic aims attention at the transition mechanisms used for tunneling IPv6 network traffic over IPv4 networks. The analysis reacts to the increasing adoption of IPv6. Nevertheless, the majority of networks is still IPv4-based. Therefore, the portion of tunneled IPv6 traffic increases as well. Our monitoring experiment investigates the nature of the tunneled and tunneling network traffic, namely the distribution of TTL and HOP values, country distribution, packet size distribution, IP flow duration distribution, and location of servers performing the transition. Our research on host-related information focuses on the development of approaches to host identification both in a static and dynamic network. We also extend the results of our previously described research on client identification in encrypted network traffic. We conduct several experiments to demonstrate, which information is possible to mine from encrypted network traffic.

The research areas mentioned above were chosen based on the challenges of CSA and IP flow monitoring presented in previous sections. We respond to cyber-related complexity (see pp. 24) and data-related value challenges (see pp. 26) by addressing the problem of host identification in network traffic. The IP flow open issues that we address in this chapter include, but are not limited to, analysis of encrypted network traffic (see pp. 37), changing paradigm (see pp. 48), and host-based view (see pp. 54). The changing paradigm open issue is addressed by research on network tunneling and IPv6, the lack of host-based view open issue is tackled by providing a possibility to identify a host in network traffic and inspecting a statistics commonly used for host behavior description.

This chapter is based on a collection of results published in following author's peer-reviewed publications: [A5–A10].

This chapter is structured as follows:

- *Section 5.1 presents our research on Top N statistics. The properties of the statistics are evaluated and the suitability for host identification is assessed.*
- *Section 5.2 investigates the IPv6 transition mechanisms. We perform several experiments to describe properties of tunneled and tunneling network traffic.*
- *Section 5.3 examines the capabilities of IP flow analysis to derive a host-related information. We focus on operating system detection and extend research on host identification in encrypted network traffic presented in the previous chapter.*

5.1 Top N Statistics

In this section, we research comprehension of information about individual hosts in a network. The information of a host in the network is widely used in many areas related to cyber situation awareness, such as network security, network accounting, and so forth. There exists a variety of both raw and derived statistics that can be gathered about individual hosts. However, the research lacks focus on the characteristics of the information provided by statistics and their properties. Therefore, we focus on the *Top N* statistics and try to describe the properties of information that can be obtained by *Top N* statistics.

Our research can be summarized into the following areas; *(i)* research of the characteristic of information provided by Top N statistics, and *(ii)* suitability of *Top N* statistics for identification of a host in network traffic. We provide a deep insight into the *Top N* statistics and present the available information that can be retrieved using this statistics on the real-world example. As a result, the information in this section should ease the decision whether *Top N* statistics is suitable for a proposed task which enables an operator to choose and focus on relevant data.

We describe in detail the *Top N* statistics itself and identify statistic's parameters and their influence on the statistics' outcome. We choose to evaluate the *Top N* statistics with regard to following perspectives: *availability*, *uniqueness of the information*, and *time stability*. These perspectives provide an insight into the characteristics of information provided by *Top N* characteristics. The *availability* is necessary to be able to obtain the results. The *time stability* represents the variability of provided information in time. The *uniqueness* represents the similarity between the *Top N* statistics of different hosts. After the detailed analysis of the statistics, *Top N* 's suitability for host identification is scrutinized. We discuss *Top N* each property and describe the implications of the host identification. The theoretical discussion is then validated on experiments based on the real world data from university campus network.

5.1.1 Statistics Description

"Find 3 IP addresses that transferred the most bytes during last 5 minutes" is a typical query for *Top N* statistics. The statistics is an internal part of tools for analyzing network data, such as already mentioned *nfdump*, *fbitdump* or *ntop*. Therefore, it is widely used for various applications in network traffic monitoring, such as identifying top talkers, providing an overview of the most important events in network traffic, port utilization statistics, or discovering popular network applications. Its results are used for optimizing network performance, identifying abnormal events [211], security monitoring, or for management reports [212]. In the following paragraphs, we will specify *Top N* statistics, scrutinize its parameters, and examine its computational requirements.

A full specification of a *Top N* query is the following:

$$\textit{Top N of X sorted by Y, over period of time P,} \tag{5.1}$$

where N is the number of output records of return characteristic X. The records of X are sorted descending by the variable Y and counted over a time period P. From the sorted list, the first N records are returned. A *Top N* query processing consists of four basic operations. First, data from a defined period P is selected. Second, the selected data is aggregated by the characteristic X, and aggregated values of characteristic Y are computed. Third, the aggregated records are sorted descending by variable Y. Lastly, the first N records from the sorted list are returned. All parameters have a significant effect on the results of the *Top N* statistic. We discuss each of them further in the paper.

The first parameters to determine are the return characteristic X, and sorting variable Y. The choice of both X and Y depends on the purpose for which the *Top N* statistic is used. The characteristic X defines the return of the statistics (e.g., an IP address is used as X for top talkers identification). Sorting of variable Y needs to be defined on a totally ordered set. A totally ordered set ensures that records of X can be sorted by Y. For example, the sorting variable Y can represent the number of transferred bytes, packets or flows, duration, and the number of occurrences. Derived characteristics can also be used, such as the average number of connections, maximum packet size or the number of distinct web pages visited.

The next parameter to set is the period P for *Top N* statistic computation. The period influences the amount of information which is processed and consequently determines the aggregation level of the *Top N* statistic. Short periods are chosen when detailed data is needed, whereas long periods are used for getting an overview. Since the period affects the amount of information processed and aggregated, it also affects the computational resources needed to compute the statistic. The longer the period is, the more information is processed, and the more computational resources are needed.

The last parameter to set is the number of returned records, N. This parameter plays the role of a cut-off. Only information which passes the cut-off, is presented. Therefore, the proper setting of this parameter is crucial. N depends on purpose *Top N* statistics are used for. When we want to identify the most active host in a network, N equal to one is sufficient. This is not the case when we want to create a report on port usage in a network. When N is set to low, only limited information is returned.

Let's consider the computational requirements of the *Top N* statistic. The statistic computation includes aggregation and sorting operations, which are computationally demanding. The aggregation process aggregates variable Y by return characteristic X. The aggregation process that covers a longer period or large scale network may result in the need to keep billions of records in memory. There are approaches to decrease the amount of memory needed. One approach leverages *map and reduce* technique [213], where partial *Top N* are computed in the map phase, and only results which pass a predefined threshold are passed to the reduce phase. Another approach leveraging the statistical properties of network traffic is presented in [211]. The aggregation process is succeeded by a sorting operation. The sorting operation adds significantly to the time complexity of *Top N* statistic computation. Depending on the choice of sorting algorithm, the comparison-based sorting algorithms cannot perform better than $O(n \log n)$.

5.1.2 Host Identification using Top N Statistics

A host can be identified in the network via its MAC/IP address. This identifiers are not however always available (MAC address) or reliable (IP addresses in dynamic addressed networks, NATs). Therefore, other approaches to host identification are developed. In general, these approaches are looking for a unique key, based on which a host could be identified. A key could be imprints of a host in observed data or can leverage a host's characteristics, e.g., ciphersuites, as shown in our work before [A6]. We will discuss the suitability of *Top N* statistics to generate such an identification key.

The parameters of *Top N* statistic are affected by the available data, which we use to compute the statistics. Given our data source, network traffic, we can use any information included in IP flow records. An overview of basic information that can be retrieved from network flows is provided in [214]. Next, we need to identify the return characteristic X and the sorting variable Y such that the computed *Top N* statistic can identify a host. We believe that a trace, that a host leaves in network traffic identifies the host in a network. *Top N* statistics based on X and Y need to be chosen such that it transforms the trace into as much of a unique statistics results as possible. Information from the link layer (L2) can be used only for LAN monitoring. Hence, we focused on L3 - L7 layer information. L3/4 layers of the OSI/ISO model provide information about communication partners, ports, and the protocol used. Since the number of distinct protocols that can be observed in network traffic is low compared to the number of entities, it is impossible to create enough variations of *Top N* statistic achieve uniqueness. This leaves us with information about communication partners and ports used. There are 65536 distinct ports in total, which allow us to create $\frac{65536!}{(65536-N)!}$ different variations of the *Top N* statistic. Assuming $N = 10$, this results in 1.46×10^{48} unique variations. However, the distribution of port utilization in a network is not uniform. A group of ports exists which are used more often than others (e.g., ports up to 1000). The amount of unique variations of actually used ports is then much lower. Nevertheless, a port seems to be a suitable return characteristic.

The destination IP address represents the communication partner. The number of distinct IP addresses is 2^{32} for IPv4 and 2^{128} for the IPv6 protocol, which ensures a high number of different *Top N* statistics. The set of a host's communication partners to identify is, however, much smaller than the theoretical maximum, therefore less distinct statistics exist. Still, the number of communication partners is much higher than the number of entities. Hence, we consider communication partner as a suitable return variable. The L7 layer is much more information-rich than the previously discussed layers. We can retrieve information from HTTP, DNS, SMTP, FTP protocols and many others. The information from the layers usually provides enough variability and a deep insight into the host's behavior. Therefore, the results of *Top N* are likely to be unique. Considering L7 layer as an information source is, however, limited by the increasing portion of encrypted network traffic. When network traffic is encrypted, only information from L3 and L4 can be used. Lastly, we need to investigate *Top N's* period P. The period affects the computational complexity of *Top N* statistic, and therefore also affect suitability for host identification. To be able to use the key for identification, the period needs to be reasonably short to be able to compute the *Top N* statistic. However, a short period P results in greater variability in *Top N* results, as the aggregation level of the statistic is fairly low.

5.1.3 Experimental Evaluation of Top N Statistics

In this section, we provide an experimental evaluation of *Top N* information value. We describe the dataset used for evaluation, evaluate the statistics with respect to *availability*, *uniqueness* and *time stability* and provide results of a host identification suitability experiment.

Dataset

The dataset contains network traffic captured from a university campus network. The dataset is divided into two subsets: training and testing. The training dataset is used for *Top N* characteristics evaluation and creating host's signatures. The signatures sets are then used on data from the testing dataset to assess the suitability for host identification. Table 5 provides a general description of the datasets. We choose to capture information about communication partners (destination IP), destination ports and the HTTP host information field. A host to detect is represented by the source IP address. To overcome the problems of IP address assignment, we chose only such networks where only static addressing is permitted and no proxies or NAT devices are present. The granularity of captured information is 5 minutes. Every 5 minutes, for each source IP address, we retrieve a set of communication partners with a given IP address (DstIP), a set of destination ports (DstPort) the address communicated to, and a set of web pages the address visited in the interval (HTTP_host).

Top N Properties Evaluation

We evaluate the *Top N* statistics properties using the following methodology. Firstly, we research a variability in data. Secondly, we inspect a choice of data used for computing the statistic and examine the information availability in the data. Thirdly, we compute *Top N* statistics and observe their behavior during the time to check the *time stability* requirement. Next, we compare the *Top N* statistic of each distinct IP addresses with each other to assess the *uniqueness* requirement. Lastly, we compute *Top N* statistics and evaluate their characteristics on the real-world dataset.

The variability of the information can be measured by the number of distinct values for each observed variable. There are 126 596 distinct communication partners, 47 516 distinct destination ports and 36 865 distinct HTTP_hosts visited in our training data sample. In the worst case scenario, we can still compute $\frac{39865!}{(39865-10)!} = 1.01 \times 10^{46}$ different keys (given $N = 10$). We counted the number of distinct values for each variable for every IP address to inspect the variability per IP address. The results are shown in Figure 30. The variability for a DstIP and HTTP_host is sufficient as more than 500 distinct values were observed at the majority of IP addresses. In the case of the DstPort variable, the majority of IP addresses are represented by less than 100 distinct ports. After further analysis, we discovered that 28.34 % of IP addresses used only 10 or less distinct destination ports. In conclusion, the variables DstIP and HTTP_host vary enough to provide sufficient information for generating a key. The variable DstPort provides only limited variability in the data. Therefore the generated key does not need to be unique.

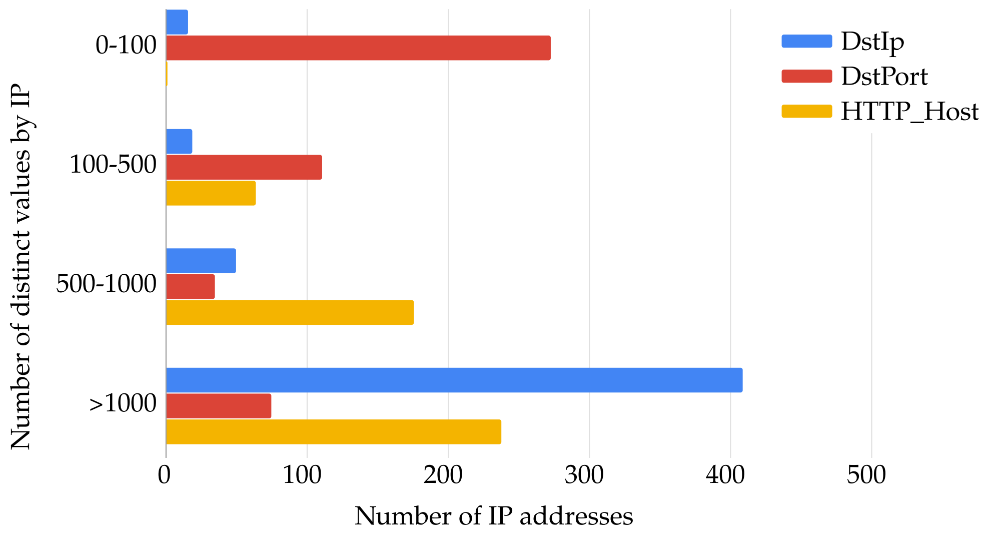


Figure 30: Distribution of variability per IP address.

Next, we investigated the availability of the statistics in time. For each source IP address, we computed *Top N* statistics with a different setting of period *P* and counted the non-empty results. Values of *P* were set to 5 minutes (the minimum value), one hour and one day. The results of the analysis are presented in Table 6. The table shows that a 5 minute period is not suitable for retrieving host information as there are only a few observations at the majority of IP addresses (25% of addresses are present in less than 288 observations from 2016 in total). The longer a

	Training DS	Testing DS
Observation Period	05 - 11/10/2015	19 - 25/10/2015
Unique IP Address	497	507
Total Flows	3 711 378	3 357 389
Total Bytes	36.6 GB	29.4 GB
Total Packets	236.4 M	228.6 M

Table 5: Datasets description.

period *P* for *Top N* computation is, the more IP addresses are observed in a higher portion of observations.

P = 5 minutes		P = 1 hour		P = 1 day	
# of observations	% of IP addresses	# of observations	% of IP addresses	# of observations	% of IP addresses
0-288	25.506	0-24	14.575	1	1.417
288-576	36.235	24-48	34.413	2	1.417
576-864	21.053	48-72	19.838	3	7.085
864-1152	11.741	72-96	20.648	4	15.992
1152-1440	2.429	96-120	6.478	5	19.231
1440-1728	1.417	120-144	1.417	6	15.789
1728-2016	1.417	144-168	2.632	7	36.032

Table 6: Statistics availability in time.

Thirdly, we investigated the stability of information provided by *Top N* statistic over time. For *P* equals one hour and one day, we computed a relevant number of *Top 10* statistics on the whole dataset (e.g., for $P = \textit{1 hour}$ we computed $7 * 24$ *Top N* statistics). Next, we counted a number of *Top N* statistics per IP address, in which the ten most frequent results of the *Top N* statistic were presented. Regarding the hour period, all of the ten most frequent results were observed in less than 42% of observations in 69% of IP addresses, which covers less than 15% of total observations. This indicates a higher variability in the data. The day period setting provides better results as the ten most frequent results of *Top N* statistics are observed more than in 57% of observations in 57.8% of IP addresses.

	P = 1 hour			P = 1 day		
	% of IP addresses					
Equal records	DstIP	DstPort	HTTP _host	DstIP	DstPort	HTTP _host
0 - 2	11.0	11.7	4.6	7.1	13.1	2.3
3 - 4	66.1	51.7	62.4	38.5	30.2	18.6
5 - 6	21.3	31.9	31.3	44.8	38.5	56.8
7 - 8	1.6	4.3	1.5	9.4	15.8	21.8
9 - 10	0.0	0.4	0.2	0.2	2.3	0.4
Jaccard	% of IP addresses					
0 - 0.2	45.2	2.0	28.4	22.3	4.0	6.6
0.2 - 0.4	51.3	5.5	66.4	61.3	25.8	56.8
0.4 - 0.6	3.3	27.0	5.0	15.6	36.7	33.9
0.6 - 0.8	0.2	33.7	0.2	0.8	23.5	2.8
0.8 - 1	0.0	31.7	0.0	0.0	10.0	0.0

Table 7: *Top N* time stability.

To capture the variability of *Top N* statistics for a particular host over time, we compared consecutive *Top N* statistics for each source IP address and counted the similarity of these *Top N* statistics. The higher the similarity is, the more stable the statistics are over time. We chose $N = 10$ for the comparison. We measured the similarity of *Top 10* statistics by the number of identical records and by their Jaccard index [215]. To provide an overview of the whole dataset, we computed the average values for each similarity measure per IP and showed a frequency histogram of the averages. The results are shown in Table 7.

	P = 1 hour			P = 1 day		
	% of statistics					
U(s)	DstIP	DstPort	HTTP _host	DstIP	DstPort	HTTP _host
0	34.5	2.6	16.3	51.9	0.6	28.9
1 - 9	31.3	3.4	25.3	33.9	2.8	44.2
10 - 99	34.0	21.4	51.0	14.2	15.0	26.4
>= 100	0.2	72.6	5.4	0.0	81.7	0.0

Table 8: *Top N* uniqueness.

We observed that the majority of IP addresses had 3-6 identical records in two consecutive *Top N* statistics. Regarding Jaccard similarity measure, DstPort characteristics showed more similarity than other characteristics for the one-hour interval. However, the similarity of the DstPort was low when an equal record count similarity was used. The divergence in the similarity measures is explained by the high number of IP addresses which use less than ten ports to communicate. The low number of ports decreases the similarity when using count of equal record similarity measure. However, the Jaccard similarity can handle this situation and provides unbiased results. HTTP_host performed well in both measures which prove the stability in users behavior. Generally, the similarity is higher in one day period than in one hour period.

The test for *uniqueness* requirement was also based on similarity. We used *Top 10* statistics to generate statistics for all IP addresses. The statistics results were then compared with each other. For comparison, we used the Jaccard similarity measure. We set a threshold to 0.25 and mark two results similar when the Jaccard was greater than or equal 0.25 (i.e., approx. 4 identical records in two *Top 10* statistics). The results of the experiment are presented in Table 8. DstPort characteristic did not meet *uniqueness* requirement as *Top N* statistics of the majority of the statistics were similar to more than 100 other ones for both periods. In general, period $P = 1$ day performed better as there were more unique statistics. The greater aggregation implies that more information is captured in the statistics than in the case the aggregation is low. Therefore the more aggregated the statistics is, the more likely it is unique. The DstIP provided most unique *Top N* statistics (51.9 % of statistics are unique). Hence, it should be the most suitable for host identification.

Top N Suitability for Host Identification

We computed *Top N* statistics for each host in the training dataset. The statistics were then applied to the testing dataset. The testing dataset consists of the same entities as the training dataset, which enables us to evaluate the results of the host identification process. Since a statistics consists of a number of records, a host was identified by a given statistics based on Jaccard similarity of *Top N* statistics. We set $N = 30$, period $T = 7\ days$ and period $P \in \{one\ hour, one\ day\}$ and Jaccard to 0.2 (approx. 10 equal records out of 30). The results are shown in Table 9.

True positive rate (TP) shows, how many of the hosts are correctly identified, i.e., the searched host is within the set of hosts identified by the statistics. False positive rate (FP) says how many hosts have been misclassified, i.e., the searched host is not within the set of identified hosts. We observed that the day period had a higher TP rate than the hour period. The highest TP rate was achieved by HTTP_host characteristic. In total, 59.5 % of the hosts from the testing dataset were successfully identified by the statistics based on this variable. The DstIP characteristic should be used when we prefer the precision of identification to identification rate as it had the lowest FP rate for both P. We also inspected different values for Jaccard for statistics' match. We

P	Variable	TP (%)	FP (%)	Not Found (%)
one hour	DstIP	3.04	0.61	96.36
	DstPort	34.01	21.86	44.13
	HTTP_host	8.35	2.09	89.56
one day	DstIP	20.45	7.89	71.66
	DstPort	44.13	25.91	29.96
	HTTP_host	59.50	15.66	24.84

Table 9: Experimental evaluation of *Top N* statistics. ($N = 30$, Jaccard = 0.2).

observed that with decreasing Jaccard, the TP rate increased and more hosts were identified as the similarity needed for the match was lower and more hosts were matched. The decrease of Jaccard also leads to higher FP rate as more hosts were mismatched due to the decreased level similarity of statistics.

We further evaluated cardinality of a set of identified hosts to determine the uniqueness of a statistics $U(s)$. For each key that correctly identified a host, we measured cardinality of the set of identified hosts. Table 10 shows the distribution of statistics with regard to statistics' uniqueness $U(s)^1$.

		% of hosts			
P	Variable	$\mathbf{U(s) = 1}$	$\mathbf{U(s) \leq 5}$	$\mathbf{U(s) \leq 10}$	$\mathbf{U(s) \leq 50}$
one hour	DstIP	86.67	100.00	-	-
	DstPort	1.19	9.52	13.69	24.40
	HTTP_host	85.00	100.00	-	-
one day	DstIP	77.23	93.07	96.04	100.00
	DstPort	4.59	10.55	18.35	39.91
	HTTP_host	36.49	72.98	85.61	100.00

Table 10: Experimetal Evaluation of *Top N* statistics uniqueness.

We observed, that in the one hour period, the uniqueness of the statistics was more significant, as the majority of the statistics was *unique* ($U(s) = 1$). The maximum cardinality of the set of identified hosts based on the DstIP and HTTP_host characteristics was 5. The statistics based on DstPort characteristic did not prove to be unique as the majority of the statistics identified more than 50 of hosts.

5.1.4 Summary

This section describes the information value of *Top N* statistics. We investigated the *availability* and *time stability* of the statistics, and evaluated *uniqueness* of its outcomes. The *Top N* statistics was then applied to the testing data, and the suitability for host identification was evaluated. We identified parameters of the *Top N* statistics and described their impact statistics outcome.

1. $U(s) = 1$ reads as statistics is similar to only one other statistics.

The experimental evaluation on real-world data showed that a period *P* correlates with *availability* and *time stability* of the statistics. The longer the period is, the more available and stable the statistics. The *uniqueness* has been highest for *Top N* of DstIP statistics and increased in longer periods. Moreover, we discovered that a single *Top N* statistic has a limited application on host identification problem. We were able to identify at maximum 60 % of hosts in the network traffic. However, the setting of Jaccard index threshold, which determined the equality of the statistics, was rather strict (two keys belonged to the same host when at least ten records out of 30 were equal). If we relaxed the setting, we would identify a higher portion of hosts (but it would also increase the FP rate). Nevertheless, once we were able to identify a host, the host was identified with high precision when we used the DstIP or HTTP_host characteristics (77.23% and 36.49% of the hosts were identified unambiguously).

The statistics identification capabilities could be enhanced by combining more types of *Top N* statistics. The host could be represented by both DstIP and HTTP_host statistics. This would increase the *uniqueness* of the compound statistics while preserving the *time stability* of the statistics. Moreover, we could use information from other L7 protocols for statistics (e.g., DNS protocol). Both improvements are left for future work. The host identification based on *Top N* statistic can be used for identifying a set of hosts which are similar to the searched host. Such identification can be used for law enforcement to identify a set of suspects for further investigation or in network security monitoring for identification of IoT devices in a network that need detailed surveillance.

5.2 Network Traffic Tunneling

Network traffic tunneling is a mean how to send a private communication over a public network using a tunneling protocol. The network traffic tunneling is used, for example, for connecting virtual networks in public clouds (Virtual Extensible Local Area Networks (VXLAN)), for secured remote access to a private network (OpenVPN), for creation of virtual point-to-point links (Generic Routing Encapsulation (GRE)) or for transferring the IPv6 network traffic over IPv4 network. We investigate the latter use-case in this section.

The understanding of the content of encapsulated network traffic enables better comprehension of the processes in a network, especially in complex cloud environments where encapsulation is widely used. This research reduces the complexity of the network (we provide insight into the encapsulated layer) and so responds to the complexity cyber-related challenge in CSA (see pp. 24). The analysis process also enables to cope IP flow monitoring issues related to the emerging trend of cloud services (see pps. 37, 48).

Despite IPv6 being the standard for several years, its adoption is still in process [216]. There are several ways of getting IPv6 connectivity, the dual-stack being the preferred one. Most IPv6 studies deal with native IPv6. However, there are other globally used options known as transition mechanisms. They can provide IPv6 connectivity on networks without native IPv6 connectivity enabled or without an IPv6 ready infrastructure.

The transition mechanisms tunnel IPv6 traffic through an IPv4 network. Despite being supported by major operating systems, there is a lack of studies investigating the characteristics of the tunneled IPv6 traffic. In this context, we investigate border traffic of the Czech national research and education network operator (CESNET) and attempt to reveal characteristics of IPv6 transition mechanisms, in terms of their usage, popularity, and impact on native IPv4 and IPv6. Apart from the challenges of CSA and IP flow monitoring, our research is mainly motivated by an exhaustion of the IPv4 address space and exerting pressure on network operators and content providers to deploy IPv6. The transition mechanisms are used to facilitate the IPv6 adoption. Unfortunately, they introduce extra elements in the network which add to the complexity and decrease performance and security. As a result, many existing methods for measuring and monitoring large-scale networks become ineffective.

The contribution of this section is threefold: *(i)* we provide an enhanced version of our flow-based IPv6 measurement system prototype, which enables IPv6 visibility in large-scale networks, *(ii)* we analyze and show IPv6 transition mechanisms traffic characteristics including a tunneled one and *(iii)* we show how the traffic of IPv6 transition mechanisms has evolved since 2010.

5.2.1 State-of-the-art

The most widely used and discussed tunneling transition mechanisms are Teredo and 6to4. Although several studies are focusing on performance evaluation of transition mechanisms, the characteristics of the traffic generated by the tunneling transition mechanisms are not well known.

A study by Aazam et al. [217] provides performance evaluation and comparison of Teredo and Intra-Site Automatic Tunnel Addressing Protocol (ISATAP) mechanisms with a focus on specific parameters like throughput, the end to end delay, round trip time and jitter. A study by Zander et al. [218] compares Teredo tunneling capability and performance with native IPv6 and 6to4 using measurements related to web services. Teredo increases the time needed to fetch web objects compared to IPv4 or native IPv6. The conclusion is that Teredo seems to be limited by a lack of Teredo infrastructure forcing encapsulated packets to travel long distances. Moreover, the throughput is partially limited by the performance of Teredo relay servers. A study by Bahaman et al. [219] discusses the performance of 6to4 with focus on communication over TCP. It states that the TCP transmission ability is reduced by the use of 6to4. However, it is still suitable for the early stages of the transition period.

Other related papers discuss the impact of transition tunnels on network security. Krishnan et al. [220] present security concerns with recommendations on how to minimize security exposure due to tunnels. It is pointed out that tunnels can have a negative impact on deep packet inspection and that transition mechanisms such as Teredo allow inbound access from the public Internet to a device through an opening created in a network address translation (NAT) device. This increased exposure can be used by attackers to attack a device hidden behind a NAT device effectively. A generally proposed security practice is to avoid the usage of tunnels at all and deploy other transition schemes like dual-stack.

Finally, Sarrar et al. [221] provide a brief insight into tunneled traffic in a study of the world IPv6 day impact on IPv6 traffic. The authors monitored Teredo and 6to4 transition mechanisms. The Teredo was discovered to carry mainly control traffic. The study also showed that IPv6 fragments were responsible for a significant portion of 6to4 traffic. The authors suspect that these fragments were caused by broken software which most likely forgot to take the IPv6 header size into account.

Investigated IPv6 Transition Mechanisms

The IPv6 traffic is usually divided between *native* traffic and *tunneled* traffic. The tunneled traffic is considered the one encapsulated using other protocols, e.g., UDP or IP protocol 41. This division is not necessarily accurate since the traffic that seems to be native IPv6 can, in fact, originate from a client using some transition mechanism like Teredo or 6to4. To clarify this point, we will differentiate between native IPv6 traffic, encapsulated tunnel traffic (IPv4 traffic containing IPv6 payload) and decapsulated tunnel traffic. The word tunnel might be omitted for the sake of brevity.

Teredo and 6to4 are the two most frequently used transition mechanisms in the CESNET network. Mechanisms like ISATAP, Anything in Anything (AYIYA) and others based on IP pro-

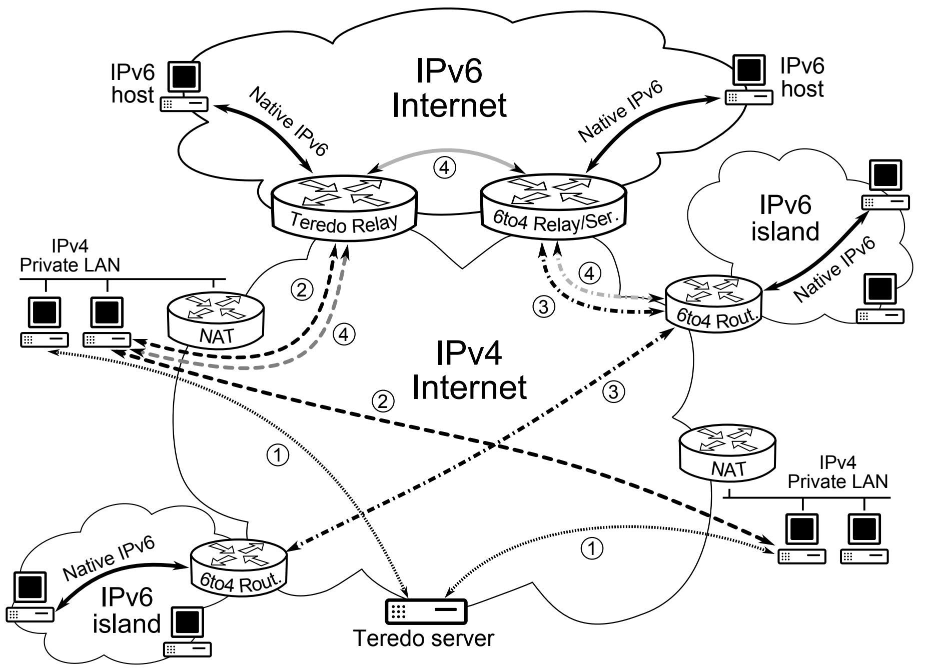


Figure 31: Teredo and 6to4 principles: ① Teredo start setup, ② Teredo traffic transiting over IPv4 network, ③ 6to4 traffic transiting over IPv4 network, ④ communication between Teredo and 6to4 endpoint.

tocol 41 (6in4, 6over4) do not contribute to the tunneled traffic significantly and do not appear in our analysis. Therefore, we will not describe them in detail. We did not analyze NAT64 and DNS64 mechanisms since they should appear as native IPv6 traffic on the outside.

Teredo [222] is designed to provide IPv6 connectivity to an endpoint behind a NAT device. It requires two network components for operation: *relays* and *servers*. Teredo servers are used for initialization of Teredo (Figure 31, communication ①), and after that for opening a port on the user's NAT device in case of a communication which is not initialized by the user. Relays are used for routing and bridging the IPv4 and IPv6 networks. Each Teredo endpoint uses a statically configured server and a relay, which can cause increased latency and low throughput in case of a distant server or relay. Teredo uses UDP for packet encapsulation making the traffic harder to identify.

6to4 [223] is only suitable for hosts with a public IPv4 address. It uses encapsulation in IP protocol 41 packets hence it is relatively easy to detect and monitor. The 6to4 relay servers are acting as a bridge between the IPv4 and IPv6 networks. These relays use any-cast prefix 192.88.99.0/24, therefore, the optimal (nearest) relay server should automatically be used for communication.

Figure 31 shows traffic between two endpoints (communication ④), one of which uses Teredo and the other one 6to4. The IPv6 traffic from Teredo client travels part of its path in Teredo tunnel to be later decapsulated on the edge of IPv6 Internet and shortly after that to be encapsulated again, this time by 6to4 to travel the rest of its path over the IPv4 Internet to the network of its destination. Depending on where the observation point is located, the tunnel (either Teredo or 6to4) or native IPv6 traffic can be observed.

5.2.2 Methodology and Measurement Setup

To perform a thorough inspection of tunneled traffic, we need to decapsulate packet headers of inner packets. We use the same IP flow-based framework as in [224] which we further modified [225] to extract more detailed information from tunneled data. The central part of the framework is a plug-in which replaces input and processing parts of current flow generator INVEA

FlowMon Exporter [176]. Every packet is being processed to extract basic flow statistics, and the processing of inner headers continues to the point when previously extracted fields indicate the absence of observed encapsulation types.

Teredo protocol is detected when IPv6 header is found encapsulated in UDP packet, AYIYA is searched for in packets on TCP or UDP port 5072. Other protocols are recognized by IPv6 address format, which is protocol specific and the 6to4 protocol can be additionally identified by usage of IPv4 anycast address belonging to 6to4 relay. If encapsulation is present, its type and encapsulated IPv6 header fields are used to extend the set of extracted fields and to identify individual IP flows taking place inside the tunnel.

Since we need to define new elements for IP flow records, Internet Protocol Flow Information Export (IPFIX) protocol is used. It allows using *Enterprise Elements* which can extend IP flow records with additional tunnel information. The framework can recognize and extract information from Teredo, AYIYA and other protocols that are based on IP protocol 41 such as ISATAP, 6to4 and 6over4. We provide the source code of the measuring tool under BSD license at the project web page [225].

Resulting IP flow records provide us with information about the encapsulated source and destination addresses, ports and transport protocol, which is a common five-tuple used to distinguish individual flows. We respect this principle and thus have separated the flows encapsulated in the same tunnel based on the value of these elements. Apart from these key elements the framework gives information about *Time to Live* (TTL), *encapsulated HOP limit*, *TCP flags* and *ICMPv6 type* and *code*, when present. Moreover, additional information about tunnel type is provided, including Teredo header and trailer types when present. The framework also newly supports geolocation using MaxMind [226] GeoIP database for both outer and encapsulated addresses.

The data are collected from several observation points located at the borders of CESNET network by passive probes; see Figure 32. All measured lines are at least 10 Gbit/s and together transport about 80,000 flows/s during work hours, which results in total traffic of 15.4 Gbit/s. We use IPFIXcol [227] framework to collect the extended IP flow records over TCP and to store them. New elements can, therefore, be defined and used without any further difficulties or limitations.

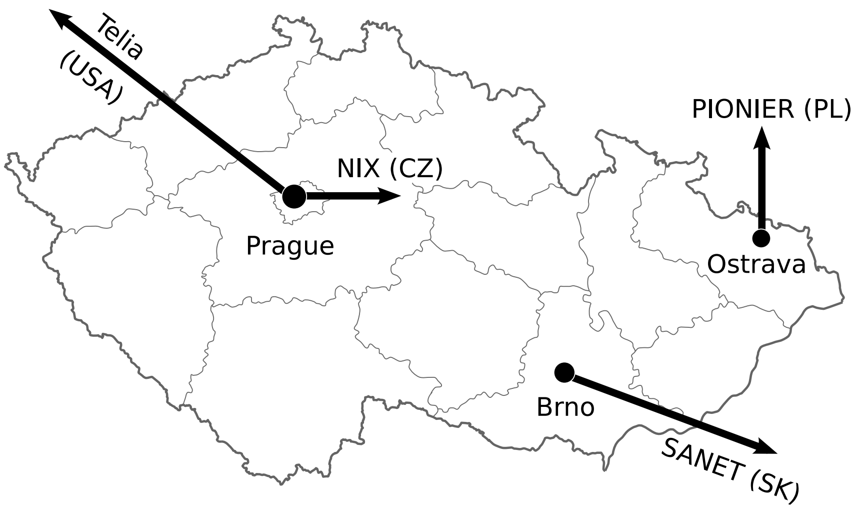


Figure 32: CESNET monitored links.

Flexible IPFIX collector is needed to handle the IP flow records containing information about IPv6 tunnels. We use IPFIXcol [227] framework to receive and store the extended IP flow records since new elements can be defined in configuration XML and used without any further difficulties. FastBit database was chosen as an IP flow record storage. The data from observation points are sent to the collector over TCP protocol, to ensure that no IP flow records are lost during transport.

The IPFIX data was collected over one week in January 2013 without the use of any sampling. Table 11 shows the average amount of traffic for all observation points. The total amount of stored data took approximately 2,485 GB of disk space. All statistics presented below are based on flow count.

Observation Point	Bits/s	Packets/s	Flows/s
Telia	1.65 G	274.9 k	22.1 k
NIX	7.17 G	1072.4 k	26.7 k
PIONIER	0.51 G	75.6 k	2.8 k
SANET	1.87 G	242.3 k	5.3 k

Table 11: Observation points IPFIX statistics.

5.2.3 Characteristics of IPv4 Tunnel Traffic

In this section, we describe characteristics of IPv4 traffic containing IPv6 payload. The analysis is based purely on information from IPv4 headers, and extended IP flow records are only used to identify relevant flows accurately. Three characteristics that can give us insight into tunneled traffic are addressed. Firstly, we describe the TTL values of the various traffic sets, and then we look into a geolocation aspect. Lastly, the basic flow statistics are presented.

Frequency of TTL and HOP Values

We study the distribution of *TTL values* of the observed IP flows. It is known that some operating systems use specific values, as shown in [228]. Microsoft Windows has the default TTL set to 128. The value of 64 is mostly used by Mac OS X and Linux devices, including devices running Android. We expect that these operating systems form a majority and are therefore the most significant. Figure 33 shows the most frequently used TTL values for IPv4 flows carrying an IPv6 payload. The TTL values are most frequent near the values set by OS vendors and the frequency is decreasing rapidly in less than ten hops. Therefore, we assume that most of the packets reach their destination in less than 32 hops. Thus we classify the flows according to their TTL numbers into four significant groups. The Windows traffic seems to be the most frequent one taking 60.3 % of the total, while Linux machines are not present so often with only 23.8 %. Apart from the Windows and Linux ranges, there are devices that set TTL to 255 and 32. Although the 255 are usually Cisco routers, in case of tunneled traffic we observed that the 6to4 traffic from anycast addresses have TTL set to 255 as well. The portion of the 255 range is 3.8 % and most of it originates from 6to4 relays. TTL numbers 24 and 26 are dominant in the group of values from 1 to 32, which makes 12.2 % of the total number of flows. We discovered that this is caused by a 6to4 tunnel that passes two observation points. The tunnel is heavily used and causes a large portion of tunneled traffic, which also affects other 6to4 measurements.

Overall, the TTL distribution of IPv4 traffic is different as shown in Figure 34. The Linux portion of the traffic is higher, and the TTL values of 32 and 255 are not as significant. A more detailed examination of the flow records shows that this is caused mainly by a high ratio of HTTP traffic. Even though more clients are using Windows operating system, most of the web servers are based on Linux, and therefore the responses have TTL less than 64. The DNS protocol shows similar characteristics except that the DNS traffic that we observe at our metering points

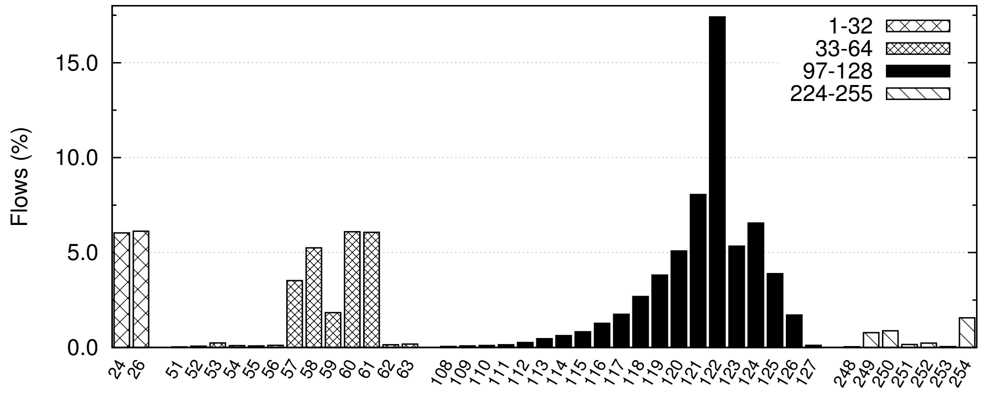


Figure 33: TTL value distribution of IPv4 traffic containing IPv6 payload.

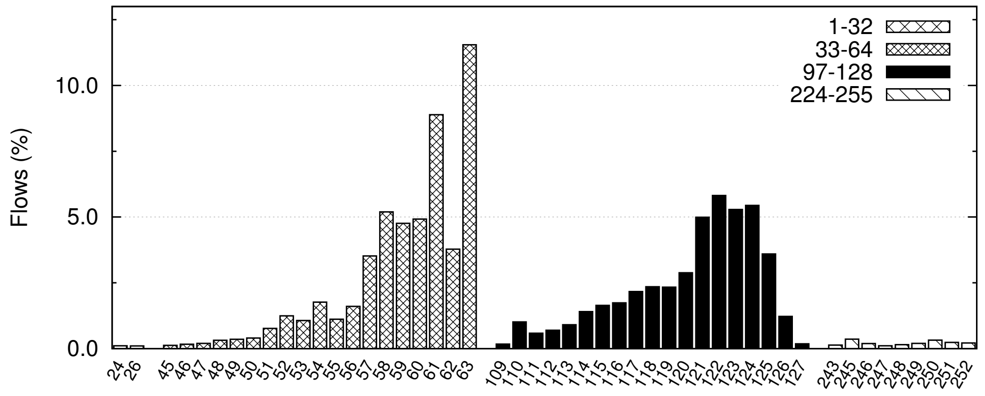


Figure 34: TTL value distribution of total observed IPv4 traffic.

is mostly generated by recursive domain name servers. Since the Linux DNS servers are the most widely used, they significantly contribute to the traffic generated by Linux machines.

IPv6 uses *HOP limit* instead of TTL. Figure 35 shows HOP limit distribution of native and decapsulated IPv6 traffic. Unlike IPv4, the HOP limit of 64 is the most frequent. We assume that Linux based machines use default HOP limit 64 and Windows machines use default HOP limit 128. This setting can be overridden by Stateless Address Autoconfiguration. Therefore, clients in managed networks (e.g., universities) might have the HOP limit set to a different value, regardless of their operating system. We verified this fact on several Linux- and Windows-based machines. Due to the significant share of HTTP(S) in IPv6 traffic, a large portion of Windows traffic is expected. Since the share of HOP limit 128 is negligible, we expect that common HOP limit in observed IPv6 networks is set to 64.

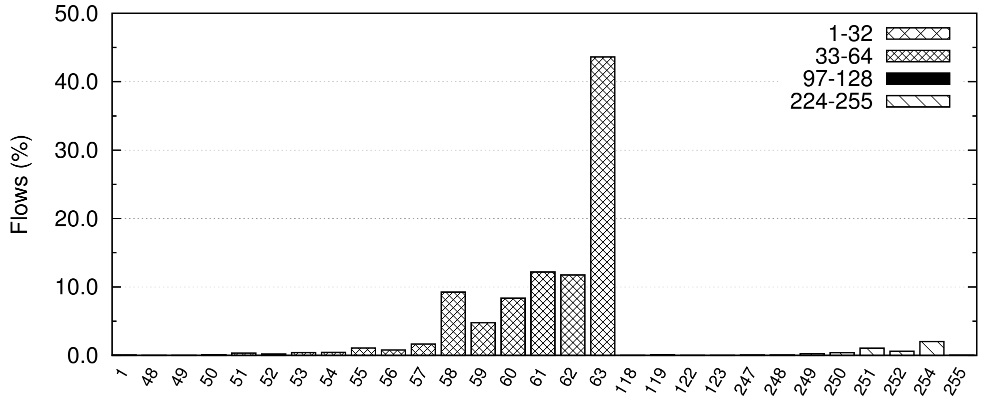
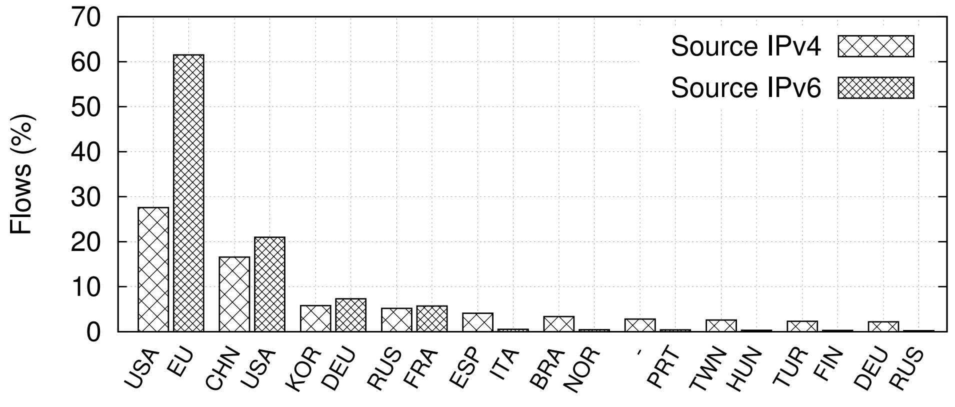


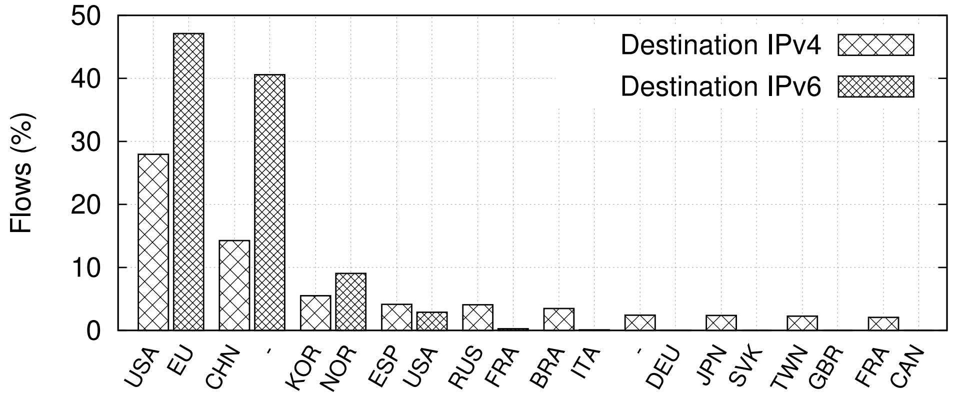
Figure 35: HOP value distribution of IPv6 traffic.

Location of IPv4 and IPv6 Endpoints

The second characteristic that we evaluate is *geolocation aspect* of the IPv4 and IPv6 traffic. We focus on data from Telia link only, which connects the CESNET network to the United States. This observation point highlights the differences in geolocation characteristics better. The statistics are computed separately for the incoming and outgoing lines and are shown in Figure 36. The IPv4 is more symmetric since the country statistics for both directions are similar. This behavior is reasonable since most of the requests initiate a response and the routes are also symmetric. The IPv6 have different properties. We discovered that the addresses that cannot be geolocated are mostly link-local addresses (fe80::/10) or local-link multicast addresses (ff02::/16). Such addresses should not be routed at all, which indicates that there are routers with erroneous IPv6 configuration. This misconfiguration also causes the asymmetry of the traffic, as such requests cannot be answered.



(a) Incoming traffic.



(b) Outgoing traffic.

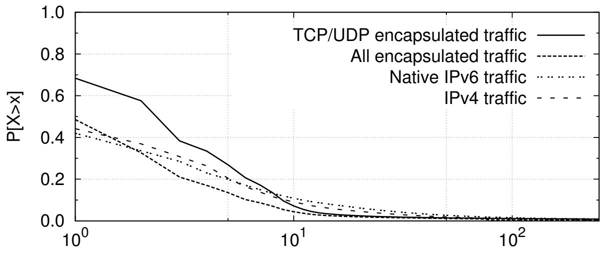
Figure 36: Top 10 country distribution for native IPv4, IPv6 and decapsulated IPv6 addresses.

Duration and Size of Flows

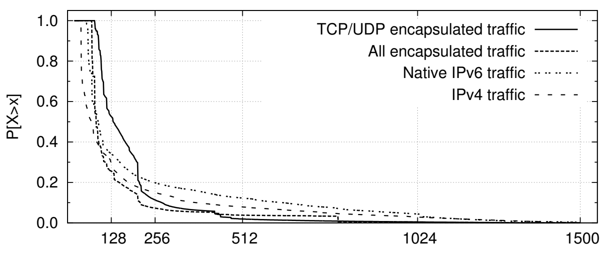
The third group of characteristics is represented by flow duration, a number of packets per flow and packet size (bytes per packet) statistics. For evaluation we employ *empirical complementary distribution function* (CCDF). We use the following formula to compute CCDF values:

$$\bar{F}(x) = P(X > x) = 1 - \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{x_i \leq x\} \tag{5.2}$$

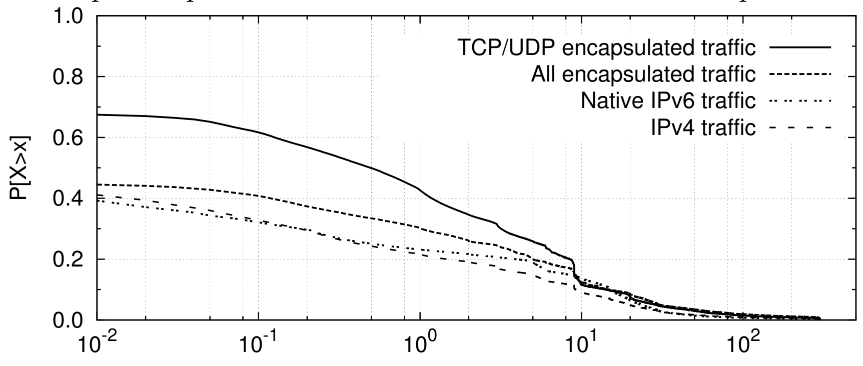
where $\mathbf{1}\{x_i \leq x\}$ is the indicator whether the event $\{x_i \leq x\}$ has occurred or not. The CCDF function describes how often the selected variable is above a particular level. From all



(a) CCDF of packets per IP flow.



(b) CCDF of packet size.



(c) CCDF of IP flow duration.

Figure 37: CCDF functions.

traces we filtered out four subsets of traffic: TCP or UDP encapsulated traffic (TCP/UDP), all encapsulated traffic (ALL), IPv6 native or decapsulated traffic (IPv6) and IPv4 traffic (IPv4). The subsets were chosen to compare the tunneled traffic with other common traffic types. Further, for each of the subsets and each of the characteristics, CCDF has been computed. Figures 37c, 37a, and 37b show the calculated CCDFs.

The majority of the IP flow duration of all subsets (Figure 37c) accounts for durations shorter than 10 seconds. The IP flow durations longer than 10 seconds represent only 11 % or less. The TCP/UDP contains much fewer short duration IP flows (TCP/UDP: 32.16 % $\leq$ 0.01 sec.; ALL: 54.66 %; IPv4: 59.84 %) which is explained by the absence of connection-maintaining IP flows necessary for other subsets. We can observe slightly increased frequencies of the IP flow duration between 7 and 11 seconds especially at TCP/UDP and ALL subsets. Hence, the tunneled traffic generally contains fewer short duration IP flows than IPv4 or IPv6 traffic.

The packets per flow distribution (Figure 37a) suggests that single packet IP flows form only 31.6 % in the case of TCP/UDP, 51.42 % in the case of ALL, 57.99 % and 55.82 % in other cases. We expected tunneled traffic to behave similarly as IPv6 traffic. Nevertheless, we can observe a vertical shift between ALL and IPv6 CCDF. This shift is caused by single packet IP flows, and it is caused by DNS traffic to root DNS servers, which uses IPv6 protocol. The slope of CCDF for TCP/UDP and ALL is higher than the slope for other subsets, which states higher frequencies of specific packet counts. In conclusion, the distribution of the packet counts of the tunneled traffic is slightly different from the distribution of the IPv4 and IPv6 traffic.

The last characteristic described by CCDF is packet size (Figure 37b). In the case of encapsulated traffic, we consider the outer packet size including the encapsulation header. The earlier study of the packet size distribution mentions a significant difference between the CCDFs of packet sizes. The authors of [229] state that the distribution of IPv4 traffic fits a heavy-tailed distribution, whereas IPv6 traffic does not. We expect the tunneled traffic to have similar characteristics as IPv6 traffic, and thus the CCDFs are expected to be similar, too. The Figure 37b shows some discrepancies in this hypothesis. The packets larger than 400 bytes in the tunneled traffic represent only 5 %, while in the IPv6 the portion is still 15 %. Furthermore, packets smaller than

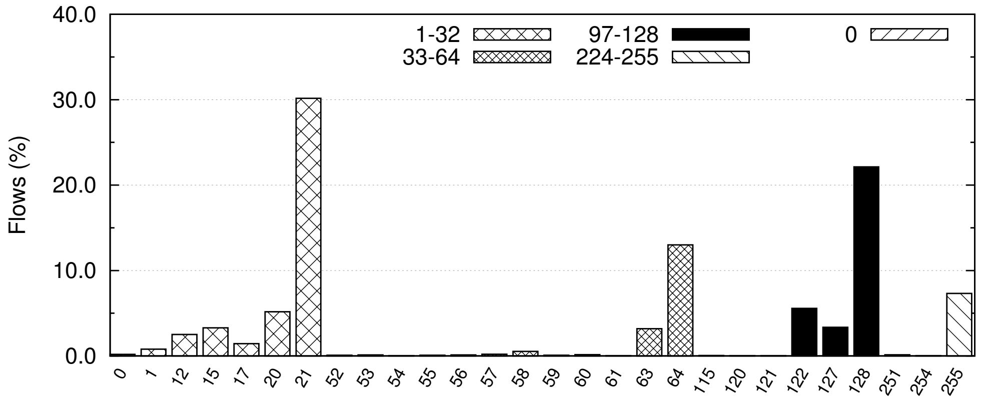


Figure 38: HOP value distribution of encapsulated IPv6 traffic.

70 bytes are not present in the tunneled traffic, although they account for 26 % of IPv6. This distribution is caused by a shift of graph to a higher packet size given by the encapsulation. The IPv4 header is usually 20 bytes long, and in case of Teredo, there are another 8 bytes for the UDP header. Even taking this shift into account there is still a noticeable drop at the size of 200 bytes which is caused by a high share of ICMPv6 and BitTorrent control traffic.

5.2.4 Characteristics of IPv6 Tunneled Traffic

In this section, we focus on characteristics of encapsulated IPv6 traffic for which we use the same data set as in Section 5.2.3. We show HOP limit statistics, detected Teredo servers and geolocation characteristics. We discuss the most used TCP and UDP ports inside the tunnels.

Distribution of HOP Limits

Figure 38 shows HOP limit distribution for the encapsulated IPv6 traffic. The main difference from the TTL (Figure 33) and HOP limit (Figure 35) statistics is that the values here are distributed with much less entropy. The limits 21, 64, 128 and 255 are achieved and also the most frequent ones. This difference is caused by the fact that most of the traffic never traversed the IPv6 network and the HOP limit was therefore never decreased. In fact, when the values are lower, we can be reasonably certain that the packets already traversed the IPv6 network and are heading towards the IPv4 destination. The value 21 is used for Teredo bubbles by Windows 7 with Service Pack 1 and earlier.

The Teredo bubbles are used as a special mechanism for NAT traversal, which is consistent with the fact that most of the clients are behind a NAT. We can see that some packets reach the value of the zero HOP limit, which is a known problem when the HOP limit is set as low as to 21. The value of 255 is used for IPv6 neighbor discovery messages so that when a host receives such packet with HOP limit lower than 255, the packet is considered invalid [230].

Teredo Servers

There are two ways of detecting Teredo servers. Firstly, we can look at the traffic using UDP protocol on port 3544, which is a well-known Teredo port, and select the addresses that communicate most often. The shortcoming of this approach is that some other services might be using the Teredo port and therefore the results might not be accurate. Since we can decapsulate Teredo traffic, we can derive IPv4 addresses of Teredo servers directly from Teredo IPv6 addresses. This way we can even detect Teredo servers that are not communicating directly through our observation points. Table 12a shows Top 10 servers that were discovered in the encapsulated IPv6 addresses.

Server IP	Ratio	Owner	Ctry
65.55.158.118	28.33 %	Microsoft	US
94.245.121.253	27.98 %	Microsoft	GB
157.56.149.60	26.49 %	Microsoft	US
157.56.106.184	10.18 %	Microsoft	US
94.245.115.184	6.41 %	Microsoft	GB
83.170.6.76	0.04 %	B. Schmidt	DE
170.252.100.131	0.01 %	Accenture	US
94.245.127.72	0.01 %	Microsoft	GB
94.245.121.251	0.01 %	Microsoft	GB
217.31.202.10	0.01 %	CZ.NIC	CZ

(a) Based on Teredo IPv6 addresses.

Server IP	Ratio	Owner	Ctry
94.245.121.253	43.24 %	Microsoft	GB
65.55.158.118	18.91 %	Microsoft	US
157.56.149.60	17.86 %	Microsoft	US
94.245.115.184	10.00 %	Microsoft	GB
157.56.106.184	6.50 %	Microsoft	US
94.245.121.254	0.72 %	Microsoft	GB
94.245.115.185	0.22 %	Microsoft	GB
65.55.158.119	0.18 %	Microsoft	US
83.170.6.76	0.18 %	B. Schmidt	DE
157.56.149.61	0.17 %	Microsoft	US

(b) Based on UDP port 3544.

Table 12: Top 10 discovered Teredo servers.

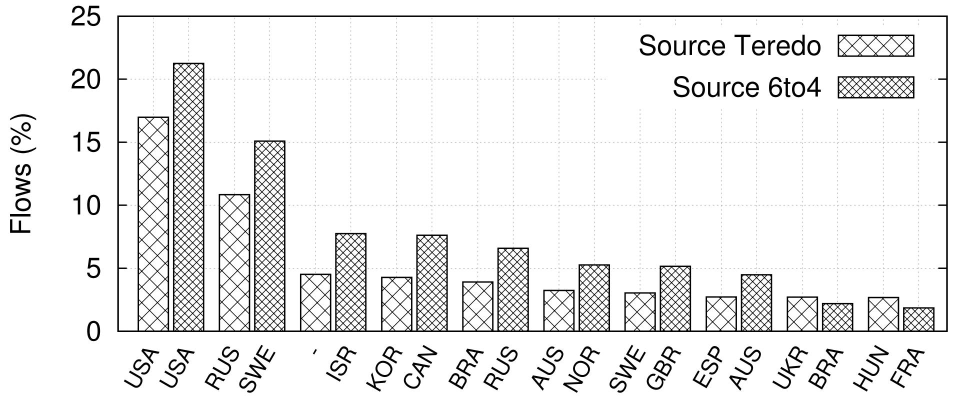
Using the WHOIS database, we confirmed that a majority of servers is operated by Microsoft, which is only to be expected since Teredo is a Microsoft technology. The most of Microsoft Teredo servers we identified are IP addresses of "teredo.ipv6.microsoft.com", which is the default Teredo server name configured under Windows. The address 83.170.6.76 has a hostname indicating that it serves as a Miredo server (Teredo implementation for Linux and BSD). The last address belongs to CZ.NIC (Czech top-level domain operator), which is known to promote IPv6 deployment in the Czech Republic and operates local Teredo and 6to4 servers.

Table 12b shows Teredo servers discovered at the most active on Teredo port 3544. This way we detect only Teredo servers that are establishing connections through our observation points. We can see that most users use Teredo servers in the United States or Great Britain to get IPv6 connectivity. Such a considerable distance of Teredo servers is known to increase the latency of such connections, and therefore we would recommend using local servers to Czech users, such as the CZ.NIC servers.

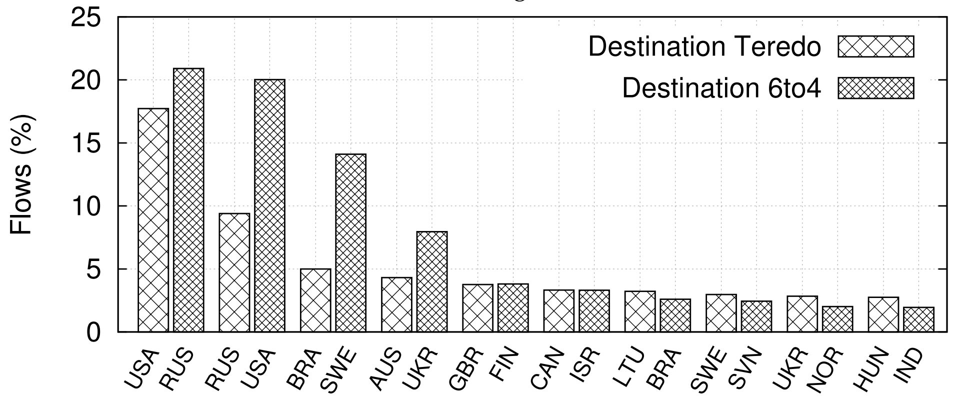
Location of Tunnel Endpoints

The geolocation statistics of tunneled traffic are computed for Teredo and 6to4. We use encapsulated IPv6 addresses to determine the countries for each flow. Incoming and outgoing traffic is taken separately just as in Figure 36. The statistics are shown in Figure 39. The tunneled traffic shows very different geolocation characteristics compared to native and decapsulated IPv6 traffic even though both are from the same link. Most of the native and decapsulated IPv6 communication takes place inside the EU, while large portion of tunneled traffic communication is performed with the USA and Russia.

To identify applications that are using IPv6 connection provided by transition mechanisms, we created a list of the most used encapsulated TCP and UDP ports. We observed several ports that can be found both in the source and destination port Top 10. The source and destination ports Top 10 represent 32.0 % and 40.5 % of the traffic respectively. The well-known ports are - *HTTP - 80* (0.85 % of traffic as source port, 5.61 % as destination port), *HTTPS - 443* (0.58 % and 1.48 %) and *DNS - 53* (1.49 % and 1.48 %). Among the most frequent ports are ports 49001 (15.96 % and 9.91 %) and 51413 (10.12 % and 16.07 %) which are used by BitTorrent clients (namely Vuze and Transmission). We discovered that these ports are heavily used within the 6to4 tunnel as mentioned in Section 5.2.3.



(a) Incoming traffic.



(b) Outgoing traffic.

Figure 39: Top 10 country distribution for encapsulated IPv6 addresses.

5.2.5 Evaluation of IPv6 Adoption

In this section, we describe the deployment of the IPv6 protocol with respect to tunneled traffic. The overall statistics of IPv6 and tunneled traffic are mentioned. We provide a historical comparison to our previous measurement [224].

The network activity shows the correlation with human activity. Both IPv6 and tunneled traffic are considerably smaller during the weekend than during weekdays. As for the IPv6 traffic, the increase in traffic volume starts at 6 AM, reaches the peak around 11 AM and holds the high level till 4 PM. Then the traffic steadily decreases and reaches the minimum at 3 AM the next day. The tunneled traffic shows a slow increase that starts at 6 AM and peaks around 6 PM. The decrease begins at 10 PM and reaches the minimum at 5 AM the next day. The possible cause of this shift from the IPv6 diurnal pattern is the fact that the tunneled traffic is widely made by BitTorrent clients.

We measured the tunneled traffic back in 2010 on three CESNET border links to SANET, PIONIER, and NIX. We found that the tunneled IPv6 was responsible for 1.5 % of total flows, which is the same share as we measured today. However, the relative amount of bytes transferred has almost doubled from 0.66 % to 1.28 % of total bytes today. The share of the native and decapsulated IPv6 was only 0.10 % (0.21 % of bytes) compared to 3.39 % (4.42 % of bytes) today. The known services by port (HTTP, HTTPS, and DNS) had a share of less than 1 % of total flows. Today's share of these services is significantly higher (see Section 5.2.4). The measurements show that the overall usage of both tunneling IPv6 transition mechanisms and native IPv6 has been raising.

We distinguish between the encapsulated and decapsulated tunneled traffic, as mentioned in Subsection 5.2.1. The decapsulated tunneled traffic is included in the measured IPv6 traffic.

When we filter out the decapsulated Teredo and 6to4 traffic, they account together for 5.91 % of the measured IPv6 traffic. Teredo traffic takes part of 83.13 % of the decapsulated tunneled traffic and 6to4 16.87 %. Hence we should not consider all measured IPv6 traffic as native IPv6 traffic. The main contributor to tunneled traffic was Teredo with an occurrence of nearly 89 % followed by 6to4 with over 11 %. Therefore the relative amount of 6to4 traffic has increased. We then detected the use of 13 Teredo servers compared to 53 today.

5.2.6 Summary

In this section, we have taken a detailed look at the IPv6 transition mechanisms. We have provided an improved version of our tool for investigating IPv6 tunneled traffic. Considerable progress has been made concerning understanding tunneled traffic behavior, especially concerning Teredo and 6to4 traffic. The results of this section suggest that encapsulated traffic differs from IPv4 and IPv6 in several characteristics including TTL values, geolocation aspect, and flow duration. Moreover, we have provided the list of Teredo servers and described the evolution of IPv6 adoption.

This section is the first step towards enhancing our understanding of encapsulated IPv6 traffic. We hope that our findings will be beneficial as a background for additional research into IPv6 transition mechanisms. To further our research we are planning to carry out an in-depth analysis of tunneled IPv6 traffic concerning the security matter. To be able to handle security incidents, the security threats will be identified, and detection methods will be developed. Since our results are encouraging, they should be validated on other large-scale networks. In a broader level, research is also needed to evaluate the contribution of the IPv6 transition mechanisms to the IPv6 adoption.

5.3 Host-related Information

In this section, we perform several experiments that demonstrate the capabilities of IP flow analysis to derive host-related information from network traffic. The host-based information discussed in this section is mainly used for a host identification in a network. The host identification is a necessary prerequisite for understanding the complex network (see CSA complexity challenge pp. 24) and creation of host-based view (see open issues of IP flow analysis pp. 54).

Firstly, we focus on the identification of a host operating system (OS) both in static and wireless networks. Being aware of all OS of hosts present in a network brings an advantage for operators protecting network security. We provide following use-cases where security managers can leverage OS fingerprinting to demonstrate its benefits:

- *Unsupported OS* – rapid development of OS versions lead to many users still using an outdated version without security updates. This problem is obvious especially in mobile OS where nine % of Android users have a version older than 4.4 [231].

- *Decision making* – knowledge of what devices are connected helps security administrators to take the right decision and react to incidents more precisely to security incidents [232].

- *BYOD security policy* – Bring Your Own Device is the principle of dynamic networks, but some networks have policy restrictions on which devices can users bring or on using only specific systems.

- *Static networks* – any change of device behind IP address or disallowed OS in the segment can indicate a rogue device and should be investigated. Methods designed for dynamic networks are implicitly capable of such detection.

Secondly, we focus on host-related information, that can be derived for encrypted network traffic. This research responds to the current uprising trend of network traffic encryption (see pp. 37). We extend our research on identification of client in HTTPS traffic described in Section 4.2 by the description of the information available from a dictionary and demonstration of its applications in the classification of network traffic, network security and forensics, and browser fingerprinting.

5.3.1 State-of-the-art

Passive OS fingerprinting is a transparent method for OS identification. Each OS has specific settings which leave a fingerprint in packets sent by the system. The passive OS identification analyses packets originating from a system. Based on the fingerprint found in a packet, a particular OS is identified. The majority of information for an OS fingerprint is collected from TCP/IP packet headers and application protocols. An approach using TCP/IP packet header is well investigated. Lipmann et al. [233] present classifiers capable of identifying nine classes of OS from packet headers. The classifiers leverage machine learning techniques including cross-validation testing. Tyagi et al. [234] employ SYN packet properties such as TTL, packet length, TCP window size setting, and TCP options. Their classifier based on Euclidian distance can correctly identify 95.5 % of operating systems from nearly 2000 SYN packets. From application protocols, information such as HTTP User-agent, HTTP domain, or DNS queries can be used. Authors of [198] utilize characteristics of DNS queries specific to individual OS, e.g., unique domain names, query patterns, and time intervals. Machine learning (ML) techniques are used to generate new OS features and signatures. Aksoy et al. [235] employ machine learning algorithms to identify packet features suitable for OS detection. They use genetic algorithms for feature subset selection in three ML algorithms. The combination of automatic feature selection and ML algorithms enables the adaptive OS detection and reduces the number of packet features needed for OS classification. The limitations of automatic OS fingerprint are discussed by Richardson in [236] and they identify four major challenges for automatic OS detection. The first challenge lies the inability of current tools to find a generalizable and sufficiently discriminative classification rules for different OS versions. Secondly, fully automatic tools cannot easily exploit semantic knowledge of protocols or generate multi-packet probes and attributes. The remaining challenges are the ineffectiveness of the tools in real networks and overfitting of the detection techniques.

synSize	winSize	TTL	OS
52	8192	128	Windows 10.0
52	8192	128	Windows 6.1
52	65535	128	Windows 10.0
60	65535	64	Android 6.0
60	14600	64	Android 4.4
60	29200	64	Ubuntu
64	65535	64	Mac OS X 10.12
64	65535	64	iOS 10.3

Table 13: OS specific TCP/IP header values.

We differentiate four main approaches to IP flow OS detection:

- **TCP/IP Parameters** – Each OS uses different settings for certain fields in TCP/IP headers. The OS specifics fields are initial SYN packet size (synSize), TCP window size (winSize), and Time to Live (TTL). The example of TCP/IP fingerprints with associated OS are presented in Table 13. The identification using TCP/IP parameters is not unambiguous as more OSs, typically belonging to one OS family, share the same field values, as can be observed in the table.

- **HTTP User-Agent** – HTTP User-agent is defined in [237] as a string from user originating the HTTP request to provide information about operating system and browser to the server. The purpose of such identification is that a web server can serve content customized for a specific device or software and increase the user experience while browsing web pages. The User-agent string construction is fully under control of application software independent of the underlying operating system, as User-agent belongs to the application layer of network communication. However, in practice, it is common that applications fill in the operating system name with its major and minor version and often even with the specific build of that version.
- **Specific Domains** – Modern operating systems are configured to do many specific actions upon connecting to a network. These actions include connectivity testing (e.g., *connectivitycheck.android.com*) and checking for system updates (e.g., *update.microsoft.com*). This activity can be monitored on the level of DNS communication during name resolving, or as the communication to external servers. Observing connections to these specifics domains in IP flow records enables us to identify a host's OS system.
- **Hybrid Approach** – The methods mentioned above can be combined to achieve improved accuracy of OS detection. We implemented a combination method that benefits from the advantages of the individual methods and is not limited to any specific network layer. We treat the methods equally, and the final decision is based on the majority voting principle. In the case when each method has different results or only two methods can identify OS, and they disagree, the results are taken in the order of User-agent, Specific domains, and TCP/IP parameters. We have decided for this order because TCP/IP parameters are often the same for multiple operating systems and the decision is based on the highest probability to be correct. User-agents are then preferred over specific domains as it is a long-established method.

Each method has a different level of details that it could provide about the OS. For their comparison later in this work, we need to set a common level based on the OS hierarchy. The level of detail each method can provide is summarized in Table 14. TCP/IP method can provide major and minor version information, but parameters are often the same for multiple versions and can lead to overfitting of identification. Hence, the levels could be omitted and are marked in brackets.

Method	**Vendor**	**OS name**	**Major version**	**Minor version**
User-agent	✓	✓	✓	✓
TCP/IP parameters	✓	✓	(✓)	(✓)
Specific domains	✓	✓	×	×
Combination	✓	✓	✓	✓

Table 14: OS identification methods level of detail.

5.3.2 OS Fingerprint in Static Networks

To demonstrate the detection capabilities of the IP flow records, we deployed a commercial detection tool Flowmon [238] in a university campus network. We use commercial Flowmon probes to create IP flow records since these probes have built-in capabilities for User-agent parsing. When a packet with User-agent is captured, it is compared to probe's User-agent database and OS name, major and minor version, and build values are assigned to the whole flow. When an unknown pattern of User-agent is encountered, the flow is treated as no User-agent was present at all. The TCP/IP stack information is collected by the probe by default.

First, we prepare a database which maps specific parameter values to the operating system. Our approach to creating such a fingerprint database is similar to the one suggested by Matousek et al. [239] which maps information gained from HTTP User-agents to TCP/IP parameters. We have processed IP flow records captured from the whole university network during two months in 2017 to cover as many different devices as possible. We aggregated the IP flow records into groups with the same triple (synSize, winSize, TTL), and finally assigned the corresponding OS to each triple. Following this process, we have created a database of 2078 unique mappings from TCP/IP parameters to 51 unique operating systems and their major and minor versions (when available) weighted by their appearance (i.e., number of flows with the same triple) in our network. This weighting called *confidence* is necessary to deal with different operating systems or their versions that send packets with the same triple of TCP parameters. We compute confidence as the fraction of the number of flows of a specific OS version compared to the total number of flows with the same triple. Example of our fingerprint database based on TCP/IP parameters including confidence listed in Table 15.

synSize	**winSize**	**TTL**	**OS**	**Confidence**
52	8192	128	Windows 10.0	55.2 %
52	8192	128	Windows 6.1	31.9 %
52	65535	128	Windows 10.0	74.9 %
60	65535	64	Android 6.0	48.2 %
60	14600	64	Android 4.4	28.4 %
60	29200	64	Ubuntu	20.4 %
64	65535	64	Mac OS X 10.12	26.5 %
64	65535	64	iOS 10.3	10.3 %

Table 15: Example fingerprint database.

To validate our measurement, we compare our fingerprint database to databases of *p0f* and *Ettercap*. Surprisingly, neither of them uses the size of the initial SYN packet, and the comparison is limited to only two parameters. The characteristics of the main operating systems (Windows, Mac OS X) are the same. However, the *p0f* and *Ettercap* DBs generally lack updates (the last update of fingerprints on GitHub was 21 May 2014 for *p0f* and respectively 26 Oct 2011 for *Ettercap*). Because of this outdated fingerprint DBs, new systems like Windows 10 or Android 4.4 and higher, which currently dominate the network traffic, are not included and hence not recognized by the tools. Creating a complete, up-to-date database of fingerprints is a challenging task. Our goal is to create a database containing fingerprints of as many different systems as possible. We decided not to use any strictly managed environment as there would be a lack of desired diversity. Instead, we focused on processing data from User-agents which can be done in large scale and covers most currently used systems. Our fingerprint database contains confidence rating calculated from the large volume of network traffic. The computed confidence rating allows us to deterministically identify OS of an IP flow even if the DB contains more records for the same triple. However, this approach results in identification biased by current market share of each OS within our university population where an uncommon OS can be overshadowed by a popular one. We identify this fact as a general limit of TCP/IP parameters identification method because every fingerprint database must deal with parameter collisions.

Next, we evaluated the uniqueness of the OS identification from IP flow record, i.e., how many different OSs is assigned to one host. During the two hours measurement period assigned for this experiment, we observed 10.221M flows from 12 897 hosts in the campus network. 33.5 % (3.425M) of all monitored flows contained all the information needed for OS detection which represented 70.33 % (9 072) of all hosts. We observed that in some cases more than one OS was detected for one IP address. The cause of this behavior could be dynamic addressing in networks. Therefore we removed all dynamically addressed subnets from evaluation.

The results (see Table 16) show that the portion of the IP addresses with more than one detected OS has decreased after the removal of dynamically addressed networks. However, still 4 % of IP shows characteristics of two or more OS. This fact can be explained by the presence of more devices with different OS using the same IP address. The presence of more OS with the same IP address implies the presence of Network Address Translation (NAT) devices. Therefore, the OS detection can also be used as NAT detection assuming that only a static addressed network is monitored.

# of unique OS	**# of IP in A**	**% of all A**	**# of IP in B**	**% of all B**
1	7898	87.059	3996	95.989
2	1071	11.806	159	3.819
3	80	0.882	7	0.168
> 3	23	0.253	1	0.024
Total	9072	100	4163	100

A - whole network, **B** - dynamically addressed subnets removed

Table 16: Number of unique OS detected at one IP.

5.3.3 OS Fingerprint in Dynamic Networks

The previous experiment showed the negative impact of dynamic address allocation on the uniqueness of OS. Therefore, we further investigate the possibilities of OS identification from IP flow records in a dynamically addressed network in this section. We conduct an experiment to demonstrate the coverage of IP flow based OS detection methods, and evaluate them concerning accuracy, precision, and recall measures. The dataset used in this experiment covers data from all subnets of the wireless network (Eduroam) of our university, including buildings of multiple faculties and dormitories. It contains data from the first week in May 2017 and is a combination of three data sources – IP flow records, Radius logs, and DHCP logs.

In total, our dataset covers

- 79 087 345 flows in IPFIX format,
- activity of 21 746 unique users,
- 253 374 Wi-Fi sessions,
- 25 642 unique MAC addresses (1692 vendor prefixes),
- 6 104 unique IP addresses assigned.

IP flow records represent our primary source of information and contain extension fields used for OS identification described above (i.e., HTTP User-Agent field, TCP/IP stack information). Our IP flow observation points are located at the backbone network connecting the university to the Internet. Hence, our visibility covers the communication from and to university. As we are interested in dynamic networks, we have filtered the traffic so that it contains only traffic originating from (i.e., source IP address is from) our wireless subnets. The opposite direction IP flows (target IP in our subnet) are not relevant for OS identification of a host in the monitored network and were omitted.

Logs from DHCP servers then enrich the sessions by device MAC address and device name. As the network is dynamic and we have no control over connected devices, we have to derive the *ground truth* for OS identification from these logs. A large portion of devices comes with a pre-installed operating system with default device name. In many cases, it is hard or impossible for a common user to change the device name. For example, Android devices use string "android-<android_id>", Apple products use "<user>-iPhone", "<user>-iPad", and Microsoft uses default name "Windows-Phone" for its mobile products.

All connections to our wireless network must be authenticated with username and password. We have collected logs from Radius servers, which provide the authentication service. From these logs, sessions are created as 4-tuple (*id, assignedIP, startTime, endTime*), where id is a simple auto-incrementing counter. Sessions then serve as ground truth to measure methods coverage. We have removed the user identity from dataset due to privacy reasons.

We implemented the OS detection methods described before and used them to identify OS of each Wi-Fi session from our dataset. The identification does not work on a single flow but takes every flow from the session with OS identified into account. For a session containing more flows, OS is assigned to each flow, and the final result for the session is decided based on majority voting principle. This principle is used because some devices exhibit fingerprints of multiple OS during one session.

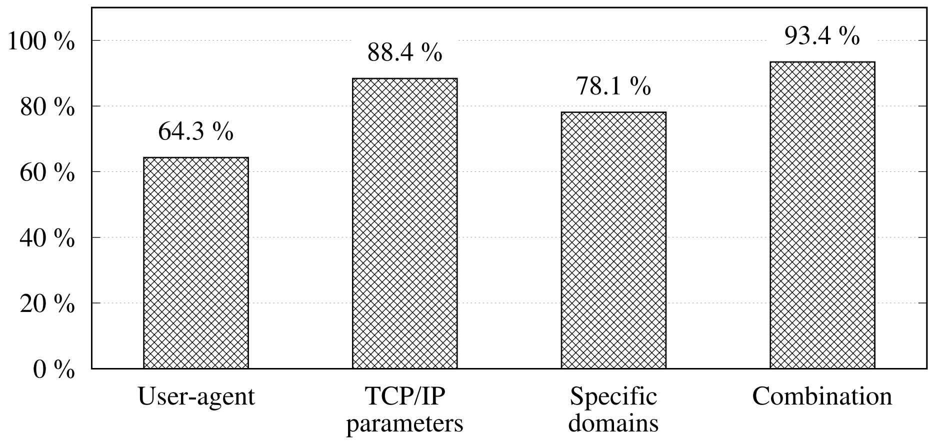


Figure 40: Coverage of OS identification methods.

Our first experiment regarding OS identification is to measure how many sessions can the methods evaluate. To evaluate a session, at least one flow in the session needs to carry the required information for a method to work. The User-agent method needs an HTTP request with a UA containing OS information. This requirement is fulfilled by 64.3 % of the sessions from the dataset. Specific domain method requires an HTTP(S) request on a domain from its dictionary. The HTTP(S) request on a domain from its dictionary is present in 78.1 % sessions. TCP stack method is the most generic and requires just a TCP connection, from which the parameters can be measured which is covered in 88.4 % of sessions. Our combination method needs at least one of the previous methods to have results and can identify OS in 93.4 % of the sessions. Summary of the coverage of OS identification methods in the dataset is depicted in Figure 40.

Combination method was able to identify OS in more sessions than other methods, which was caused by the presence of HTTP flows without TCP/IP parameters in our dataset. We have identified the cause of this problem in the flow export of initial SYN packets with flags CE (Congestion Window Reduced, ECN-Echo) set, which caused the exported to think it is not an initial SYN packet and to skip filling the extended fields. We have reported this issue to the flow exporter vendor who confirmed our findings and implemented the fix in the next version of the exporter.

The second experiment explores the performance of the methods for in OS identification. In terms of statistical analysis, our methods represent the multi-class classifiers with l non-overlapping classes. We have computed performance measures of accuracy, precision, recall, and f-score for each method according to [240]. As the distribution of operating systems is not uniform and there are significant differences in their appearances in the dataset, we have decided to use the micro-averaging technique to favor the bigger classes.

Our dataset contains ground truth for OS identification extracted from DHCP logs on the level of details of OS name from the OS hierarchy. It is also the highest level of detail the specific domains method could achieve and hence all methods performance is measured on this level of detail to ensure a fair comparison.

We have calculated confusion matrix with true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values for each of the l classes (OS names) by comparing classification result to the ground truth. Then we computed the performance measures from the following equations:

$$\begin{aligned}
AverageAccuracy &= \frac{1}{l} \cdot \sum_{i=1}^{l} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \\
Precision_\mu &= \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} (TP_i + FP_i)} \\
Recall_\mu &= \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{l} (TP_i + FN_i)} \\
F - score_\mu &= \frac{2 \cdot Precision_\mu \cdot Recall_\mu}{Precision_\mu + Recall_\mu}
\end{aligned}$$

Exact performance measures for individual methods are listed in Table 17, their comparison is then depicted on Figure 41.

Method	Accuracy	Precision	Recall	F-score
User-agent	0.9189	0.9812	0.6063	0.7495
TCP/IP parameters	0.8088	0.5249	0.4643	0.4927
Specific domains	0.8402	0.6286	0.4907	0.5512
Combination	0.8582	0.6587	0.6041	0.6302

Table 17: Micro averaging for multi-class classifier performance measures.

The user-agent method generally shows best results as applications generating HTTP requests are usually honest about its operating system. The significant drop in recall compared to other measures is caused by the high number of sessions without usable User-agent (i.e., no or encrypted User-agent sent, or it contains information only about the application and not OS) which causes many false negatives.

TCP/IP parameters method exhibits the worst performance from tested methods. We identify the primary cause in two areas – Apple products sharing the same parameters, and Windows and Android devices using many different parameters. The first issue causes that most sessions with OS from Apple family (MAC OS X, iOS, Darwin) are identified as MAC OS X since this method alone has no way to distinguish between them. On the other hand, Windows and Android devices communicate with many parameters configurations, and their traffic dominance hides other operating systems in classification.

The new method using detection of specific domains proves to be capable of OS identification at large scale and even surpass the established TCP method. It can distinguish OS with small market share but suffers from the Apple issues discussed above. All Apple products communicate with a similar set of domains and even their applications installed on different OS (e.g., iTunes) download updates from the same domains. We can generalize this statement to all vendors as it is not economical to maintain a distinct update server for every product they develop.

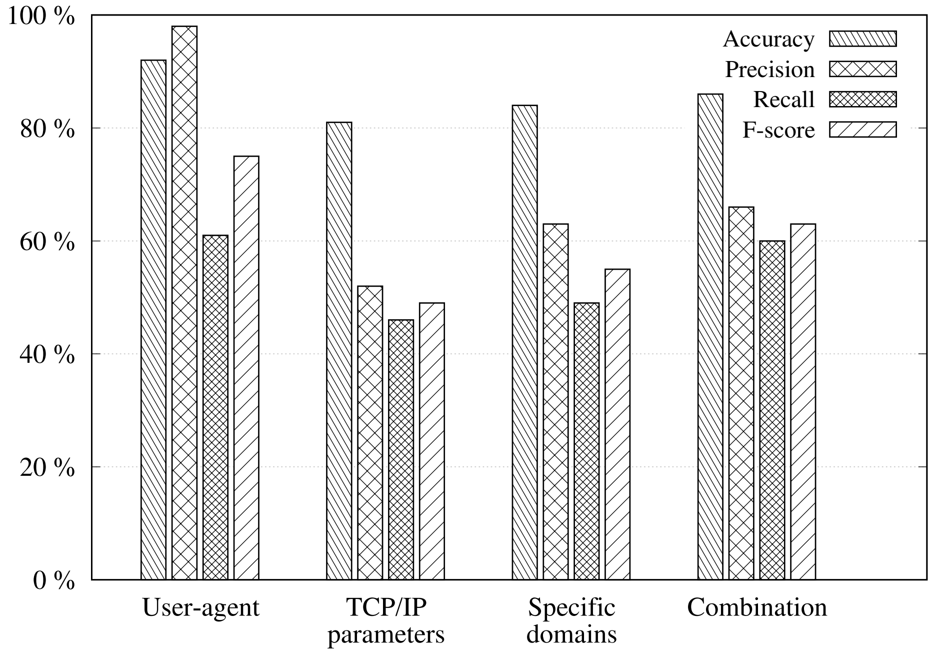


Figure 41: Micro averaging for multi-class classifier performance measures.

Our combination method embraces all positives and negatives from previous methods. It can identify OS in the highest portion of sessions and performance measures are better than TCP and specific domain methods. However, it is not as precise as the User-agent method which is the result of each method having the same weight during identification and flows without UA pushing the decision towards a wrong one.

Besides the performance of the methods, we can look at the situation in our Wi-Fi network according to the identified operating system. Figure 42 shows the market share of vendors based on results from the combination method. Mobile devices such as phones and tablets dominate the dynamic network (Android 56.18 %, MAC OS X 30.07 %) and traditional operating systems currently have decreasing popularity (Windows 4.48 %). *Unknown* means that the method could not evaluate the session and *Other* category is the rest of operating systems with a low market share (e.g., BlackBerry).

During our experiments, we had to tackle several challenges. Finding the method for establishing ground truth poses the first challenge. Our method of using DHCP log parameters of

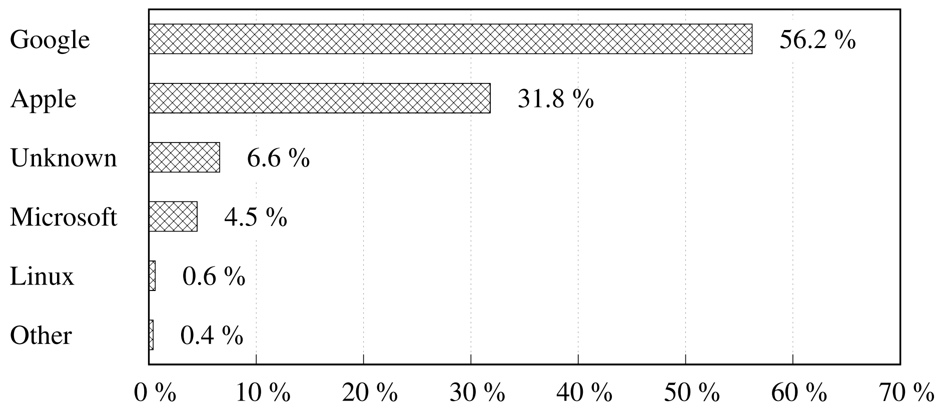


Figure 42: Operating system usage share of our network grouped by vendor.

device name works well for Apple and Google operating systems that use easily identifiable device name by default and its change by a user is discouraged or impossible. On the other hand, Microsoft and Linux operating systems allow simple device name change, and thus these devices could not be identified as easily. The second parameter we tried to use was MAC address, but it proved not to be helpful for ground truth establishment. Unfortunately, we were unable to find any meaningful mapping of Network Interface Controller vendors to operating systems of the devices due to the usage of same hardware family by multiple products with different operating systems. Ground truth establishment in large scale thus remains an open research problem for now.

Next challenge relates to the tested methods for passive OS fingerprinting. These methods will have to adapt to emerging new standards of network communication. We identify the following trends and protocols, that will have to be supported: IPv6, encrypted communication through TLS and HTTP/2.0 or QUIC [241] by Google. All of them will affect each of the methods in some way, although not necessarily to the same extent. The user-agent fingerprinting method will be the most affected when the protocols mentioned above are commonly used. Because the current trend in network communication is to encrypt all data transfers with TLS by default, the User-agent field will not be readable during data transfer. In HTTP/2.0 [242], the encryption was not made mandatory, but browser developers have explicitly stated that it will be supported only in encrypted format [243]. The QUIC protocol explicitly requires encryption of content, and only a few implementations send UAID (User Agent equivalent) in the first unencrypted client hello message. This indicates that the User-agent method's usability will be declining soon. TCP/IP parameters method will not be affected by encryption, but it will have to be adapted for concurrent use of IPv4 and IPv6. The diminishing address pool and the evergrowing number of IoT devices that need public IP addresses will require extended usage of IPv6 protocol. In IPv6, the TTL parameter equivalent, Hop Limit, is suggested by Router Advertisement messages for all connected devices and the IPv4 header field Total Length was changed to Payload Length with slightly different semantics. Specific domains method will remain functional with encrypted traffic and unaffected by underlying network protocols as the SNI field remains in all new protocols. It will even be able to identify new operating system versions from already known vendors if the domain remains unchanged. However, to keep its database up to date, it is necessary to monitor any changes in specific domains done by the vendor. If such a process is in place, we believe that this method will be the most reliable and accurate in the near future.

All these challenges push OS fingerprinting method toward decreasing level of details about the OS. It can be expected that only OS name (or even only vendor) will be distinguishable by the methods. While this situation helps protect user privacy, it conflicts with the use-case of unsupported OS version detection.

5.3.4 Host Identification in Encrypted Traffic

This section extends our approach to identification of a client in encrypted traffic described in Section 4.2. We discuss the information which can be derived from a cipher suite list (and its corresponding User-Agent) and their application. First, we present a breakdown of the dictionary for client identification according to identifiers found in User-Agents, e. g., types of a client application or a device. Second, we focus on the most interesting type of clients, web browsers and operating systems, which can be analyzed in more details. Applicability of our results is discussed in the context of browser fingerprinting, network security, and network forensics. We also discuss possible defenses against SSL/TLS fingerprinting. In following experiments, we use a dictionary created from dataset used in Section 4.2.

Classification of Network Traffic

As previously shown, we can assign a User-Agent to a known cipher suite list. However, the assignment is not exact as there are typically multiple User-Agents which correspond to a single cipher suite list (see Figure 29). The multiple User-Agents are typically similar per one cipher suite list. Thus, we can extract common information, e. g., a type of application, from them. We used HTTP::BrowserDetect tool [244] to mark and classify the User-Agent strings. The tool extracts general information on a given client, e. g., browser name, version, vendor, and operating system. Although the tool was designed to analyze User-Agents of web browsers, it can recognize web crawlers as well. The interesting option is the detection of a mobile device, its type, and vendor.

We extracted four pieces of information from the User-Agents: device type, operating system, application type and type of a web browser. If the cipher suite list corresponded to more than one User-Agent, we selected the most frequent values. For example, if the cipher suite list corresponded to Chrome four times and to Firefox one time, we assigned the Chrome browser. This method is not the most accurate but provides the most probable value. In case of the wide variability of values or inability to parse User-Agent, the *unknown* value is used. The result is a dictionary with concatenated values corresponding to the client fingerprint for every cipher suite list.

The share of client types in the dictionary is presented in Figure 43. This figure represents only the structure of a dictionary, not the relevance of particular client types. However, we can see significant shares of client types which are hard to detect using a host-based pairing method. Nevertheless, we were at least able to detect the application types using unknown device type records from the dictionary. Over one fifth of the client types remained unknown both for device and application type, though. Desktop and mobile applications typically communicate only to specific servers with a specific service. These findings demonstrate the contribution of the flow-based pairing method.

Client Identification and Browser Fingerprinting

The grouping presented in the previous paragraphs reflects the structure of clients in the dictionary. To discover the structure of clients in the HTTPS traffic, we assigned dictionary records to the real traffic. Moreover, we performed further analysis on the shares of operating systems

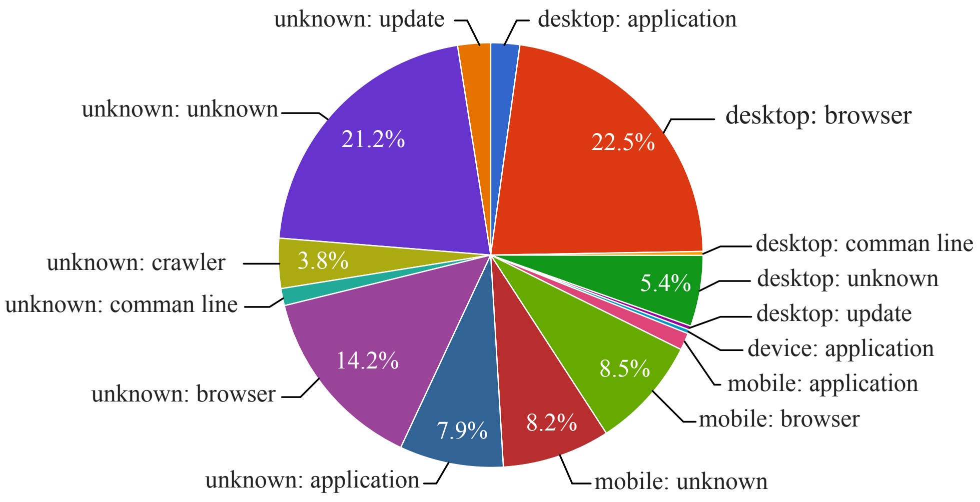


Figure 43: Shares of HTTPS client types in the dictionary.

in HTTPS traffic and on shares of web browsers in HTTPS traffic to gain deeper insight into the encrypted traffic.

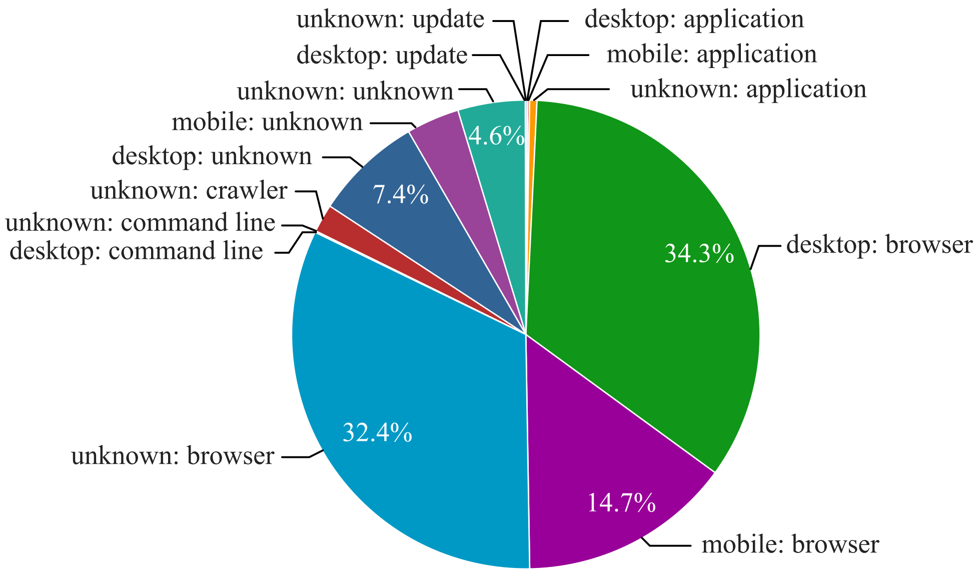


Figure 44: Shares of HTTPS client types in live network traffic.

The shares of client types in the live network HTTPS traffic are presented in Figure 44. We observed that a majority of connections were initiated by browsers. This approves the fact that we analyzed HTTPS connections primarily designed for web communication. About a third of the connection was initiated by desktop browsers. One interesting figure is the relatively high amount of traffic initiated by mobile devices. We were also able to capture machine-generated HTTPS connections by identification of traffic belonging to crawlers and updates. Only 4.6 % of the HTTPS traffic remained completely unknown.

The group of browsers, desktop, and mobile still dominated in both categories. Therefore, we picked the browser's part of analyzed HTTPS traffic to perform further analysis in more detail. The results of the analysis are presented in Figure 45. Nearly one-half of the total connections were represented by Chrome web browser, followed by Firefox with one fourth and Internet Explorer with 14.6 % of browser's encrypted traffic.

Described distribution of browsers in HTTPS traffic strongly correlates with the distribution of browsers in national browser usage statistics showed in Table 18. Even though the dictionary

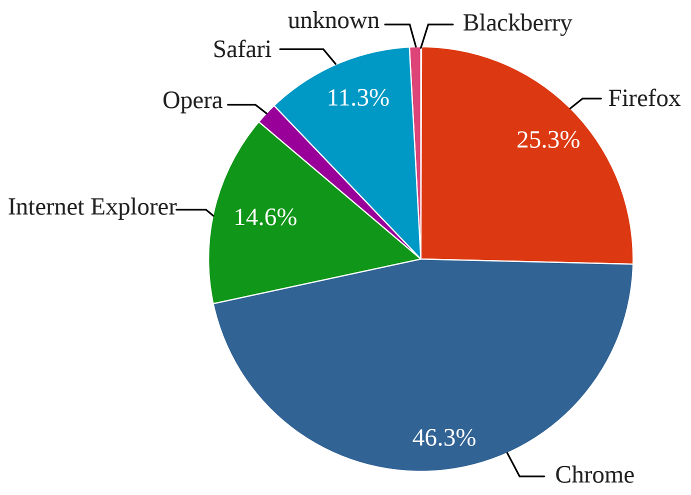


Figure 45: Shares of web browsers.

did not provide exact translation, the table shows that our browser fingerprinting method gave accurate results. We have noticed, that the distribution of browsers in unencrypted traffic varies among different regions. Therefore, the distribution of browsers can be used as lightweight geolocation of a data observation point. Using the discovered correlation of browser distribution, we can apply the browser-based geolocation in the case of encrypted traffic, too.

Browser	**Traffic share [%]**
Chrome	41.94
Firefox	26.39
Internet Explorer	17.27
Safari	5.52
Opera	4.59
Other	4.29

Table 18: Shares of used web browsers for Czech Republic. [245]

In the last analysis of HTTPS traffic we were not interested in client's application type, but rather in its implementation, i. e., operating system. The results of the analysis are presented in Figure 46. We used our dictionary to assign an User-Agent to the cipher suite list of encrypted connection. The User-Agent was then further analyzed to retrieve operating system information. The majority of the operating system represents the Windows platform. Unfortunately, there was a high share of the traffic for which we were not able to determine client's operating system due to missing or contradictory information in the User-Agent string. The shares of other operating systems were evenly distributed. To obtain a more precise operating system identification, we could use additional information, such as TCP window size or SYN packet size.

Network Security and Forensics

The next step following the classification of the network traffic and identification of the clients was the detection of specific or unusual clients in the network traffic. The problem arises in the fields of network security and forensics. Network forensics use cases typically focus on tracing the activity of a specific host in the network. If a client is known to be malicious, e. g., it is a malware, then it is also relevant for network security. Monitoring the activity of clients, which are known or suspected of being malicious, helps in the detection and prevention of attacks or

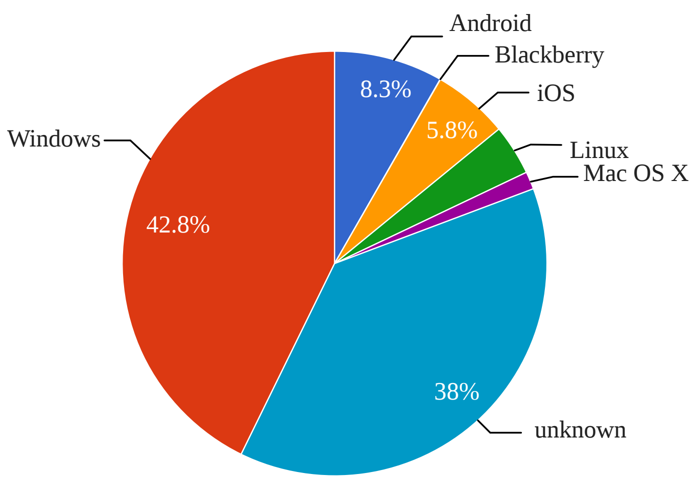


Figure 46: Shares of operating systems.

malware spreading. However, recognizing malicious activity and marking a client as malicious is a hard problem.

One of the most interesting pairs, from the network security perspective, included a specific sequence of a Bash vulnerability Shellshock in the User-Agent string. The malicious sequence "`() { :; };`" appeared in other HTTP headers as well in this case. The string is not a User-Agent, which also implies that we have no other identifier of the client. The fingerprint was obtained via a host-based method and is trustworthy. We searched for the corresponding cipher suite list in the network monitoring data but did not find any other match. Therefore, we may assume there is a unique fingerprint of a malicious client. Although we do not know more about the client, we can use the knowledge to detect suspicious cipher suite lists in the network traffic. Detection of malicious clients, which are known to exploit a vulnerability, is another use case of fingerprinting in network security.

Defense against Fingerprinting

Bujlow et al. [180] surveyed defense strategies against currently known tracking mechanisms, e. g., browser fingerprinting. We are not aware of any method which would prevent SSL/TLS fingerprinting apart from using a proxy or manually changing the cipher suite list. Using a proxy, for example, mitmproxy [246], causes the fingerprinting method to detect cipher suite list of the proxy instead of cipher suite list of a client. On the other hand, the cipher suite list of a proxy can be recognized, and the corresponding network traffic can be marked accordingly.

The second option of defense is manually changing the cipher suite list of a client. The change of the cipher suite list can be done by forced forbidding of specific cipher suites. Then, a client is communicating with reduced cipher suite list. The fingerprinting method cannot recognize the reduced cipher suite list and thus fails at finding a corresponding User-Agent. On the other hand, it is not easy to reduce a cipher suite list to forge cipher suite list of a different client.

5.4 Summary

This chapter presents several experiments that improve comprehension of a computer network. First, we investigate the properties of Top N statistics, a frequently used query aggregation used in the IP flow analyses. We focused on the availability, uniqueness and time stability aspects of the statistics. Our time stability experiment shows that the ten most frequent results of Top N statistics are observed more than in 57 % of observations in 57.8,% of IP addresses. The highest uniqueness of the Top N statistics is achieved when the HTTP host key is used as the aggregation variable. Moreover, we discovered that a single Top N statistic has a limited application on host identification problem. Nevertheless, once we were able to identify a host, the host was identified with high precision when we used the destination IP or HTTP host characteristics – 77.23 % and 36.49 % of the hosts were identified unambiguously.

Next, we researched the IPv6 transition mechanisms to enhance our understanding of the encapsulated IPv6 traffic. We describe characteristics both of the IPv4 tunnel traffic and the IPv6 tunneled traffic. We analyze the TTL and HOP distributions, geolocation of IPv4 and IPv6 endpoints, and evaluate IPv6 adoption. Our results show the increased relative amount of 6to4 traffic compared to Teredo traffic.

Last, we analyze the host-related information to demonstrate the capabilities of IP flow analysis to derive host-related information from network traffic. We investigate the identification of the operating system from IP flow records captured both in static and dynamic networks. Further, we extend our research on host identification in encrypted traffic presented in the previous chapter and discuss the information which can be derived from a cipher suite list and their application. We display feasibility for network traffic classification, client identification, and browser fingerprinting.

The main contributions of this chapter are:

- evaluation of the availability, uniqueness and time stability aspects of Top N statistics,
- evaluation of the feasibility of Top N statistics for host identification,
- analysis of the IPv6 encapsulate traffic,
- assessment of OS fingerprinting methods both in static and dynamic networks,
- research of identification of client and browser in encrypted traffic.

6 Towards Real-time Cyber Situation Awareness

Cyberspace is a highly dynamic environment that changes every second. The dynamics of the cyberspace imposes demanding requirements on the cyber situation awareness. To achieve the primary purpose of the cyber situation awareness, i.e., to provide relevant and up-to-date support for a decision process, a large volume of data needs to be processed at high speeds to support an operator with an instant analysis results. The IP flow network monitoring workflow is designed to handle high volumes of data at high speeds. However, the IP flow network monitoring introduces several delays, as shown in Chapter 3. These delays slow down the analysis process and the results provided to an operator might be outdated at the time they are computed and delivered due to the high dynamics of the network.

This chapter describes our contribution to the improvement of the speed of the analysis process in the cyber situation awareness. We reduce the time needed for IP flow analysis by introducing a stream-based IP flow network monitoring. The shift from the traditional batch-based to the stream-based analysis approach increases the analysis speed as IP flow records are analyzed instantly. Moreover, the stream-based workflow is naturally distributable and scalable, which increases performance and further reduces analysis time. A shift from bath-based to stream-based approach, however, implies also a shift in the analysis approach. We discuss the implications of the shift and describe the pitfalls and advantages of the stream-based analysis.

The stream-based approach is not meant to substitute for the batch-based IP flow network monitoring. On the contrary, the stream-based approach is a natural complement of the traditional batch-based IP flow network monitoring. We propose a next-generation IP flow monitoring infrastructure that merges the batch- and stream-based approaches into a single approach that keeps the advantages of both approaches. We describe individual parts of the infrastructure and suggest suitable software for these parts.

This chapter concludes with a proposal of an approach to real-time cyber situation awareness, that builds upon the described next-generation IP flow monitoring infrastructure. The next-generation IP flow monitoring infrastructure is generalized to process any cybersecurity-related data in a unified manner allowing for the enhanced analyses in real-time.

This chapter responds to the performance-related open issues of the cyber situation awareness (see pp. 25,26). Further, we address the data challenges of the cyber situation awareness (see p. 25). From the IP flow monitoring challenges, we address the performance, scalability, and response time open issues of IP flow monitoring (see p. 47) and the complexity open issue of the IP flow analysis (see p. 53).

This chapter is based upon the research results published in the following author's peer-reviewed publications: [A1–A3, A11].

This chapter is structured as follows:

- *Section 6.1 describes the application of the data stream processing techniques to the IP flow record analysis to achieve real-time analysis results.*
- *Section 6.2 presents a next-generation IP flow network monitoring created by the combination of the batch- and stream-based approaches to the IP flow analysis.*
- *Section 6.3 proposes a framework for real-time cyber situation awareness derived from the generalization of the next-generation IP flow network monitoring infrastructure.*

6.1 Real-time IP Flow Analysis

IP flow monitoring and their analysis play a vital role in network traffic measurements for cybersecurity. Currently, IP flows are widely used for traffic measurement in large-scale, high-speed networks, cloud environments, and various enterprise networks. IP flow analysis is used for detecting the majority of severe contemporary cyber threats, such as Denial-of-Service (DoS), botnets, and Advanced Persistent Threats (APT) [125]. Moreover, the analysis can be done both on unencrypted and encrypted traffic, as IP flows gather information only from packet headers. Such advantages have made IP flow network monitoring a fundamental part of traffic measurement for cybersecurity.

Nevertheless, IP flow analysis still faces several challenges raised by the rapid evolution of the threat landscape as demonstrated in Chapter 3. In this section, we focus our attention on the following ones. First, network traffic measurement has become a Big Data problem. Due to the increasing volume of network traffic, it has become expensive and impractical to store first and then read again all IP flows from large networks for analysis. Second, it has become impossible to analyze the large volume of IP flows from networks in real-time. Current approaches try to face this challenge by increasing the hardware performance of analytical machines or simple master-slave architectures. Still, the IP analysis itself stays centralized, scalability of these solutions is limited, and analysis time remains relatively long. We demonstrated this fact in the description of IP flow monitoring workflow timeline in Section 3.3.5. Third, the time needed to detect a cyber attack remains a challenge for IP flow analysis. Current IP flow-based cybersecurity solutions exhibit a detection delay in the order of minutes. Such a delay may be fatal when we try to reduce the harm caused by an attack. Therefore, demands for near real-time attack detection have risen recently. Current IP flow traffic measurement tools fail to satisfy this demand, though.

To address the challenges mentioned above, we present a transformation of the current IP flow monitoring and analysis into the stream-based paradigm. In this approach, the IP flows are processed and analyzed in data streams immediately after an IP flow is observed. The analysis of IP flows in data streams reduces the volume of data that needs to be stored because data is kept in primary memory for the time necessary for processing and only results are stored in the secondary memory. This feature represents the greatest advantage of the stream-based concept. It allows us to perform the immediate data analysis, which makes the real-time attack detection possible.

6.1.1 Basic Concepts

To cover the basic concepts used in our approach, we recall relevant information on IP flow-based network monitoring from Chapter 3 and discuss the basics of data stream processing. This overview can be considered as a high-level abstraction of the main ideas of both areas, for a detailed description consult [88, 125, 247].

IP Flow Network Monitoring

IP flow monitoring was principally designed to monitor high-speed network traffic in large-scale networks. Since the performance limitations do not allow us to process, store and analyze all information from each packet in such networks (Deep Packet Inspection), an abstraction of single direction communication, called an IP flow, was introduced. An IP flow is defined as a set of packets passing through a point in the network during a certain time interval. All packets belonging to a particular IP flow have a set of common properties called IP flow keys. The traditional 5-tuple of IP flow keys is comprised of source and destination IP address, source and destination port, and transport protocol. Apart from the traditional 5-tuple, the IP flow contains statistics about the connection (such as the number of packets in an IP flow) and may be enriched by information from the application layer of network traffic. IP flow information is stored in IP flow records.

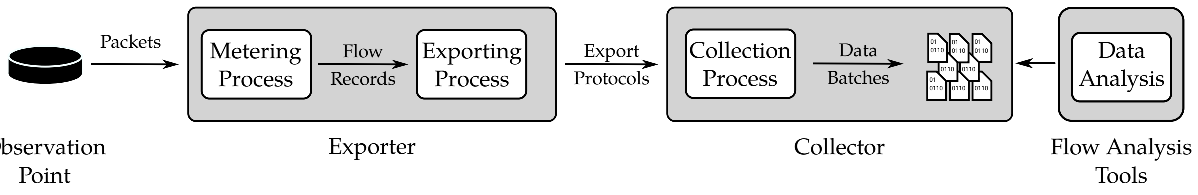


Figure 47: IP flow monitoring workflow.

The IP flow network monitoring process is a complex system, as depicted in Figure 47. An IP flow record is generated at an observation point in a network by an exporter. The exporter captures information from packet headers and creates flow records during the metering process. The created flow records are submitted to the exporting process to be sent to a flow collector via export protocols, such as NetFlow or IPFIX. The collector receives flow records from one or more exporters, processes them, and stores them for further analysis. The collecting process manages flow records and stores them, usually in one- to five-minutes batches into binary flat files (*nf-dump*, *SiLK*, and so on) or column-oriented databases (*FastBit*, *Vertica*, and so on). Row-oriented databases (MySQL, PostgreSQL, and so on) are not suitable for flow storage and querying due to their insufficient performance. Individual batches are then available for further data analysis.

There are three main application areas of IP flow analysis: Flow Analysis and Reporting, Threat Detection, and Performance Monitoring. Flow Analysis and Reporting covers querying and filtering flow data for relevant information (network visibility), statistics overview of the data (Top N statistics and so on), traffic accounting, reporting, and alerting (such as exceeding transfer data quota). The Threat Detection area focuses on the analysis of specific traffic events, most often scans, DoS, worms, and botnets [2]. Performance Monitoring reports the status of running services on the network by observing application metrics, such as delay, jitter, and round-trip-time.

All three application areas for IP flow analysis have one thing in common – a time aspect (see pps. 44, 52). As the network is monitored in time, the majority of statistics, detection methods, and performance characteristics are aggregated over a given time window (such as top talkers in the last hour, the number of transferred bytes in the last minute). The time window is strongly influenced by the settings of the network monitoring process, namely the size of the batches in the collecting process. Data analysis can be performed only when a new batch occurs. The batch is set to five minutes in the majority of IP flow analysis tools. The analysis is then run every five minutes. This means that, for example, an attack or service outage may be detected with a five-minute delay. However, the demand for real-time analysis has risen recently to achieve shorter detection and reaction times. The analysis delay can be shortened by replacing the batch-based analysis with stream-based analysis, where each flow record is analyzed immediately as it arrives.

Data Stream Processing

Stream processing systems (historically referred to as Active Database Systems [248], or later, as Data Stream Management Systems [247]) emerged in response to the poor performance of traditional persistent databases, which were not designed for the rapid and continuous updates of individual data items continuously arriving at high velocities. Use-cases, where it is suitable to employ the stream processing approach, are applications where data is modeled best not as a persistent relation but as a transient data streams [247]. These application are collectively referred to as *information flow processing (IFP)* domain applications [249]. Examples of the IFP application range over disparate fields; from environmental monitoring that process incoming streams of data from sensors deployed in a field, e.g., in a rivers, and aims to understand the observed world, predict anomalies, e.g., floods, over financial application that can process tick prices at an exchange to identify trends or detect a fraud or market manipulations, to security monitoring of a computer network, where observed network traces are continuously analyzed to detect attacks.

The data streams differ from the traditional relational model in the following ways [247]:

- The data transferred in a stream arrives online, instantly.
- The system for data stream processing does not have any control over the order in which data arrive in the system. Moreover, the data can be processed either within data stream or across different data streams.
- The data streams are possibly infinite sequences.
- Once a data in a data stream is processed, it is discarded or archived.

A data stream is a possibly infinite sequence of data elements with a given schema and assigned timestamp of its occurrence, which is discrete and ordered. The data stream can be continuously transformed by an operators [250]. The most common operators are *split* and *merge*. The split operators divide a single input data stream into multiple output data streams, while the merge operators combine multiple input streams into a single output stream. The split and merge operators may forward data in data streams either unmodified, or transformed; a relational *join* operator that includes a nontrivial data transformation,for example.

The queries over data streams are in many ways similar to the queries addressed to traditional databases. Nevertheless, there are two important distinctions intrinsic for data streams. The first distinction is the suitability of query types. While traditional Data Management Systems (DMS) are suited for *one-time queries*, i.e., queries that are evaluated only once, the data stream model is suited for execution of *continuous queries*, i.e., the queries that are reevaluated in time and reflect data seen so far. Naturally, both query types are available in both (traditional and stream) models. Nevertheless, the continuous queries in data streams outperform these queries in conventional DMS. The other distinction is between an approach to predefined and ad-hoc queries. The *predefined queries* are known in advance, i.e., before a data stream is processed. The *ad-hoc queries* are known after a data is processed. The ad-hoc queries are a challenge for a data stream processing systems as it is not known, which data is required to resolve the query (the data may already be discarded), the data streams cannot be optimized for query resolution, and so forth.

The data streams are ordered mostly by an assigned timestamp. We differentiate between two main timestamps: *implicit* and *explicit* [247]. The implicit timestamps are attached to each arriving data in a data stream by a data stream processing system. In many cases, this timestamp assignment (and data ordering) is sufficient. It is often used, when data itself does not contain any timestamp, or when the exact moment the situation occurred is not important, and generalizations such as a *"recent event"* are acceptable. In other cases, explicit timestamps are used. The explicit timestamps are timestamps that are provided in a data stream. These timestamps are assigned to a data usually by the data producer itself, for example by a metering sensor. The explicit timestamp represents an exact time of event occurrence. These timestamps are used when the exact ordering of the data stream is essential, for example, in sequence based algorithms. When explicit timestamps are used, we need to keep in mind, that the data can arrive in a data stream processing system unordered, i.e., unordered by the explicit timestamps, due to data transmission latency, for example. The correct ordering of the data can be achieved by adding an input buffer for incoming data and order the incoming data in this buffer.

Queries over data streams are usually evaluated over a time windows. According to Gama and Rodrigues [251], following window models are most relevant to data stream processing:

- **Landmark Windows** – the landmark window is initiated by an event (landmark) in the data stream, and all data succeeding the landmark belong to the window. The landmark can be, e.g., a start of a day, or a start of production.

- **Sliding Windows** – given the size of the sliding window l and slide of the sliding window s, the sliding window that represents data in interval $\langle t_0, t_l \rangle$, $\langle t_{0+s}, t_{l+s} \rangle$, respectively. The window slide is usually triggered by timeout. Nevertheless, it can be triggered by the arrival of new data as well. Based on the slide size, we differentiate between overlapping windows, i.e., there is an intersection of the windows, or disjoint window, where there is no intersection of two consecutive windows.

- **Titled Windows** – the titled windows compress the time scale so it can be used for long-term analyses. The compression is usually one of the following: natural, or logarithmic. The natural compression stores the latest data with the highest granularity and as the data gets older, the granularity of the stored data decreases, i.e., data are aggregated. The typical model of natural compression is to store last hour in minutes (60 observations), last day in hours (24 observations), last month in days (31 observations), last year in months (12 observations). The logarithmic compression the granularity decreases logarithmically as data gets older.

Stream processing systems are designed to evaluate continuous queries over many data streams in real-time, while predominantly using only primary memory for storage. The existing implementations of stream processing systems differ in several aspects including, but not limited to, query language capabilities, the nature of processed data, time model, and so on. For example, *Esper*[1] is a full-fledged stream processing engine focused on evaluating continuous SQL-like queries over streams of events. For an extensive survey of stream processing systems see [249].

Nowadays, a new generation of stream processing systems is emerging that is generally referred to as distributed stream processing frameworks. These systems are used to process generic data streams and provide capabilities for distributed processing. In many cases, users must implement their own processing logic, yet they are provided with powerful abstractions that allow them to transparently execute the implemented logic in a parallel-distributed way. The most notable examples of distributed stream processing frameworks include *Samza*, *Spark*, *Storm*, and *Flink* (all maintained by Apache Software Foundation)[2]. The benchmarks of the systems are available in [252, 253].

The key differences between traditional data processing and stream processing are summarized in Table 19.

6.1.2 Stream-based IP Flow Analysis Framework

The interconnection of the stream processing framework and IP flow-based network monitoring raises new challenges and requirements that must be addressed. We summarize the functional and non-functional requirements for these systems and describe the logical architecture of such systems. To demonstrate the possibilities of real-time analysis, we present the *Stream4Flow* framework that is based on modern systems for large data processing.

1. `http://www.espertech.com/`
2. `http://{samza|spark|storm|flink}.apache.org/`

Traditional processing	vs.	Stream processing
Data stored as persistent sets	**Data**	Infinite streams of individual data tuples
Large secondary memory	**Storage**	Small primary memory
Ad-hoc	**Queries**	Continuous
No real-time capabilities	**Real-time**	Real-time processing
Single-query	**Optimization**	Multi-query
Mature tools and technologies	**Maturity**	New tools and technologies

Table 19: Differences between traditional and stream data processing.

Design Considerations

To successfully transform the batch-based IP flow processing into stream-based, it is necessary to meet the same requirements as the original approach and in real-time. The data processing speed plays an especially important role, but so do other requirements must be met, such as a set of available data processing operations, fault tolerance, and system durability. As regards to the minimal data processing speed of the approach, it must at least correspond to the average number of flows generated by observation points inside the monitored network. For example, in a medium sized network of 24,000 active IP addresses, we observed an average of 12,000 flows/second and 110,000 flows/second in the national-wide research and education network. It can be expected that these numbers will grow in the future and, for that reason, the scalability possibilities of the stream-based processing should also be considered so that it will not be necessary to significantly change the data processing algorithms.

The stream-based approach of data processing must enable the analysis of the IP flow data in a similar way as traditional batch-based approaches. This means that it should provide at least the same basic set of data processing operations. Based on the common IP flow analysis algorithms, we identified the following minimal set of operations that should be provided: filter, count, aggregation, combination, sort, and Top N. The stream-based approach should also enable us to apply these operations to larger units of data and, thus, the window function is necessary to supply traditional batch-based approaches. In addition to the available operations, stream-based data processing must also ensure that each flow was processed just once to avoid skewed results. Thus, the recoverability and durability options of data processing system should be considered too.

Architecture Design

Analyzing IP flows in real-time was almost impossible in previously due to the poor performance of data processing systems. In recent years, however, a change has occurred. Tools for fast batch-based and stream-based processing of large volumes of data were progressively introduced. In the paper by Cermak et al. [253], the authors demonstrated that distributed stream processing frameworks, such as *Spark*, *Samza*, and *Storm*, are able to process at least 500,000 flows/s using 16 or 32 processor cores, which significantly exceeds the minimum demand. Thanks to this, it is possible to utilize these frameworks and extend current IP flow monitoring and analysis architecture. This enables to analyze IP flows in real-time and provides other analytical methods that are not possible, or difficult to achieve, in common batch-based systems.

A generic interconnection of modern distributed stream processing frameworks into the common IP flow monitoring architecture is shown in the Figure 48. We can see that the first part of the monitoring architecture is the same as in traditional IP flow network monitoring. The difference is in the second part, focused on the analysis of monitored data. To allow such interconnection, it is necessary to enable the collector to transform IP flow records into a suitable Data Serialization Format (DSF). Alternatively, the collector can be omitted from the architecture if the IP flow exporter can provide flow records in such a format. The typical format for distributed stream processing frameworks is the JavaScript Object Notation (JSON) format, which enables it to suitably represent any data records. However, the JSON format is not space efficient and can cause overloading of the network if a lot of IP flows are processed. In the case of a large amount of transmitted data, it is better to utilize a more space efficient data serialization format, such as binary JSON (BSON) or MessagePack.

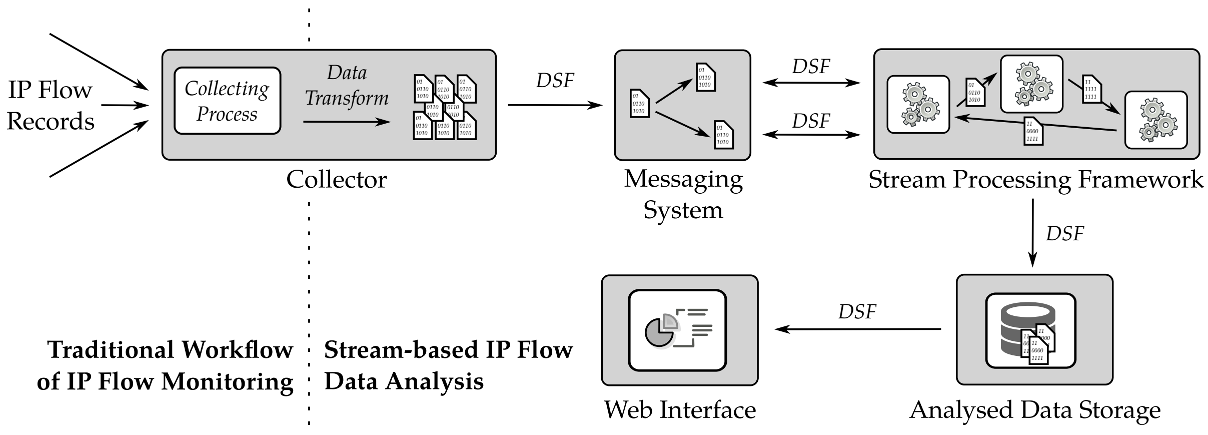


Figure 48: The architecture of a stream-based IP flow analysis framework.

The collector's ability to transform IP flow records into a suitable data serialization format is currently not widespread for common IP flow collectors, but several solutions, such as *IPFIXcol* or *Logstash* (part of Elastic Stack software family[3]), exist, and it can be assumed that new solutions will emerge in the near future. To effectively distribute the transformed IP flows, it is advisable to utilize a messaging system that serves as an input interface for the stream processing framework. There are many such systems, such as *ActiveMQ*, *RabbitMQ*, or Apache *Kafka*[4] (for the full list see [254]), however, to process IP flows it is necessary to select one providing sufficient throughput. Currently, the most suitable system is Apache *Kafka*, which offers sufficient message throughput and is compatible with most data stream processing frameworks.

As mentioned earlier, modern distributed stream processing frameworks, such as *Samza*, *Storm*, *Spark*, and *Flink*, provide sufficient data processing throughput. Thus, in the case of selecting an appropriate system, the decision needs to consider the deployment environment and functional requirements, such as data reliability, scalability, that suit to the considered use well [253]. The intended use must also be considered during the selection of an appropriate data storage for analysis results that can be stored in a common relational database, as well as in a next-generation database. The storage should support advanced queries over stored data and provide an optimal interface for a web interface so that IP flow analysis results can be visualized to a user.

As discussed above, the stream-based IP flow analysis framework combines several interconnected components. The choice of systems for each of the components should reflect the proposed use of the framework, deployment environment, and other mentioned requirements. A list of the most suitable systems for a stream-based IP flow analysis is listed in the Table 20.

3. `https://www.elastic.co/`
4. `https://kafka.apache.org/`

Architecture component	Suggested systems
Collector	IPFIXcol, Logstash
Messaging system	Apache Kafka, NATS, RabbitMQ (The full list available at [254])
Stream processing framework	Spark Streaming, Flink, Samza, Storm, Trident (All maintained by Apache Software Foundation)
Data storage	Elasticsearch, Druid, OrientDB (Next Generation Databases)
User interface	Kibana, Grafana, Tableau

Table 20: Suggested systems for the stream-based IP flow analysis architecture.

6.1.3 Stream4Flow

To demonstrate possibilities of the architecture for real-time analysis of IP flows, we introduce the *Stream4Flow* framework[5]. The *Stream4Flow* serves as an automated framework for rapid prototyping of real-time IP flow analysis. The architecture of the framework is based on the design described in the previous section. Besides the real-time analytics, the framework provides the following features:

- **Full stack solution** – we integrate all tools necessary for IP flow analysis. The framework enables a user to collect data from various network probes via IPFIX, NetFlow v5/v9 protocols, process IP flow records, and store and present the analysis results.
- **High performance** – the framework leverages the distributed nature of its components, which provides scalability in terms of memory, number of CPU, and disk space available for computation. Moreover, included tools for data analysis allow for parallel computing, which ensures high throughput capabilities.
- **Easy deployment** – we are aware, that deployment of a monitoring infrastructure is not a trivial task. Therefore, we automated the deployment of the framework using Vagrant[6] and Ansible[7], tools for software orchestration. The deployment is available either for cloud environment or in a standalone mode. Initial tests and an application template are included to ease the familiarization with the framework.

The architecture of *Stream4Flow* is depicted in Figure 49. We slightly modify the general stream-based architecture presented in Figure 48. The results from a stream processing framework are not sent directly to analyzed data storage. Instead, they are sent back to the messaging system, so the messaging system serves as a central data hub. This optimization provides us with a coherent interface for stream processing applications and allows for the creation of chains of application used for advanced analyses.

Collection of the data from network probes is handled by *IPFIXcol*[8] collector. The *IPFIXcol* is an advanced substitution for *nfdump* that enables online transformation of the incoming IP flow records into JSON format, contains advanced tools for IP flow record filtering and anonymization, and provides methods for IP flow record forwarding, for example over a network (TCP/UDP), or to other tools (Apache *Kafka* connector). The collector can also serve as long-term storage of the IP flow records. The captured IP flow records are transformed into

5. `https://stream4flow.ics.muni.cz/`
6. `https://www.vagrantup.com/`
7. `https://www.ansible.com/`
8. `https://github.com/CESNET/ipfixcol`

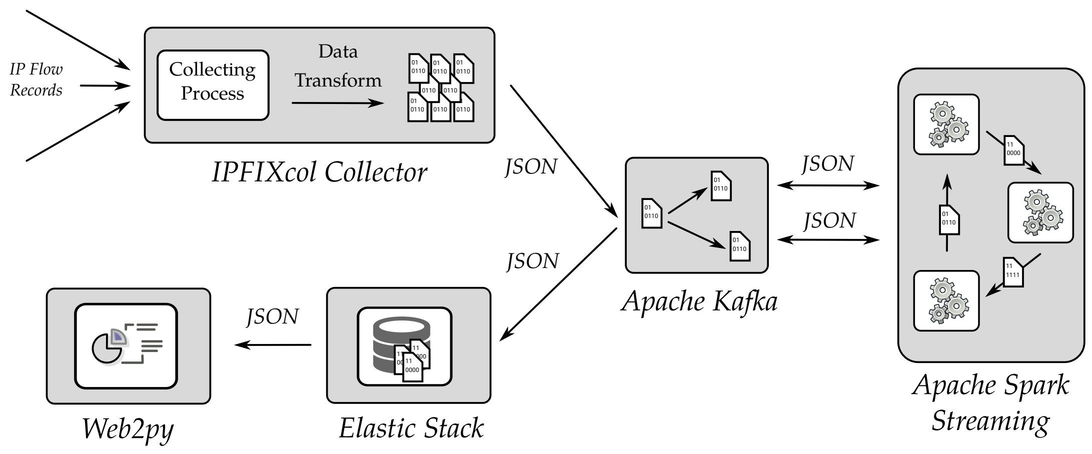


Figure 49: *Stream4Flow* architecture.

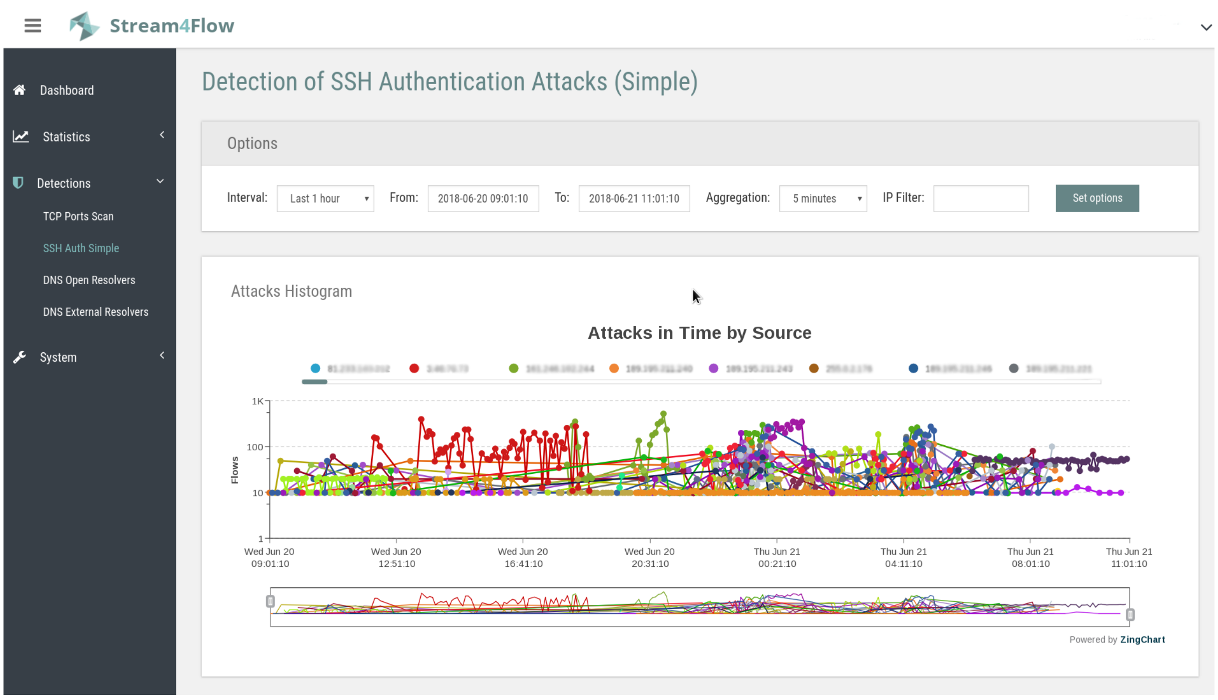
JSON format, anonymized if required, and sent to a messaging system. The JSON format was chosen for its readability and easy processing. We are aware that JSON format is not optimized for transmitting data over a network. To save network bandwidth, we recommend using BSON or MessagePack data serialization formats as proposed earlier in the description of general architecture.

The Messaging system is represented by Apache *Kafka*. *Kafka* is a distributed streaming platform that enables you to publish and subscribe to streams of records. Further, it allows you to store streams of records in a fault-tolerant way and to process data streams as they occur. Apache *Kafka* is suited for building real-time streaming data pipelines that reliably get data between systems or applications. The selection of *Kafka* was based on its scalability and partitioning possibilities, which provide sufficient data throughput [255].

The data processing core of the framework is Apache *Spark*[9]. Apache *Spark* is a fast and general-purpose cluster computing system. We employ an extension of the core Spark API *Apache Spark Streaming*[10] that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like *Kafka*, *Flume*, *Kinesis*, or TCP sockets, and can be processed using complex algorithms expressed with high-level functions like map, reduce, join, and window. Moreover, it is possible to apply *Spark*'s machine learning and graph processing algorithms on data streams. *Apache Spark Streaming* was selected as the data stream processing framework for its quick IP flow data throughput [253], available programming languages (Scala, Java, or Python) and MapReduce programming model.

Elastic Stack[11], namely *Logstash*, *Elasticsearch*, and *Kibana*, are used as a storage for analyzed data. Results are sent from *Kafka* to *Logstash* that serves as connector to the next-generation database *Elasticsearch*. *Kibana* is a user interface for *Elasticsearch* that enables a quick and effortless first visualization of the analysis results. The whole stack is distributed and designed for horizontal scalability and reliability. Although *Kibana* can provide initial visualization of the analysis result, we still included a customized web interface for result presentation. The web interface presents a highly customized and domain-specific visualization analyzes, that have proven worthy. The web interface, shown in Figure 50, is built upon *Web2py*[12], an open source full-stack model-view-control web framework.

9. `https://spark.apache.org/`
10. `https://spark.apache.org/streaming/`
11. `https://www.elastic.co/`
12. `http://www.web2py.com/`



Detection of SSH Authentication Attacks (Simple)

Options

Interval: Last 1 hour From: 2018-06-20 09:01:10 To: 2018-06-21 11:01:10 Aggregation: 5 minutes IP Filter:

Figure 50: *Stream4Flow* web interface of application for SSH attack detection.

We provide following applications for IP flow record analysis. The applications implement analyses common for in traditional batch-based approaches. The analyses are transformed into stream-based paradigm using time windows set to the batch size.

- **Protocol Statistics** – computes utilization of basic IP protocols (TCP/ UDP/ Other).
- **DNS Statistics** – computes basic Top N statistics of observed DNS servers, DNS record types, and queried domains.
- **Host Statistics** – creates a basic profile of a host in a network.
- **TCP Port Scan** – detects horizontal and vertical port scans.
- **SSh Auth Simple** – is a basic detection for SSH brute-force attacks.
- **DNS Open Resolvers** – detects open resolvers in a specified network.
- **DNS External Resolvers** – detects external resolvers in a specified networks.

To demonstrate the performance of the *Stream4Flow* framework, we display data from our production deployment of the framework. The framework is used for monitoring of campus backbone network. The framework is deployed in VMware resource pool of total 35 GHz CPU and 97.66 GB memory. The resource allocation of individual machines is presented in Table 21. *Producer* machine includes *IPFIXcol collector* and messaging system *Apache Kafka*. *Consumer* machine provides *Elastic Stack* and web interface of *Stream4Flow*. *Spark Master* hosts the master machine of the *Spark Streaming* cluster and *Spark Workers* host the workers of the distributed *Spark Streaming* cluster. We assigned the largest amount of resources to the *Spark Master* machine as the master node of the Spark cluster performs both computation distribution to the workers and the collection and reduction of the partial results computed by the workers.

Protocol Statistics application represents a basic example of the stream based analysis of IP flow records. The primary purpose of the application is to compute the shares of different

Machine	**CPU Allocation** (MHZ)	**Memory Allocation** (MB)
Producer	2500	4096
Consumer	2500	8192
Spark Master	4000	16384
Spark Workers 1-4	3000	16384

Table 21: Resource allocation of *Stream4Flow* machines.

transport protocols in the network traffic. The schema of operation over input data stream is depicted in Figure 51.

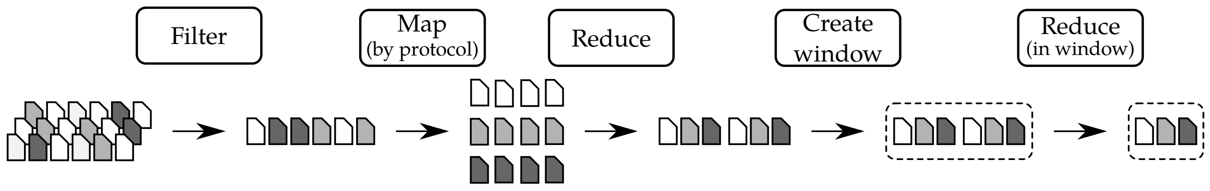


Figure 51: Schema of data processing in Protocol Statistics application.

By default, the application is set to use five-second micro batches, and the statistics of the protocols are computed over ten-second observation windows. The input data stream contains all IP flow records observed at an observation point. First, we reduce the amount of data processed in the data stream. For computation of the protocol statistics, we require only protocol number from the whole IP flow record. Hence, we filter out IP flow record with the filled-in protocol number key field, and we remove all other IP flow record keys. Second, we initialize Map function to distribute the computation of the protocol statistics to workers. The protocol number field serves as the mapping key. Third, the overall number of flows, packets, and bytes statistics for all protocols are computed using the Reduce function. The Reduce function creates a stream of the statistics calculated for each resilient distributed datasets[13] and each protocol. Fourth, we group the stream of the protocol statistics into a sliding window. Size of the sliding window is set to ten seconds, and the slide is set to the same value which results in disjoint sliding windows. Last, we compute statistics for each protocol over the given window using Reduce function once again.

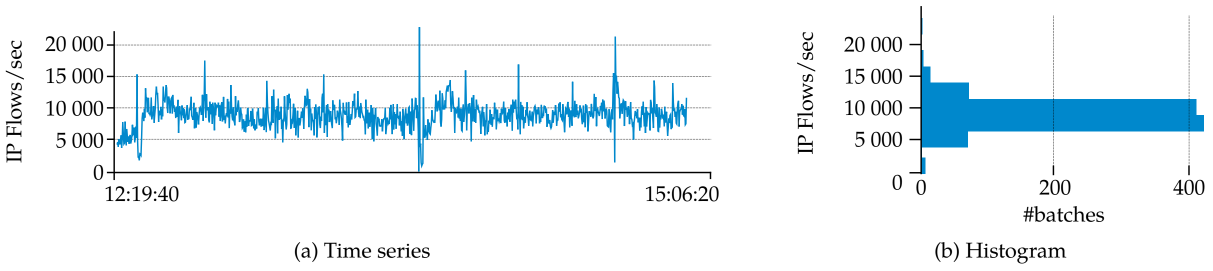
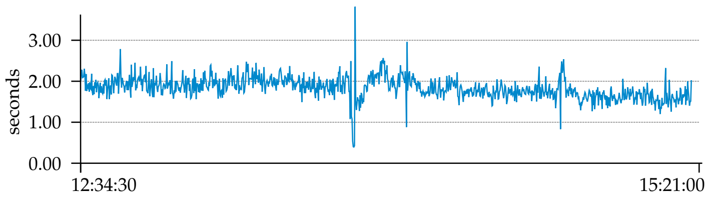


Figure 52: IP flow input rates.

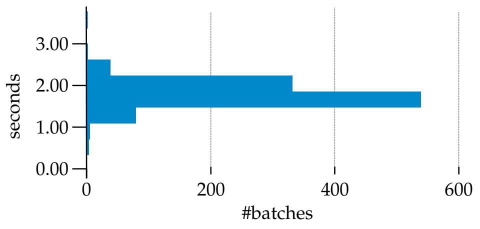
We deployed the application in the *Stream4Flow* framework and allocated 3 CPUs out of 32 CPUs total and 2 GB per worker (i.e., 8 GB in total) out of 64GB total memory for it. The application processed all monitored IP flows observed at the backbone line. The input rate of

13. Resilient Distributed Dataset (RDD) is the basic abstraction in *Spark*. RDD represents an immutable, partitioned collection of data that can be operated on in parallel.

the IP flows that entered the application is depicted in Figure 52. On average, the measured input rate was 9 101.44 IP flow records per second which reflects the distribution of input rates in individual batches presented in the histogram.



(a) Time series



(b) Histogram

Figure 53: Batch processing time.

We measured the time needed for processing a batch of RDDs. The results of the measurement are presented in Figure 53. The batch size was set to ten seconds. The processing time of a data contained in a batch (i.e., on average 91 014.4 IP flow records) was 1.836 seconds. Hence, one IP flow record was processed in 20.1726 μs. Since the batch size is set to ten seconds, i.e., every ten seconds a new batch ready for processing, the time needed for batch processing is required to be lower than 10 seconds. Otherwise, the scheduling and total delays increases, the data are not processed on time, the back pressure increases and the application is not stable in the long term. Our measurement showed that the average time needed to process a batch is 1.836 seconds with maximum values lower than 4 seconds, which is lower than ten seconds making our application stable.

Host Statistics application is designed to compute selected statistics for all hosts in a monitored network from IP flow record. As explained earlier, IP flow records resemble a connection in network traffic and need to be transformed at high computation costs to provide a host-based view. The host statistics application demonstrates the advantages of the distributed data stream processing that allows us to compute the host statistics for connection-oriented IP flow records even in real time. The implemented application computes following statistics for each host in the monitored network for each observation window[14]:

- *Basic statistics* – sum of flows, packets, and bytes
- *Port statistics* – number of distinct destination ports
- *Communication peers* – number of distinct communication peers
- *Average flow duration* – an average of duration of all flows in a given observation window
- *TCP flags* – count of each individual TCP flags in all connections

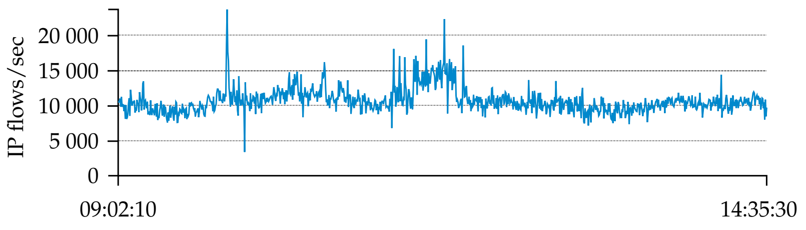
The pseudocode below provides an example of distributed data stream processing to computed basic statistics and port statistics.

```
# Filter out only relevant IP flows that contain required IP flow keys
flow_with_keys = input_stream.filter("Keys necessary for statistics computation")

# Compute basic hosts statistics – a number of flows, packets, bytes sent by a host
## Create a map
flow_ip_total_stats = flow_with_keys.map( key: "src IPv4 address", values:("flows",
    "packet count", "bytes count"))
```

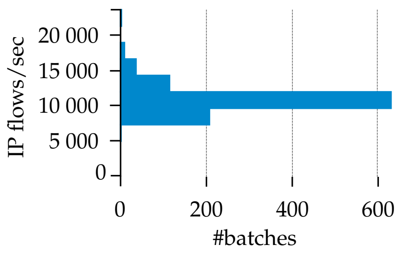
14. Minimal observation window is a one-second window.

```

## Reduce a map and compute results
flow_ip_total_stats.reduceByKey(lambda actual, update: (
                        actual("flows") + update("flows"),
                        actual("packet count") + update("packet count"),
                        actual("bytes count") + update("bytes count")))

## Create sliding windows for aggregation
flow_ip_total_stats.window("window duration", "window slide")

# Aggregate information in windows
flow_ip_total_stats.reduceByKey(lambda actual, update: (
                        actual("flows") + update("flows"),
                        actual("packet count") + update("packet count"),
                        actual("bytes count") + update("bytes count")))

# Compute a number of distinct communication peers of a host
## Create communication pairs for each host
communication_peers = flow_with_keys.map(
    key: "src IPv4 address", values: ("src and dst IPv4 address pair", 1))

## Collect information to master
communication_peers.reduceByKey(lambda actual, update: actual)

## Create sliding windows for aggregation
communication_peers.window("window duration", "window slide")

## Aggregate information for host in a window (remove duplicity in communication
    pairs).
communication_peers.reduceByKey(lambda actual, update: actual + update)

## Transform stream to contain only src IPv4 address
communication_peers.map(key: "src IPv4 address", values(1))

## Compute the number of communication peers for the key src IPv4 address
communication_peers.reduceByKey(lambda actual, update: (
            actual(values) + update(values)))

# Join statistics into one stream
joined_streams = flow_ip_total_stats.fullOuterJoin(communication_peers)
```

The input stream of IP flow record is reduced by filtering to only those IP flow records with keys that are required for the computation of the statistics. The stream with keys is used for parallel calculation of the defined statistics, the basic statistics and port statistics, given our sample listing. To compute basic statistics, we first create a map with map key set as a source IP address to distribute the computation on the workers in *Stream4Flow* cluster. The statistics for each RDD are computed using the function `reduceByKey`. Next, the data stream is split into aggregation windows and the host's statistics over a window is computed. The computation of distinct communication peers of a host requires a different approach. First, we create a map that contains a communication pair mapped by the source IP address. Next, we remove duplicity in communication pairs by applying `reduceByKey` function and obtain a stream of unique communication pairs. The stream is divided into observation windows and the duplicity of communication pairs is reduced again. Last, the number of distinct communication pair is computed by computing number of records (communication pairs) for a given source IP address using a combination of `map` and `reduceByKey` functions. The data streams containing the computed characteristics are joined into one stream to aggregate different host statistics in one stream for further processing. Another implemented statistics are computed in parallel data streams analogous to our example.



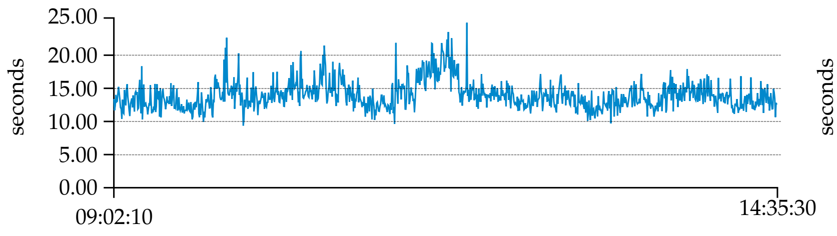
(a) Time series



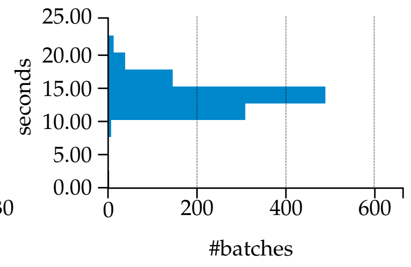
(b) Histogram

Figure 54: IP flow input rates.

We deployed the Host Statistics application in the *Stream4Flow* cluster and assigned it with 8 CPU cores total and 4 GB memory per node (16 GB in total). The application measured all IP flow record observed at the backbone network. The input rate of the IP flow record in an approx. 5 hour long observation sample is depicted in Figure 54. The average input rate in the measured sample was 10 679.51 IP flow records per second with maximum rates over 20 000 IP flow records per second as displayed in the histogram.



(a) Time series



(b) Histogram

Figure 55: Batch processing time.

Analogous to Protocol Statistics application, we measured the time needed for processing a batch of RDDs by Host Statistics. The batch size was set to ten seconds, and the sliding window was set to twenty seconds with a twenty-second slide. The processing time of data contained in a batch is depicted in Figure 55. The average processing time for all batches was 13.960 seconds which is shorter than the size of the sliding window which ensures the long-term stability of the application. We observed the processing times over twenty seconds during our experiment. The increase of the processing time values correlates with the increased IP flow record input rate. When the computation time of a batch was longer than the length of the observation window, a scheduling delay occurs. The scheduling delay means that a batch is not processed immediately as it needs to wait before the processing the proceeding batch is finished. The scheduling delays are depicted in Figure 56. Since the average processing time is shorter than the size of the observation window, the observed scheduling delay vanished in time.

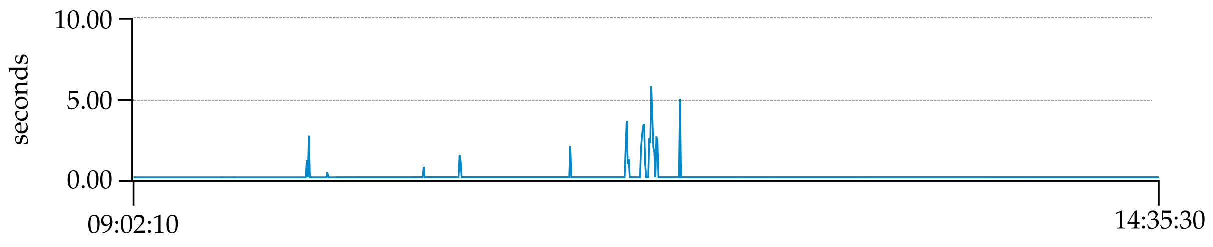


Figure 56: Scheduling delays.

6.1.4 Practical Implications of IP Flow Analysis in Data Streams

Having described stream-based IP flow monitoring architecture, we would like to emphasize the possible benefits and pitfalls in the stream-based processing of IP flows, both in general and those specific to stream processing frameworks. We, moreover, present a recommendation for real-world use-cases of stream-based IP flow processing in cybersecurity. The following information does not serve only for network security administrators, but it can assist anybody who intends to process IP flows or other data in the data stream.

Benefits

A key advantage of stream processing IP flows is the real-time insight into network traffic. Analysis of IP flows can be done over particularly short time windows, which can reveal information, such as bursts of network traffic, that would be lost in aggregation when using the usual five-minute batches. A difference of IP flows statistics computed over short and long time windows is shown in Figure 57. Furthermore, it is also natural to implement sliding windows with slides smaller than the window size in stream processing. This approach allows us to analyze the data and detect network attacks which would be split into two batches in non-stream approaches. Analysis done over a short time window consequently lowers the detection and reaction time to an attack. A detection method reports an attack immediately as it receives the IP flow, which triggers an alert. Therefore, there is no analysis delay as in batch processing. Immediate detection is essential for, among others, DoS attacks or intrusion detection, so that measures to minimize the potential damage by an attacker can be taken instantly.

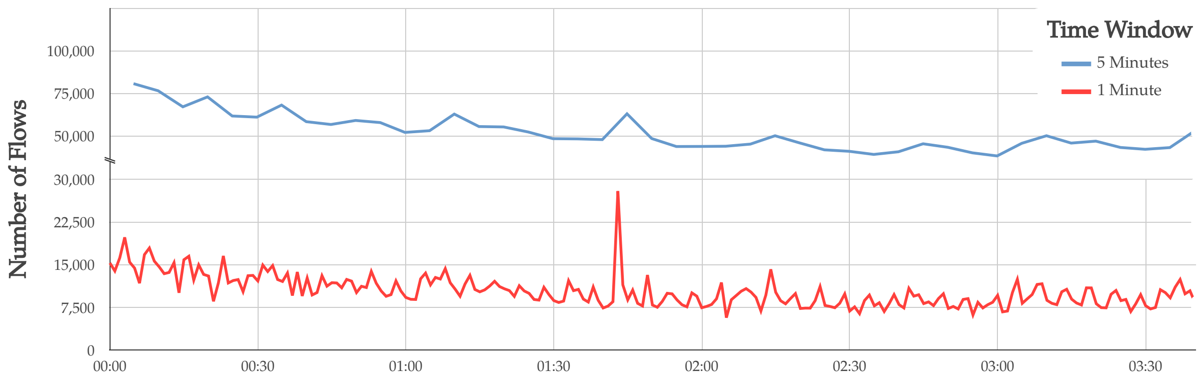


Figure 57: Effects of different time windows used for IP flow aggregation.

Compared to batch-based IP flow processing, stream processing also has an advantage in processing queries. In traditional batch-based approaches, data are first stored on a hard drive and then read from the drive for querying. In stream processing, the data are analyzed in the primary memory (RAM) and are not stored on the hard drive. Stream processing is, then, faster in query processing as no disk I/O operations are needed.

Besides the abovementioned advantages, stream processing frameworks provide two key advantages over contemporary frameworks for IP flow analysis; scalability and a MapReduce programming model. Scalability is ensured by choice of a distributed framework, both for data reception and data processing. If higher performance is needed, a node can be easily added to the framework to boost performance. The distributed character of the systems makes the use of the MapReduce programming model possible. The MapReduce model splits the analysis into parts, and the parts are mapped and distributed to computational nodes. The nodes provide partial results of the analysis, and the results are then reduced to get the final overall result. The distribution of partial computation tasks enables to parallelize and speed up the computation.

Our experience shows that this approach is suitable for computing host statistics as a host's IP address is suitable as a map key. Whereas batch based network monitoring usually analyses a network as a whole, the MapReduce programming model enables us to gather information about all hosts in the network efficiently.

Pitfalls

The time aspect plays an important role in stream processing of IP flows. A timestamp assigned to an IP flow determines its position in the data stream and the order of processing. It also places the IP flow into context. Assigning the timestamp incorrectly may bias flow analysis. However, which timestamp should be assigned to an IP flow may be unclear. An IP flow lasts a portion of time, so there are start and end timestamps for the IP flow. Using either of these timestamps has several pitfalls. First, we need to ensure precise time synchronization (in milliseconds) of all probes in a network to get a properly ordered stream of IP flows. Second, data stream processing frameworks must support external timestamps which is not always true. Last, IP flows from probes may arrive at the stream processing framework unordered due to link latencies, or may be lost during transmission over the network. A suitable substitution to the latter timestamps is to use implicit timestamps (see pp. 127) representing the time when an IP flow is received by the stream processing framework. This timestamp does not require probe time synchronization and external timestamp support, as the timestamp is assigned to an IP flow by the framework. It also ensures the correct order of IP flows as the observation is done in the framework. In all the above cases, we generally need to ensure that timestamps are assigned to IP flows consistently through the whole monitoring infrastructure and that IP flows are properly ordered in a data stream.

Stream processing also changes the nature of the analysis itself. In batch-based IP flow data processing, we were able to perform a query over historical data or search back through raw data for additional information after detecting a successful attack. In the stream-based IP flow analysis, we are not able to analyze data back in time. We can only analyze the data from the time a query arises and after. Moreover, the raw data are not usually stored, only the results of the analysis. This means that the analysis cannot be done ex-post and it must be predefined before the data is analyzed. In other words, we need to know queries over data before we begin to analyze the data in the stream. This fact makes stream processing of IP flows unsuitable for ex-post data analysis. On the other hand, it seems suitable for continual queries, regular reporting, computing statistics, and detection methods. This disadvantage can be solved by using lambda data processing architecture, which combines batch and stream based analysis [256].

As mentioned earlier, IP flows are analyzed over a time window. The effect of processing IP flows in the data stream is a reduction of the time window for IP flow analysis from minutes to seconds. The reduction of the window may influence the results of the analysis, and it is necessary to adapt current data analysis techniques to it. Due to the window reduction, computed statistics may become more volatile, as shown in Figure 57, and prediction techniques may report higher prediction errors. Also, existing reporting and detection thresholds need to be adapted (usually lowered) to the new windows.

Recommendations

We have discussed the possible benefits and pitfalls of stream-based IP flow analysis above. Now, we provide a recommendation for its efficient application in industry. We disclose suggestions for designing stream processing applications and provide suitable practical applications of stream-based IP flow analysis in traffic monitoring for cybersecurity.

The efficient design of applications for stream-based IP flow analysis should properly order the operators over a stream of IP flows. The proper optimization of operators increases the application's performance significantly. High-level techniques for operator optimization are described in [250]. We further recommend embracing the MapReduce programming model [257], which allows the computation tasks to be distributed among more computational nodes. For example, source or destination IP addresses are eligible map keys for computing host statistics. Source or destination ports may also serve for computing network services statistics.

To ease the adoption of stream-based IP flow analysis in cybersecurity, we have provided a list of example suitable stream analysis applications:

- **Real-time attack detection** – the vast majority of batch-based detection methods can be transformed into stream based detection methods. By shortening the time window for analysis and the adapting parameters for detection methods, we can achieve real-time detection, which is desirable in DoS or intrusion detection, among others.

- **Adding context to cybersecurity** – stream-based IP flow processing is suitable for predefined, continuous data analysis. It can be used to supplement traditional network monitoring for pre-processing of additional information needed for network security. The additional information may add context to detection methods, such as the number of total IP flows for a given IP, a list of the top talkers in the network, all counted over several different time periods. The information is immediately accessible on demand, and no demanding queries over historical data are needed.

- **Host IP profiling and trending** – whereas traditional network monitoring analyses the network as a whole and provides network summary statistics, the stream approach is also convenient for efficiently monitoring individual hosts in the network. MapReduce enables to compute characteristics that may represent a host's activity on the network (the number of transferred packets, bytes, communication partners, and so on), and a host's communication profile (such as frequently visited IP addresses, web pages, or active hours in a network). This approach can also be used for creating long-term profiles for all host or services in the network with trend estimation and behavior stability monitoring.

6.1.5 Summary and Outlook

Stream-based IP flow analysis represents a natural complement to currently mainstream batch-based approaches to cybersecurity. It allows security analysts to perform real-time analyses on network data. The stream-based framework benefits from compatibility with current monitoring systems and excels in real-time attack detection, monitoring both network and individual hosts, and providing a context to network security. The presented distributed stream-based framework for IP flow analysis can handle streams of a large volume of data at high speeds and keeps up with the latest network monitoring trends.

Since the volume, speed, and diversity of network traffic will continue to increase in the following years, network monitoring tools should follow this trend. The tools of the future should be able to process traffic at speeds of over 100 Gb/s, gather more information from network traffic (such as service-specific or IoT information), and should natively support a wider variety of formats for exporting IP flow records. In a similar manner to the probes, stream processing systems will have to process more data at higher speeds. This challenge is partially solved by the above-described scalability of the current systems. Nevertheless, we expect optimizations for managing resources more efficiently in memory allocation, or query response time via the use of sketches, for example, or approximative data structures which are supposed to emerge. Moreover, advanced data mining and machine learning methods for intrusion detection are expected to be adapted and natively supported by future data stream systems.

We anticipate the increasing utilization of IP flow network monitoring for both network and host security. With the emergence of new visionary paradigms, such as the Internet of Things, host-based security will become obsolete as it will be impossible to guarantee the proper setting of host security systems for all connected devices. Network traffic measurement, namely IP flow analysis, will become essential for network defense. We believe that the stream-based IP flow analysis is a suitable approach to achieve the next-generation network security.

6.2 Next-generation IP Flow Monitoring

An IP flow network monitoring infrastructure of the future should adapt to emerging trends in IP flow monitoring (see pp. 36), and face the current challenges both of IP flow monitoring (see pp. 47) and IP flow analysis (see pp. 53). The infrastructure of the future should scale easily to store and should be able to process large volume of network traffic, provide sufficient throughput to monitor high-speed networks, should have enough memory and computational power at disposal for advanced analyses, be capable to analyze data in real time, and should respond instantly to operator's queries.

In Chapter 3, we presented a traditional batch-based IP flow monitoring infrastructure. As we showed, the batch-based type of infrastructures suffers from, among other, performance, scalability, and high response time issues. To address these issues, we introduced the real-time IP flow monitoring infrastructure in this chapter. However, our real-time approach also suffers from shortcomings, such as an inability to raise ad-hoc queries, that are present in batch-based infrastructure. We believe that a convenient merge of approaches mentioned above might result in an infrastructure capable of facing a majority of the presented challenges of IP flow monitoring.

In this section, we combine the advantages of batch-based and stream-based approaches and present a network IP flow monitoring infrastructure build upon *lambda architecture* concept. First, we explain the basic principles of the lambda architecture. Next, we map these concepts to the domain of IP flow network monitoring and outline a next-generation IP flow monitoring infrastructure. Then, we discuss the advantages and disadvantages of the proposed architecture.

6.2.1 Basic Principles of Lambda Architecture

The lambda architecture is a generally acknowledged approach for building big data systems proposed by Marz and Warren [256]. The lambda architecture represents a solution with higher performance while it avoids the complexity of fully incremental architectures [256]. The premise behind this architecture is that it is possible to run a query over the whole data. However, it is very demanding regarding computational resources. The main idea is to pre-process the data and provide a set of views, and then query the views [258]. The lambda architecture has already been used in several areas, such as in monitoring the World Wide LHC Computing Grid network traffic [259], Internet of Things and smart city monitoring [260], or in energy consumption monitoring [261].

The high-level schema of the lambda architecture is depicted in Figure 58. The lambda architecture is built as a series of three layers: *batch layer*, *speed layer*, and *serving layer*. The batch layer utilizes advantages of traditional batch-based data management systems, the speed layer takes advantage of stream-based approaches, and the serving layer serves for unified access to previous two layers. We discuss the layers in detail below.

Batch Layer

The primary goal of the batch layer is to produce precomputed views for the queries called *batch views* continuously. The precomputation of the queries reduces the response time of the queries

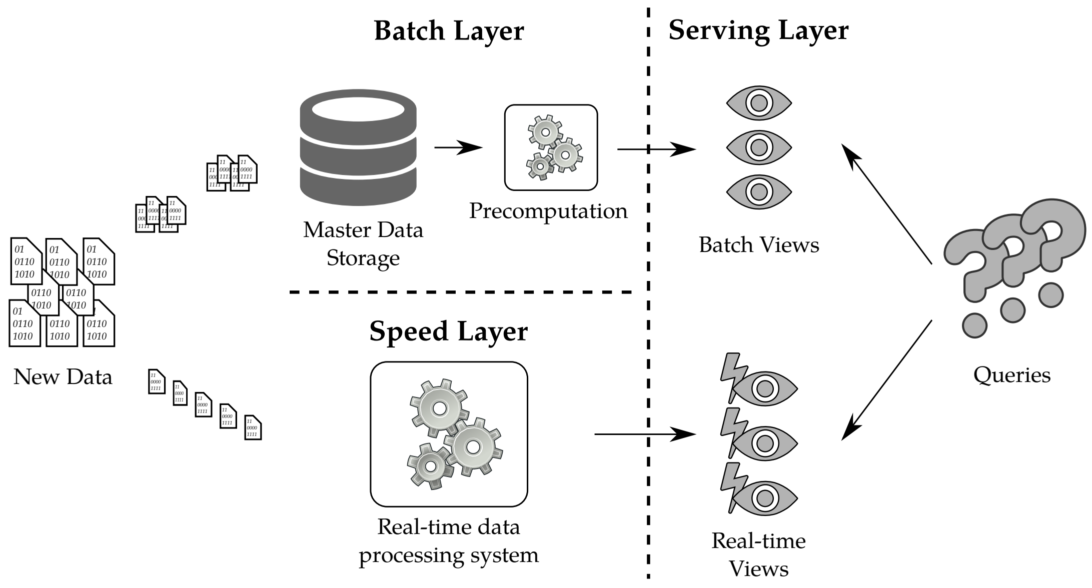


Figure 58: Lambda Architecture Schema *(inspired by [256, 258])*.

as it is not necessary to compute a query on the fly. The precomputed view is indexed and accessible with random reads. A batch view is computed continuously by running a function over the master dataset. However, the computation of a batch view is a high latency operation as all data from master dataset needs to be processed. By the time a function is finished, plenty of new records is already inserted into a master dataset, so a query using the function returns an outdated result. The out-of-the-date result issue is solved by the speed layer, as we show later.

The key performance indicator of the batch layer is how long it takes to update the batch views [256]. Since the speed layer compensates for the latency of batch layer, we need to balance carefully size of the batch view and the time needed to batch creation. The greater a view, the longer time needed for its computation and the larger volume of data is processed in the speed layer. There exist approaches to decrease the latency of batch view creation. An incremental batch processing, where a view is computed based on incremental changes of the master dataset, is worth mentioning.

To fulfill its function, the batch layer needs to be able to store an immutable, constantly growing dataset, and to precompute the views. The layer requires to scale to capture large amounts of data, be distributed to allow big data processing, and fault tolerant on the one hand. On the other hand, the batch layer is forgiving and can recover from errors [258]. Since all data are stored in the master dataset, the views can be recomputed in case of an error. A batch layer is best implemented by using a batch-based distributed data processing systems, such as *Hadoop*, *Apache Spark*, or *Presto*[15].

Speed Layer

Speed layer serves to provide a *real-time view* on the most recent data, that are unavailable in the batch layer. This layer overcomes the out-of-the-date results issue presented in the batch layer. The speed layer aims to achieve the smallest latency possible, to provide the most recent data views. The low-latency and near real-time data processing is possible, as the volume of data processed is small.

The new data are served into a speed layer in the form of continuous data streams. A new incoming data in a data stream is instantly processed, and the real-time view is updated to pro-

15. `https://prestodb.io/`

vide the most recent results. However, the stream-based data processing is much more complex compared to batch-based data processing. As shown in the previous section, several pitfalls need to be overcome, e.g., correct data ordering. Once the batch layer computes a batch view corresponding to a real-time view, the real-time view can be discarded. This property of the Lambda Architecture is called *complexity isolation*, meaning that complexity is pushed into a layer whose results are only temporary [256]. If anything ever goes wrong, you can discard the state for the entire speed layer, and everything will be back to normal within a few hours.

Examples of technologies that are used in the serving layer are frameworks for distributed data stream processing, such as Apache *Spark Streaming*, *Storm*, *Samza*, or *Flink*. The real-time views are typically stored into NoSQL databases [262].

Serving Layer

The serving layer merges the real-time and batch views so that they can be queried in a consistent manner. The serving layer is a specialized distributed database that enables random reads on the views. The serving layer ensures that most up to date results are available. As soon as a new view is available, the serving layer automatically swaps those in and uses the new view to answer a query.

Examples of technologies, that are used in serving layer are *Apache Cassandra*[16], *MongoDB*[17], *Apache Impala*[18] or *ElephantDB*[19].

The above-described layers and their integration naturally results into the following properties of Lambda Architecture [256, 262]:

- **Robustness and fault tolerance** – The system is tolerant to both human and machine failures, as all views can be recomputed from the master dataset in case of failure. The master dataset is stored in a distributed data storage. The distributed data storage uses data replication and partitioning techniques to be able to recover data in case of failure.
- **Scalability** – Tools used in lambda architecture layers are distributed systems that are scalable and can be scaled independently by simply adding a new machine to a cluster.
- **Generalization** – The concept of the architecture is general and can be applied in different areas. We later present its application to IP flow monitoring domain, and cyber situation awareness Domain.
- **Low latency update** – The architecture allows for low latency read while keeping the database robustness.
- **Ad-hoc and continuous queries** – Ad-hoc queries are supported naturally by the batch layer, while the continuous queries are optimal for the speed layer.

6.2.2 Next-generation IP Flow Monitoring Infrastructure

Since the lambda architecture is a general design, it can also be applied to IP flow network monitoring domain. We show, how can be all layers of the lambda architecture adapted to the IP flow monitoring workflow. The IP flow monitoring workflow described in Chapter 3 fits into the description of the batch layer with some extend to serving layer[20]. The real-time IP flow

16. `https://cassandra.apache.org/`
17. `https://www.mongodb.com/`
18. `http://impala.apache.org/`
19. `https://github.com/nathanmarz/elephantdb`
20. For the sake of clarity, we shall use term *batch-based IP flow monitoring* for IP flow monitoring workflow described Chapter 3 through this section.

monitoring that we proposed in Section 6.1 matches the properties of speed layer with some extend to serving layer as well[21]. In the following paragraphs, we discuss the design of the IP flow monitoring infrastructure based on lambda architecture approach in detail.

The IP flow monitoring infrastructure based on the lambda architecture is displayed in Figure 59. Modifications of the IP flow monitoring workflow related to its application to the lambda architecture paradigm applies mainly to the Collection Process of the IP flow monitoring workflow (see pp. 42), and affects the IP flow record analysis. The packet observation, Metering Process, and Exported process remain intact as they are described in Section 3.3. In the batch-based IP flow monitoring workflow, the IP flow record reception and collection are usually closely linked and handled by the same machine - the collector. In the lambda architecture, the IP flow records are served both for immediate storage to a master database located in the batch layer and for the on-the-fly processing performed in the speed layer. Hence, the IP flow reception is detached from the collection and placed into a dedicated device - the *messaging system*.

The messaging system carries out two main functions: IP flow reception and their distribution for the batch and speed layers. To provide its primary functions, a messaging system is expected to meet the following requirements. The system should be able to receive the majority of current IP flow transport protocols (NetFlow v5,9, IPFIX, sFlow, and so forth) from multiple devices simultaneously. Further, the system should provide sufficient throughput to be able to process IP flow records even from high-speed, large-scale networks. Hence, a distributed architecture that provides multiple input and output points and scalability is a suitable choice. Next, the system should be able to connect to and provide data for multiple consumers from speed and batch layers. Besides these main function several optional, but practical, functions of the messaging systems are appreciated: partitioning, and IP flow record preprocessing. The partitioning of the incoming IP flow record into a logic partitions enables to provide a different subset of IP flow records (i.e., partitions) to different data consumers. For example, only IP flows, that originates in the monitored network can be sent to speed layer to provide a real-view for a security operator, while all monitored flows are sent to master storage for data retention. Using data partitions, the data flow can be optimized, and computational costs can be reduced. The other optional function, IP flow record preprocessing, includes, but is not limited to, IP flow record anonymization, and transformation for data serialization format. To the best of our knowledge, we are not aware of any distributed messaging system designed and optimized solely for IP flow record processing. There exist distributed messaging systems designed for distribution of general data, such as Apache *Kafka*, *RabbitMQ* [254], but these messaging systems are not optimized for IP flow record processing and cannot receive the IP flow record transport protocols. Instead, IP flow records need to be transformed to a common data serialization format before they are provided to the messaging system. Moreover, these messaging systems do not provide the additional functions such as IP flow record anonymization or transformation to JSON format. *IPFIXcol* [227] is a non-distributed collector that can serve as an optimized messaging system for IP flow record distribution. It supports the majority of IP flow transport protocols, has intermediate plugins for IP flow record anonymization, and can distribute the IP flow records to the batch and speed layer over a network. It provides a native connector to *FastBit* database that is used for IP flow record storage. Our deployment experience shows that *IPFIXcol* provide sufficient throughput to monitor even high-speed networks of national ISP. *IPFIXcol* also supports a naive partitioning using round-robin distribution of IP flow records among multiple data consumers. The disadvantages of *IPFIXcol* are that it is not distributed and does not provide native connectors to frameworks for distributed data stream processing.

The data handling in the batch layer of the IP flow lambda architecture corresponds to data handling in the batch-based IP flow monitoring workflow. The IP flow record storage used in

21. For the sake of clarity, the term *real-time IP flow monitoring* will be used for IP flow monitoring workflow described in Section 6.1 through this section.

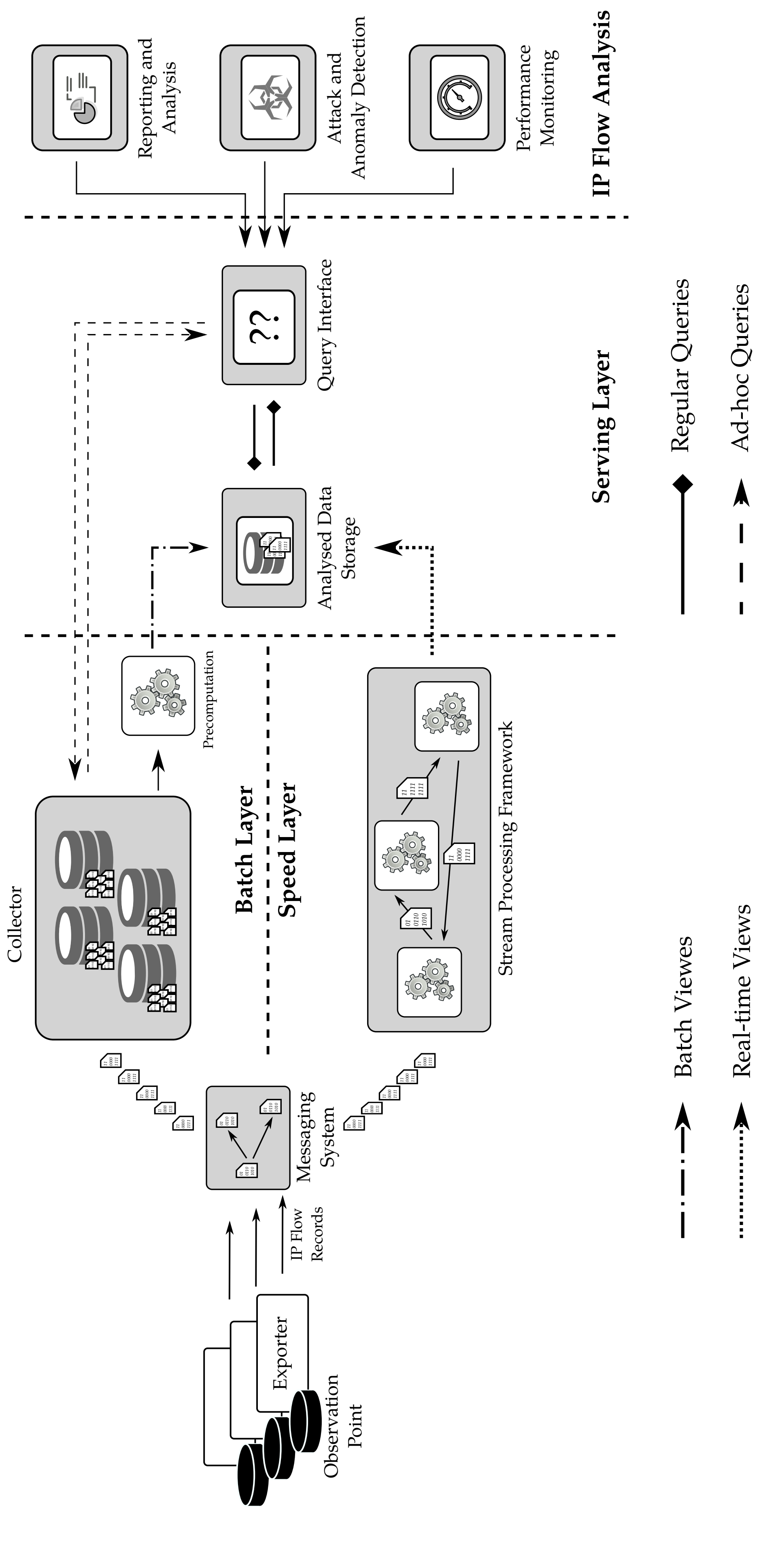


Figure 59: IP flow monitoring infrastructure based on the lambda architecture.

the batch-based workflow fulfills the role of the master storage straightforward. Zadnik, Krobot, and Wrona [133] describe the native *Hadoop*, *Hive*[22] and *Pig*[23] as obsolete technologies from the perspective of IP flow record collector. The authors recognize *Impala* as well as *Vertica* as a feasible solution to store and query flow records as these systems outperform the previous ones if we consider the same hardware setup. The precomputation of the batch-based view is also already present in some of the current tools. For example, *Nfdump tools* allows to create a profile for which all basic characteristics, such as a number of IP flows, packets and bytes are precomputed. A profile can represent a single IP address, network range, network protocol, and so forth. Views for a profile are count, usually every five minutes, and they are stored in a Round Robin Database. The views are mainly used for visualizations of the main profile's characteristics. The distributed data storage mentioned above does not provide the precomputation of IP flow batch views. Nevertheless, the precomputation of a batch view can be implemented as an application. The size of the precomputed batch views can remain the same as the traditional analysis window in the batch-based IP flow monitoring – five minutes. Besides this basic view, the batch layer can compute additional views, e.g., views based on titled windows on a day, week, quarter, or year bases.

The speed layer is based on a distributed system for stream-based data processing as described in Section 6.1. The distributed stream-processing system is connected to the messaging system by one or more connection points. The possibility to connect to a messaging system by more connection points allows for higher throughput. In the IP flow monitoring field, the speed layer serves to fill the five-minute gap introduces in the batch layer. Therefore, the update interval of the real-time views produced by speed layer is required to be shorter than the five-minute update interval of the batch layer. Recently, we observe an efforts to merge include the functionality of the speed layer directly to the messaging systems, e.g., *Kafka Streams* [24]. *Kafka Streams* implements microservices that process the data in the messaging system in real time and allows for the enrichment of the original data directly in the messaging system. The presented advantage of this approach is that it allows running a basic data preprocessing tasks without the need of any additional computational cluster.

The serving layer serves to make the batch and real-time views accessible for a query interface. In the lambda architecture described in the previous section, the real-time view is discarded as soon as a relevant batch view is computed, which means that data representing one information is processed twice – first in a speed view to create a real-time view, and second in the batch layer when a batch view is computed. In our approach, the serving layer keeps the real-time views for specific queries instead of discarding them. These specific queries can be answered from the real-time view and do not have to be recomputed by in the batch layer. On the one hand, this approach puts higher computational demands on the serving layer, as the real-time views often need to be further processed to answer a query. On the other hand, computational power is spared at the batch layer, as batch views do not have to be computed for those specific queries that are processed exclusively by the speed layer. The queries that are suitable for processing solely in the speed layer have the following properties:

- **reasonable size of observation window** – the size of the observation window needs to be suitable for processing in data streams. The size of the observation window along with the volume of network traffic determine the volume of data, that needs to be stored in the primary memory for computation. For example, if we assume an average size of IP flow 150 B, IP flow rate 50 00 flows per second, a one-minute observation window would require to keep 45 MB of data in memory. In the case of one-hour observation window the volume of memory would be in 2.7 GB, and for one day the volume would rise to

22. `https://hive.apache.org/`
23. `https://pig.apache.org/`
24. `https://kafka.apache.org/documentation/streams/`

64.8 GB needed for creating one real-time view. There exist techniques that can reduce the volume of data needed to keep in memory, such as filtering of unnecessary information at the beginning of the analysis process, or computation of the long-term window from partial short-term results.

- **aggregable** – the views that are computed exclusively in speed layer should be suitable for aggregation to allow for creating queries over different time windows. The minimal size of the time window is limited by the frequency of the real-time views. Example of a query that allows for aggregation is a query for a total number of observations. A total number of observations in one hour can be computed as a sum of one-minute real-time views. Top N statistics represents an example of a query, that is not suitable for aggregation, as Top N observations over one day do not correspond to Top N of twenty-four Top N observations over one hour.

In general, we recommend submitting continuous queries that are known in advance and that show the properties described above for exclusive processing in the speed layer and answer these queries using real-time views. Irregular queries or ad-hoc manual analyses should be processed in the batch layer and answered using batch views. A database systems suitable for serving layer are *Elasticsearch* or *Neo4j*.

The main advantage of IP flow monitoring based on the lambda architecture is the reduction of delays introduced by a collection process and a query execution time. The lambda architecture, namely the speed layer enables to obtain query results in near real time reducing the delays to the minimum. It is important to mention that the delays introduced during the exporting and metering processes of IP flows prevail. The term "real-time" refers to the IP flow export time and not to the observation of a packet at the observation point. A further discussion and elaboration on the notion of the term "real-time" in the context of the whole IP flow monitoring workflow is still needed and is left for further research. Another open research opportunity is an assessment of the impact real-time data processing on the information carried in a data. We already identified the increased volatility of a time series measured with the real-time monitoring (see pp. 6.1.4). Other effects of the high granular data on the current IP flow analysis techniques remain unexplored. Last, but not least, the optimization efforts are at place as well, as no lambda-based architecture optimized directly for IP flow measurement is available.

6.3 Toward Real-time Network-wide Cyber Situation Awareness

The application of the cyber situation awareness concept in the network security domain is not straightforward. The efficient application of cyber situation awareness faces several major challenges due to high dynamics of the network environment, speed and volume of network traffic, and complexity of the computer networks. An analyst is overloaded with the raw network data [14]. Volume, velocity, and variability of the raw data prevent him/she them from comprehending a network and taking proper decisions. The speed at which cyber events occur is another challenge. The reaction speed of defenders is much smaller in comparison to the speed of the intruder's actions due to the automation of the attacks. Last, but not least, a challenge is to provide a homogeneous toolset with a unified view both on whole network and individual elements in a network.

CSA systems are multilevel. A CSA system relies on information from intrusion detection systems (IDS), antiviruses, malware detectors, logs, flows and other information sources. This raw information is transformed into events that are further processed [263, A12]. The number of events is however still too high, and their processing is too labor-intensive to be processed manually. The systems for automatic creation of even higher abstractions are needed [14]. The higher abstractions are, for example, graphs of networks with vulnerability dependencies, or decision trees serving as cyber defense support. Prototype systems that are capable of providing the higher abstractions are emerging in literature, e.g., CAULDRON [263], AHEAD [264]. Further, the distillation of valuable information for CSA from network data is a big data problem as the volume of network traffic is increasing on a compound annual growth rate of 21 % every five years [81].

For a full comprehension of a computer network, we need both a big picture of the network (*macro view*) and atomic information about network elements (*micro view*). A macro view provides us with a holistic view of the network, allow us to capture the broader context of events, and see far-reaching consequences of individual actions. The micro view, on the other hand, offers detailed information about an element in a network. Micro view is a cornerstone data for analysis. However, without the macro view, the micro view provides only detailed, context-less information. Distributed denial of service (DDoS), and advanced persistent threat (APT) represent examples that demonstrate a need for both macro and micro views. Individual monitoring of a given server, i.e., micro view, can detect a DDoS on a hosted service. A macro view is applicable when we mitigate the DDoS, e.g., by blocking the traffic on a firewall. The macro view provides information, what is the relevant firewall to drop traffic at and what other services in the network are affected by this precaution. In the APT use-case, a compromise can be detected using a micro view. The trace of the attacker across machines in a network or the point of initial compromise can be detected using the macro view.

This section provides a proposal of the extension of the next-generation IP flow monitoring infrastructure to the cyber situation awareness domain. We identify requirements for an efficient application of cyber situation awareness that respond to challenges mentioned above. Next, we outline a novel framework that meets the identified requirements. The framework reduces data overload through distributed data computing suitable for processing of a large volume data. The speed of the reaction is improved by a merge of a traditional batch-based data analysis and the stream-based approach.

6.3.1 Requirements

A great variety of information is needed to achieve an efficient CSA. Figure 60 summarizes the current state of network perception and comprehension and highlights issues of contemporary CSA.

First, there exist many tools that retrieve data of various types from a network. Each tool usually uses own storage and leverages a tool-specific analysis language and approaches. The tools have different settings of data collection process and storage which influences analysis workflow and data comprehension. Second, raw data is collected at high speeds and volumes. As the tools for data collections differ, the data type, format, frequency, and information they carry differ as well. Moreover, information carried in raw data can overlap, duplicate, or even contradict, which further impedes the analysis. Third, a network administrator has to understand many network monitoring methods and approaches, switch between them for specific information, run different types of analyses, and use various levels of details in an analysis. The diversity of data sources and the complexity of the analysis also hinders the decision process as it takes to an administrator a longer time to collect all data necessary for an informed decision. The result is then a low reaction speed in NwCSA.

To face above mentioned disadvantages and challenges in the perception and comprehension of a network, we impose the following requirements for an effective network-wide CSA:

- **Performance** - the framework should be able to process and analyze large volumes of the data at high speeds.
- **Universality** - the framework should be able to gather and process several data from various data sources.

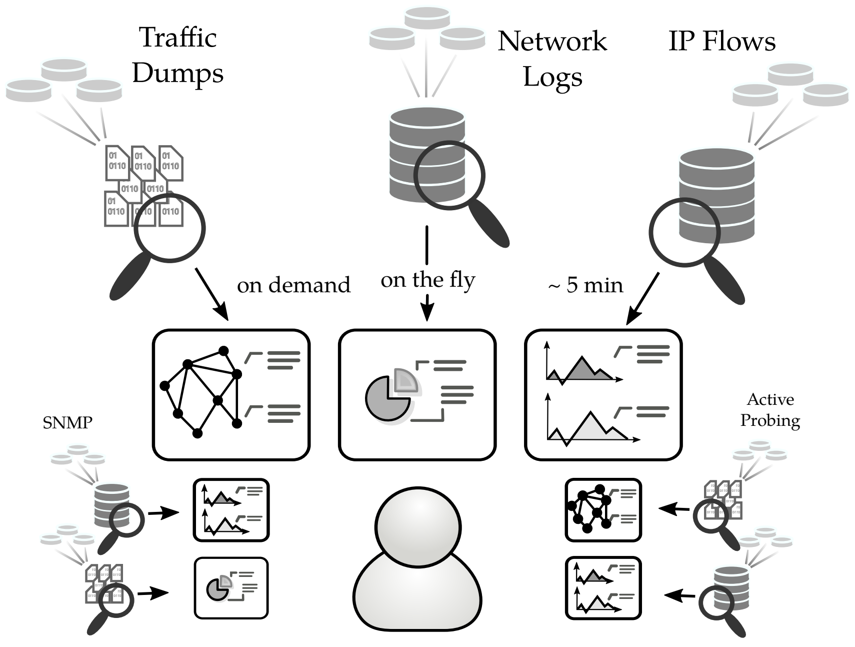


Figure 60: Current state of perception and comprehension of a network.

- **Context** - the framework should be able to offer complete information including context relevant to the information instead overwhelming a user with a flood of raw data.
- **Dynamic level of detail** - the framework should be able to provide a dynamic level of detail both in time and information domain.
- **Reaction time** - the framework should minimize the time needed for analysis to increase the speed of reaction.

In the following section, we apply all requirements and describe a novel framework for a network perception and comprehension in network-wide CSA along with its prototype implementation.

6.3.2 Real-Time Cyber Situation Awareness Framework

The proposed framework for network perception and comprehension leverages new advances achieved in distributed data stream computing and apply their concepts to the network-wide CSA domain. Such an approach results in several improvements in performance, universality, and data analysis.

The proposed framework is depicted in Figure 61. The framework is built upon the lambda architecture for IP flow monitoring described in the previous section. Since the lambda architecture is a general concept, we can abstract from processing only IP flow records and generalize its purpose to processing cybersecurity data related to network-wide CSA. However, the framework, as it is described in the previous section, is not designed to effectively process different types of data from various sources, as it is stated in the universality requirement. Once we generalize its application to any cybersecurity-related data, the framework does not provide means to represent different data in the common format, so the analysis and querying of the different data in the same query are not straightforward possible.

To be able to process data from various probes in one system, the data needs to be normalized into a general data format. Hence, we include a normalization system into the architecture. The

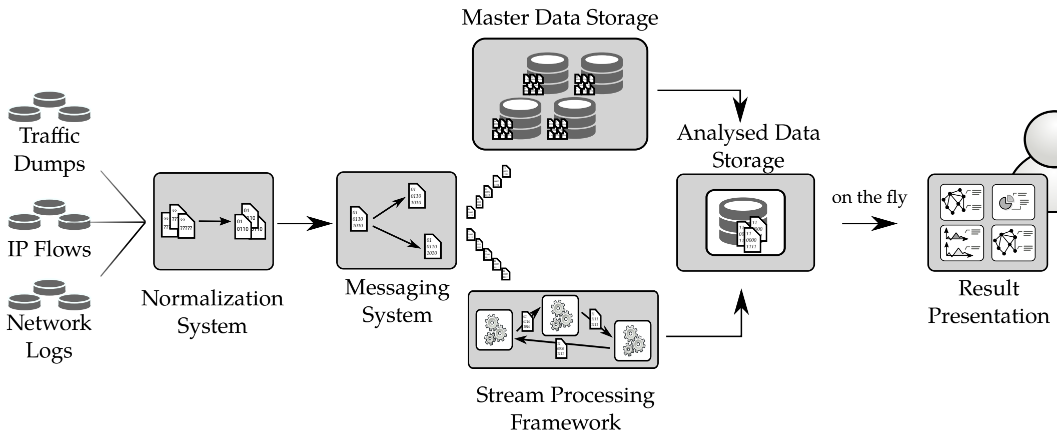


Figure 61: Framework for perception and comprehension of a network.

normalization of the data is a challenging task. Log data, for example, are repeatedly reported to be unstructured and of poor quality. This fact hinders their transformation and unification of the carried information to common data types and formats, that are accessible in common ways. The log normalization is researched by Tovarnak in [265] where the author proposes a log data normalization approach based on prototype-based programming. The approach consists of domain-specific language that can describe log data normalization in an object-oriented manner and of a normalization engine that executes the described normalization. The IP flow records can be normalized using IPFIX templates associated with the given IP flow record. In general, each data source requires a specific codec, i.e., mapping for the carried information, to be able to normalize the information into a common format. Once input data is normalized into a common format, e.g., Data Serialization Format, it is handed over to the messaging system. The messaging system is an entry part of the lambda architecture contained in the framework. The data are processed and queried in the same manner as described above.

Besides the data processing and querying, a layer for the presentation of the analysis results to an operator plays a significant role in the framework. The book collection by Kott, Wang, and Erbacher [14] includes a chapter on visualizations and analysis by Healey et al., where visualizations for cyber situation awareness are discussed in detail. The visualizations supporting CSA are various charts and maps, node-link graphs, timelines that highlight temporal patterns and relationships, treemaps, or hierarchical visualization techniques. Moreover, the authors identify a set of properties that should be reflected by a visualization layer. The layer should fit the analyst's mental model and integrate into the current working environment. The used visualization should be familiar to an analyst and have a gradual learning curve. Last, but not least, the visualization should be scalable in terms of the volume of data to visualize and data retrieval from various sources.

Let's now review each of the stated requirements for a network-wide cyber situation awareness framework. We examine the framework from performance, universality, dynamic level of detail, context, and reaction time point of view. Conclusions of the discussion are based both state-of-the-art findings and own experiences from prototype deployment.

Performance

An efficient framework should be able to process data from variable data sources at high speeds and volumes. Distributed stream processing, distributed master data storage, and system of precomputed views used in framework allow for data processing even in volumes and velocity described above. In the case of need for higher performance, an additional computational node can be added to increase throughput. Moreover, there is no single point of data input that might become a potential bottleneck as messaging systems and normalization systems can be distributed as well. The data are received through multiple input points, which further increases the throughput. It has been shown in [253] that tools for distributed stream data processing are capable of processing data in volumes and speed over 1M records/second in a small cluster of four commodity servers.

The proposed framework also brings performance advantages regarding data analysis. Some advanced analyses, such as data clustering, computation of long-term characteristics, or top N characteristics, are computationally intensive and demand large volumes of memory and computational power. The distributed nature of the stream processing systems allows task parallelization using MapReduce programming principle [257], which makes the computation of such tasks possible. Processing data in data streams also brings a performance advantage. The operations over data streams, such as duplication, split, union, enables us to parallelize the data streams and their analysis. The parallelization further increases the speed and possibilities of the analyses.

Universality

Universality requirement guarantees that a framework based system can receive and analyze data from various types of data sources. The universality is achieved in two steps. The first step is the ability to receive from different data sources. The second step is the ability to analyze various data within one system.

Data sources provide data with different structure and information. A normalization system used in framework maintains universality by using so-called codecs. A codec is a description of a data structure sent by a data source. It specifies the position and meaning of given information in sent data. Normalization tools, such as Logstash, support input codecs for a variety of data sources including a Netflow v5/9 codec for data from IP flow monitoring, *Nmap* codecs for results active network scanning, or *syslog* codecs for machine logs.

Various types of data can are analyzed in the proposed framework due to the unified internal representation of the data. An implemented normalization process transforms the data into readable, comprehensible format. Due to the dynamic and structure of the received data, a general data serialization format (DSF) can be used. Widely popular DSF is Javascript Object Notation (JSON) format. JSON is an open-standard file format that transforms objects into key-value pairs of human-readable text. The DSFs enable to share the data efficiently among the individual components of the framework no matter of the input data format.

Dynamic Level of Detail

Comprehension of a network requires two kinds of dynamic details - a dynamic level of time granularity and dynamic level of view perspective. The proposed framework implements both of the dynamic levels of details.

A challenge of dynamic level of time granularity is to provide short-term results. The long-term statistics can be derived from short-term statistics by aggregation. The short-term results require a data collection and analysis in small time intervals (so-called micro batches). The stream processing approach used in the framework is designed to process and analyze data in micro batches. The micro batches enable us to achieve a required level of minimal time granularity. The stream processing frameworks also implement a concept of sliding windows suitable for long-term analysis. An illustrative example of the advantages of a stream processing framework is an analysis of IP flows. A probe exports IP flows for analysis in a continuous stream. Current tools for IP flow analysis, however, aggregates this continuous stream into batches of five-minute data. The analysis is then done per batch. Using the stream processing frameworks, we can analyze the data per IP flow.

Dynamic level of view represents the possibility to see a network both from overview and in detail. In current network-wide CSA, both views were obtained by a different set of tools specialized for a given view. Providing both views from raw data in a unified tool was too computationally demanding as raw data for both views needed to be processed. Distributed systems and MapReduce programming principle enable such computations, which makes a macro and micro view in a single tool possible. We can assign an entity identifier (e.g., IP address, MAC address) as a mapping key. Computations of required characteristics, statistics and analyses are then distributed among the machines according to the key. Using this approach, we can monitor in detail all entities in a network and, at the same time, monitor the overall network state.

Context

Thanks to the successful implementation of performance and universality requirement, the framework also meets the context requirement. The universal nature of the framework enables us to process data from various sources. Scalability and distributed nature of the framework guarantees a sufficient computational power to process and analyze the data. Thus, all necessary raw data needed for CSA can be managed in one system.

The possibility of processing all types of data in one framework enables us to combine pieces of information from different data sources effectively. Information about a host network behavior gained from IP flow analysis can be supported by relevant log records from the host. The results of analyses of different data sources can be correlated which each other to increase the precision and robustness of the analyses.

Data stream processing systems are suitable for data preprocessing as they implement continuous queries. Using continuous queries, we can precompute several predefined characteristics and statistics. The administrator does not need to run analyses over collected raw data. Instead, the analyses are run on preprocessed characteristics. This approach reduces the data overload in CSA as an administrator handles preprocessed data instead of raw data.

Reaction time

Proposed framework improves the reaction time in two ways: it reduces analysis response time, and it enables a real-time data processing. Both enhancements shorten a time needed for delivery of information necessary for a decision to an administrator, which decreases the reaction time.

The analysis response time is reduced due to the use of distributed systems for data processing. As described earlier, distributed systems enable parallelization of analysis computation, e.g., by using the MapReduce programming model. An analysis can then finish in a shorter time and results of the analysis are available earlier. The analysis response time is also improved by a data preprocessing done by the stream processing system. The analysis process does not need to process a large volume of raw data. Instead, some partial results are precomputed, and only a reduced volume of data is analyzed.

Data stream processing systems included in the framework can process data in real time. A piece of raw information is processed immediately when a stream processing system receives it. Real-time data processing is a significant advantage over current tools for network perception. As described earlier, there are delays caused by the analysis of the data in batches. In the case of IP flows, the delay can reach up to 10 minutes. Including systems capable of real-time data processing into the CSA framework eliminates such delays and improves the reaction time.

Further Remarks

Support of various data sources and processing different data in one system opens a new issue regarding information duplication. In a real-world deployment, the data collection area of the probes can overlap. Two separate probes can then observe the same information. A suitable example is an IP flow that is routed via two probes or the log records from two different machines that represent the same network scan. Duplication of collected information needs to be kept in mind during data analysis and comprehension. In case information is not deduplicated, biased or incorrect findings may occur. The bias is then carried further in CSA framework, and misleading decisions are taken.

6.4 Summary

This chapter presents our effort to achieve real-time cyber situation awareness. We present a stream-based approach to the IP flow analysis that increases the speed of analysis and allows for high-frequency data analysis. Our approach leverages the advantages in the area of distributed data stream computing. We apply the distributed data stream processing concept to the IP flow network monitoring domain and discuss the implication of the shift from traditional batch-based IP flow record processing paradigm to the distributed data stream paradigm. To demonstrate the advantages of the stream-based IP flow record processing, we develop a framework for rapid prototyping of the stream-based IP flow analyses *Stream4Flow*.

Based on our previous research, we propose a next-generation IP flow monitoring workflow, that merges the traditional batch-based IP flow record analysis with the stream-based approach. The merge is based on the *lambda architecture*, a general architecture for big data processing. We present the basics of the lambda architecture and apply the described concepts to the IP flow network monitoring domain. The next-generation IP flow monitoring infrastructure is expected to be able to process big data at high speeds and to provide a near real-time analysis results to an operator.

The next-generation IP flow monitoring infrastructure is generalized, and we outline a framework for real-time cyber situation awareness. We describe a normalization component, that is a necessary prerequisite for the unified cybersecurity data processing. The cyber situation awareness framework is discussed with respect to the predefined requirements imposed on the effective cyber situation awareness framework.

The main contributions of this chapter are:

- design and evaluation of the stream-based IP flow monitoring workflow,
- identification of the pitfalls of the stream-based approach to IP flow record analysis,
- design of the next-generation IP flow network monitoring,
- design and discussion on the real-time cyber situation awareness framework.

7 Conclusion

Computer networks are continuously evolving in time. The volume of data transferred over a network increases, the speed of the data transfer rises as well. They increase in sophistication and complexity and paradigms of network usage change as demonstrated by the transition to IPv6 addressing or by the rising share of encrypted traffic, for example. Methods for obtaining cyber situation awareness need to keep up with these changes to be able to deliver relevant information for cybersecurity operators. In this thesis, we improve the cyber situation awareness by the research focused on the improvement of network IP flow monitoring. We define the main objective to *investigate, how IP flow monitoring can be improved to enhance the cyber situation awareness*. The paragraphs below summarize our contributions to the IP flow monitoring and the cyber situation awareness achieved in the course of our research.

7.1 Research Goals

We have defined research goals to be able to systematically address the identified joint open issues of the IP flow monitoring and the cyber situation awareness – the lack of visibility and comprehension of a network, host identification in network traffic and delays presented in the IP flow monitoring workflow. We describe our contributions to each of the research goals below.

RG1: *Propose and evaluate IP flow monitoring methods that enhance network perception and comprehension and respond to the emerging trends in the cyber situation awareness and the IP flow monitoring.*

We have addressed this research goal for network perception and network comprehension separately. As for the network perception, we analyzed how we can enhance the information value of IP flow records by adding information from the application layer of network traffic, and what impact on the IP flow monitoring the addition of application information makes. We proposed parser for monitoring of HTTP network traffic that enables to obtain information for HTTP protocol header fields, such as domain name, User-Agent, or referer. We evaluated the after-effects of the application information retrieval. We conducted experiments that showed the trade-off between the volume of exported information and the parser's performance.

Our contribution to network comprehension has been represented by investigation of tunneling of IPv6 network traffic over IPv4 networks using transition mechanisms as 6to4 and Teredo and by a description of Top N statistics. We performed measurements that revealed characteristics of both tunneling and tunneled traffic. We described the distributions of Time to Live and HOP values, IP flow characteristics, and locations of Teredo endpoint servers. We briefly discussed the rate of IPv6 adoption. As for the Top N statistics research, we provided a detailed definition of the statistics including the effects of settings of its optional parameters. Among others, we investigated the availability and the stability of the statistics in time.

RG2: *Develop methods for host identification in both unencrypted and encrypted network traffic.*

The identification of a host from the IP flow records is a challenging task since a host is represented by an IP address. Such an identification is not unambiguous due to dynamic networking, or network address translation, for example. Therefore, supporting mechanisms for host identification are required. First, we investigated the possibility of host information retrieval from encrypted network traffic. We described an approach to SSL/TLS traffic measurement and showed, how to build a dictionary of cipher suite lists – User-Agent pairs. The dictionary allows for gaining additional host-related information even from encrypted network traffic. Next, we researched, whether Top N statistics is a suitable candidate to be the supporting mechanism for the host identification. We discussed the suitability of Top N communication peers, ports, and HTTP domains from both theoretical and experimental point of view. We showed that Top N statistics has a limited application for host identification problem as only 60 % of hosts in network traffic could be identified.

Third, we examined the retrieval of the information about the operating system of a host. We developed enhanced methods for OS identification both in static and dynamic networks. For the static networks, we introduced a novel method based on OS-specific domains and we compared our approach with other two state-of-the-art approaches to OS fingerprinting. We also described the current open challenges for OS identification. Last, we focused on host-based information retrieval from encrypted traffic. On real-world measurements, we demonstrated the possibilities of client identification, browser fingerprinting, and network traffic classification using the developed dictionary of cipher suite lists – User-Agent pairs.

RG3: *Provide an option for reducing the delays in the network IP flow monitoring workflow leading to the real-time cyber situation awareness.*

We started with identification and description of delays present in the IP flow monitoring workflow. To increase the speed of IP flow monitoring workflow, we chose to focus on the delays that occur during the collection and analysis processes. We proposed a stream-based IP flow monitoring workflow that builds upon the latest advances in distributed data stream processing and enables IP flow record analysis in real time. We implemented a prototype for stream-based IP flow monitoring *Stream4Flow*, demonstrated its performance characteristics, and described advantages and pitfalls of the stream-based approach. Further, we proposed a next-generation IP flow monitoring infrastructure that combines the benefits of the stream-based and the traditional batch-based approaches. We show that the next-generation infrastructure can be used, with only minor modifications, as a tool for obtaining a real-time cyber situation awareness.

7.2 Further Research

Although we significantly contributed to several open issues of cyber situation awareness, there are still opportunities for challenging and interesting future research. The evolution of computer networks is a never-ending process, and the methods for obtaining cyber situation awareness needs to keep up with this evolution. We present directions of possible future research of IP flow monitoring and cyber situation awareness relevant to our research:

- **Attack Prediction** – The prediction level of the cyber situation awareness is out of the scope of this thesis. Nevertheless, it still represents a challenging open issue that is left to be addressed. One of the main goals of the prediction level is to gain an ability to predict the next step of an attacker based on its past actions. There already exist methods for attack prediction. However, human experts still play a vital role in this process. The automation of the prediction process, its adaptation to new paradigms such as the Internet of Things or software-defined networking, and improved accuracy of the prediction remain a challenge.

- **Stream-based data mining** – Data mining algorithms for batch-based or static analyses are well established. These algorithms cannot, however, be used for data stream analysis straightforward. The algorithms need to be modified to enable continuous data processing and update of the analysis results and models. Further, the data mining methods need to be optimized to reduce analysis time and provide results and updates instantly, since the stream-based approach is used mainly for real-time analyses.
- **Correlation of data sources** – As described in Section 6.2, it is possible to combine information from different data sources, e.g. computer logs and IP flows. The research of the possibilities for information correlation from these sources can discover new, more accurate and robust methods for attack and anomaly detection. Moreover, the correlation of the computer logs with IP flow could be used to provide the ground truth for IP flows that is still missing in the research of IP flow based detection methods. The knowledge of the ground truth opens possibilities for rigorous and reasonable utilization of machine learning and artificial intelligence methods in IP flow monitoring domain.
- **Host trustworthiness** – Once we obtain a host-based view on a network, we can extend our research by providing a cybersecurity operator with a simple measure of the security risk related to a host – host trustworthiness. The trustworthiness represents an individual's view of an entity's character or standing. One challenge is to define the concept of trustworthiness based on information from IP flows. Another challenge is to determine host features, which are relevant for estimating trustworthiness. We need to determine the class of a feature (e.g., continuous, static, sequences), type (e.g., number of communicating pairs, operation systems, used ports), granularity (e.g., average per day, median per hour). Last, but not least, it is necessary to find a suitable model for estimating host trustworthiness and define a methodology for its evaluation. The host trustworthiness would provide a simplified, initial view for a cyber operator, that would allow identification of hosts worth close attention.

Bibliography

Authored publications referenced in the thesis

[A1] T. Jirsik and P. Celeda. “Toward Real-time Network-wide Cyber Situational Awareness”. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, 2018, p. 7. doi: `10.1109/NOMS.2018.8406166` (see pp. 9, 123).

[A2] T. Jirsik. “Stream4Flow: Real-time IP Flow Host Monitoring using Apache Spark”. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, 2018, p. 2. doi: `10.1109/NOMS.2018.8406132` (see pp. 9, 123).

[A3] T. Jirsik, M. Cermak, D. Tovarnak, and P. Celeda. “Toward stream-based IP flow analysis”. In: *IEEE Communications Magazine* 55.7 (2017), pp. 70–76. issn: 01636804. doi: `10.1109/MCOM.2017.1600972` (see pp. 58, 123).

[A4] P. Velan, T. Jirsik, and P. Celeda. “Design and evaluation of HTTP protocol parsers for IPFIX measurement”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8115 LNCS. Springer Berlin Heidelberg, 2013, pp. 136–147. isbn: 978-3-642-40552-5. doi: `10.1007/978-3-642-40552-5_13` (see pp. 63, 78, 87).

[A5] M. Husak, M. Cermak, T. Jirsik, and P. Celeda. “HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting”. In: *Eurasip Journal on Information Security* 2016.1 (Feb. 2016), pp. 1–14. issn: 2510523X. doi: `10.1186/s13635-016-0030-7` (see pp. 63, 87, 89).

[A6] M. Husak, M. Cermak, T. Jirsik, and P. Celeda. “Network-based HTTPS client identification using SSL/TLS fingerprinting”. In: *Proceedings - 10th International Conference on Availability, Reliability and Security, ARES 2015*. IEEE Press, 2015, pp. 389–396. isbn: 9781467365901. doi: `10.1109/ARES.2015.35` (see pp. 63, 87, 89, 92).

[A7] T. Jirsik and P. Celeda. “Identifying Operating System Using Flow-Based Traffic Fingerprinting”. In: *Advances in Communication Networking*. Cham: Springer International Publishing, 2014, pp. 70–73. isbn: 978-3-319-13488-8 (see pp. 77, 85, 89).

[A8] T. Jirsik, M. Cermak, and P. Celeda. “On information value of top N statistics”. In: *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*. 2016, pp. 1–5. isbn: 9781509037643. doi: `10.1109/ICITCS.2016.7740357` (see p. 89).

[A9] M. Elich, P. Velan, T. Jirsik, and P. Celeda. “An Investigation into Teredo and 6to4 Transition Mechanisms: Traffic Analysis”. In: *38th Annual IEEE Conference on Local Computer Networks - Workshops*. IEEE Press, 2013, pp. 1018–1024. doi: `10.1109/LCNW.2013.6758546` (see p. 89).

[A10] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky. “Passive OS Fingerprinting Methods in the Jungle of Wireless Networks”. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, Apr. 2018, pp. 1–9. isbn: 978-1-5386-3416-5. doi: `10.1109/NOMS.2018.8406262` (see p. 89).

[A11] M. Cermak, T. Jirsik, and M. Lastovicka. “Real-time analysis of NetFlow data for generating network traffic statistics using Apache Spark”. In: *Proceedings of the NOMS 2016 - 2016*

IEEE/IFIP Network Operations and Management Symposium. IEEE Press, 2016, pp. 1019–1020. ISBN: 9781509002238. DOI: `10.1109/NOMS.2016.7502952` (see p. 123).

[A12] M. Drasar, T. Jirsik, and M. Vizvary. "Enhancing network intrusion detection by correlation of modularly hashed sketches". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8508 LNCS. Springer, Berlin, Heidelberg, 2014, pp. 160–172. ISBN: 9783662438619. DOI: `10.1007/978-3-662-43862-6_19` (see p. 147).

Other publications referenced in the thesis

[1] R. D. Gilson. "Special issue preface". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (Mar. 1995), pp. 3–4 (see p. 10).

[2] N. A. Stanton, P. R. Chambers, and J. Piggott. "Situational awareness and safety". In: *Safety Science* 39.3 (2001), pp. 189–204. ISSN: 09257535. DOI: `10.1016/S0925-7535(01)00010-8` (see pp. 10–13, 15, 16).

[3] N. B. Sarter and D. D. Woods. "Situation Awareness: A Critical But Ill-Defined Phenomenon". In: *The International Journal of Aviation Psychology* 1.1 (1991), pp. 45–57. ISSN: 1050-8414. DOI: `10.1207/s15327108ijap0101_4` (see pp. 10, 14, 15).

[4] L. Goodstein, H. Andersen, and S. Olsen. "Tasks, errors, and mental models". In: *Knowledge-Based Systems*. Vol. 1. 4. Bristol, PA, USA: Taylor & Francis, Inc., 1988. Chap. Coping wit, p. 253. ISBN: 0850664012. DOI: `10.1016/0950-7051(88)90045-7` (see pp. 10, 11).

[5] J. Reason. *Human error*. Cambridge university press, 2007. ISBN: 978-0-521-30669-0 (see p. 10).

[6] M. R. Endsley. "Toward a Theory of Situation Awareness in Dynamic Systems". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), pp. 32–64. ISSN: 0018-7208. DOI: `10.1518/001872095779049543` (see pp. 10, 14, 16–18, 21).

[7] K. Smith and P. A. Hancock. "Situation Awareness Is Adaptive, Externally Directed Consciousness". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), pp. 137–148. ISSN: 00187208. DOI: `10.1518/001872095779049444`. arXiv: `arXiv:1011.1669v3` (see p. 10).

[8] G. Bedny and D. Meister. "Theory of Activity and Situation Awareness". In: *International Journal of Cognitive Ergonomics* 3.1 (1999), pp. 41–46. ISSN: 1088-6362. DOI: `10.1207/s15327566ijce0301_5` (see pp. 10, 12, 13).

[9] M. R. Endsley. "Measurement of Situation Awareness in Dynamic Systems". In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* 37.1 (1995), pp. 65–84. ISSN: 0018-7208. DOI: `10.1518/001872095779049499` (see p. 10).

[10] U. Neisser. *Cognition and reality : principles and implications of cognitive psychology*. W.H. Freeman, 1976, p. 230. ISBN: 0716704773 (see pp. 11, 12).

[11] M. R. Endsley. "Situation awareness global assessment technique (SAGAT)". In: *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*. IEEE, 1988, pp. 789–795. DOI: `10.1109/NAECON.1988.195097` (see pp. 13, 14).

[12] M. R. Endsley. "Situation awareness: operationally necessary and scientifically grounded". In: *Cognition, Technology & Work* 17.2 (May 2015), pp. 163–167. ISSN: 1435-5558. DOI: `10.1007/s10111-015-0323-5` (see pp. 13, 14).

[13] M. R. Endsley. "Theoretical underpinnings of situation awareness: A critical review". In: *Proceedings of the International Conference on Analysis and Measurement of Situation Awareness*. 1995, p. 24 (see pp. 14–17).

[14] A. Kott, C. Wang, and R. F. Erbacher. *Cyber Defense and Situational Awareness*. Vol. 62. Springer International Publishing, 2015, p. 329. ISBN: 978-3-319-11390-6. DOI: `10.1007/978-3-319-11391-3` (see pp. 14, 18–20, 22–25, 27, 147, 150).

[15] D. G. Jones and M. R. Endsley. "Sources of situation awareness errors in aviation". In: *Aviation Space and Environmental Medicine* 67.6 (June 1996), pp. 507–512. ISSN: 00956562. DOI: 10.1039/c4qo00187g (see pp. 14, 15).

[16] J. M. Flach. "Situation Awareness: Proceed with Caution". In: *Human Factors* 37.1 (Mar. 1995), pp. 149–157. ISSN: 00187208. DOI: 10.1518/001872095779049480 (see p. 15).

[17] J. Boyd. "The Essence of Winning and Losing". In: *A Discourse on Winning and Losing* May (1987), p. 5 (see p. 18).

[18] P. Barford, M. Dacier, T. G. Dietterich, M. Fredrikson, J. Giffin, S. Jajodia, S. Jha, J. Li, P. Liu, P. Ning, X. Ou, D. Song, L. Strater, V. Swarup, G. P. Tadda, C. Wang, and J. Yen. "Cyber SA: Situational awareness for cyber defense". In: *Advances in Information Security* 46 (2010), pp. 3–13. ISSN: 15682633. DOI: 10.1007/978-1-4419-0140-8_1 (see pp. 18, 23).

[19] J. Moffat. "Mathematical modelling of information age conflict". In: *Journal of Applied Mathematics and Decision Sciences* 2006 (July 2006), pp. 1–15. ISSN: 11739126. DOI: 10.1155/JAMDS/2006/16018 (see p. 19).

[20] Department of the Army. *FM 3-38: Cyber Electromagnetic Activities*. 1. 2014, p. 96. ISBN: 9788578110796. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3 (see p. 20).

[21] J. Brynielsson, U. Franke, and S. Varga. "Cyber Situational Awareness Testing". In: *Combatting Cybercrime and Cyberterrorism*. Springer, Cham, 2016, pp. 209–233. ISBN: 978-3-319-38930-1. DOI: 10.1007/978-3-319-38930-1_12 (see pp. 20, 23).

[22] W. Gibson. "Neuromancer". In: *Neuromancer*. Ace Books, 1983, p. 276. ISBN: 0006480411 (see p. 20).

[23] L. Strate. "The varieties of cyberspace: Problems in definition and delimitation". In: *Western Journal of Communication* 63.3 (1999), pp. 382–412. ISSN: 17451027. DOI: 10.1080/10570319909374648. arXiv: arXiv:1011.1669v3 (see p. 20).

[24] D. T. Nguyen and J. Alexander. "The Coming of Cyberspacetime and the End of the Polity". In: *Cultures of Internet: virtual spaces, real histories, living bodies* (1996), pp. 99–124 (see p. 20).

[25] R. Gozzi. "The Cyberspace Metaphor". In: *ETC: A Review of General Semantics* 51.2 (1994), pp. 218–223. ISSN: 0014164X, 21689245 (see p. 20).

[26] M. Albanese and S. Jajodia. "Formation of awareness". In: *Advances in Information Security* 62 (2014), pp. 47–62. ISSN: 15682633. DOI: 10.1007/978-3-319-11391-3_4 (see pp. 21, 23).

[27] Commonwealth of Australia. *Cyber Security Strategy*. 2009, p. 38. ISBN: 9781921241994. DOI: 978-1-921241-99-4 (see p. 22).

[28] B. Obama. *International Strategy for Cyberspace*. 2011, p. 26 (see p. 22).

[29] Federal Ministry of the Interior. *Cyber Security Strategy for Germany*. 2011, pp. 1–20 (see p. 22).

[30] HM Government. *National Cyber Security Strategy 2019-2020*. 2016, p. 43 (see p. 22).

[31] Government of Canada. *Canada' s Cyber Security Strategy: For a Stronger and More Prosperous Canada*. Tech. rep. 2010, p. 17. DOI: 78-1-100-16934-7. URL: http://publications.gc.ca/collections/collection%7B%5C_%7D2010/sp-ps/PS4-102-2010-eng.pdf (see p. 22).

[32] National Security Authority. *National Cyber Security Strategy of the Czech Republic for years 2015 – 2020*. Prague, 2015, p. 23 (see p. 22).

[33] P. Liu, S. Jajodia, and C. Wang. *Theory and Models for Cyber Situation Awareness*. Vol. 10030. 2017, p. 227. ISBN: 978-3-319-61151-8. DOI: 10.1007/978-3-319-61152-5 (see p. 22).

[34] U. Franke and J. Brynielsson. "Cyber situational awareness – A systematic review of the literature". In: *Computers & Security* 46 (2014), pp. 18–31. ISSN: 01674048. DOI: 10.1016/j.cose.2014.06.008 (see pp. 22, 23).

[35] S. Jajodia, P. Liu, V. Swarup, and C. Wang. *Cyber situational awareness: Issues and Research*. 1st ed. Springer Science+Business Media, 2010, p. 252. ISBN: 978-1-4419-0139-2. DOI: 10.1007/978-1-4419-0140-8 (see p. 22).

[36] I. Friedberg, F. Skopik, and R. Fiedler. "Cyber situational awareness through network anomaly detection: state of the art and new approaches". In: *e & i Elektrotechnik und Informationstechnik* 132.2 (Mar. 2015), pp. 101–105. ISSN: 0932-383X. DOI: 10.1007/s00502-015-0287-4 (see pp. 23, 24).

[37] I. A. Cooke, A. Scott, K. Sliwinska, N. Wong, S. V. Shah, J. Liu, and D. Schuster. "Toward Robust Models of Cyber Situation Awareness". In: *Advances in Intelligent Systems and Computing*. Vol. 782. Springer, Cham, July 2018, pp. 127–137. ISBN: 9783319947815. DOI: 10.1007/978-3-319-94782-2_13 (see p. 23).

[38] W. Heinbockel, S. Noel, and J. Curbo. "Mission Dependency Modeling for Cyber Situational Awareness". In: *NATO IST-148 Symposium on Cyber Defence Situation Awareness*. 2016, pp. 1–14 (see p. 23).

[39] M. Albanese and S. Jajodia. "A Graphical Model to Assess the Impact of Multi-Step Attacks". In: *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* (Apr. 2017), pp. 1–15. DOI: 10.1177/1548512917706043 (see p. 23).

[40] S. Jajodia, S. Noel, P. Kalapa, B. C. O'berry, M. A. Jacobs, E. B. Robertson, and R. G. Weierbach. *Network attack modeling, analysis, and response*. 2011 (see p. 23).

[41] R. Ganesan, A. Shah, S. Jajodia, and H. Cam. "A Novel Metric for Measuring Operational Effectiveness of a Cybersecurity Operations Center". In: *Network Security Metrics*. Cham: Springer International Publishing, 2017, pp. 177–207. ISBN: 978-3-319-66505-4. DOI: 10.1007/978-3-319-66505-4_8 (see p. 23).

[42] S. Noel, S. Jajodia, L. Wang, and A. Singhal. "Measuring Security Risk of Networks Using Attack Graphs". In: *International Journal of Next Generation Computing* 1.1 (2010), pp. 135–147. ISSN: 0976-5034 (see p. 23).

[43] S. Noel and S. Jajodia. "Metrics suite for network attack graph analytics". In: *Proceedings of the 9th Annual Cyber and Information Security Research Conference on - CISR '14*. New York, New York, USA: ACM Press, 2014, pp. 5–8. ISBN: 9781450328128. DOI: 10.1145/2602087.2602117 (see p. 23).

[44] S. Noel and S. Jajodia. "A Suite of Metrics for Network Attack Graph Analytics". In: *Network Security Metrics*. Cham: Springer International Publishing, 2017, pp. 141–176. ISBN: 978-3-319-66505-4. DOI: 10.1007/978-3-319-66505-4_7 (see p. 23).

[45] L. Wang, M. Albanese, and S. Jajodia. "Attack Graph and Network Hardening". In: *Network Hardening*. Cham: Springer International Publishing, 2014, pp. 15–22. ISBN: 978-3-319-04611-2. DOI: 10.1007/978-3-319-04612-9. URL: https://doi.org/10.1007/978-3-319-04612-9%7B%5C_%7D3 (see p. 23).

[46] M. Albanese, S. Jajodia, and S. Noel. *Methods and systems for determining hardening strategies*. 2015. URL: https://www.google.com/patents/US9203861 (see p. 23).

[47] S. Jajodia, V. S. Subrahmanian, V. Swarup, and C. Wang. *Cyber deception: Building the scientific foundation*. Springer International Publishing, 2016, pp. 1–312. ISBN: 9783319326993. DOI: 10.1007/978-3-319-32699-3 (see p. 23).

[48] S. Jajodia, N. Park, F. Pierazzi, A. Pugliese, E. Serra, G. I. Simari, and V. S. Subrahmanian. "A Probabilistic Logic of Cyber Deception". In: *IEEE Transactions on Information Forensics and Security* 12.11 (2017), pp. 2532–2544. ISSN: 15566013. DOI: 10.1109/TIFS.2017.2710945 (see p. 23).

[49] M. Albanese, E. Battista, and S. Jajodia. "A Deception Based Approach for Defeating OS and Service Fingerprinting". In: *IEEE Conference on Communications and Network Security (CNS)*. FLorence, Italy: IEEE, 2015, pp. 317–325. DOI: 10.1109/CNS.2015.7346842 (see p. 23).

[50] S. Jajodia and M. Albanese. "An integrated framework for cyber situation awareness". In: *Theory and Models for Cyber Situation Awareness. Lecture Notes in Computer Science*. Vol. 10030. Springer, Cham, 2017, pp. 29–46. ISBN: 9783319611525. DOI: 10.1007/978-3-319-61152-5_2 (see p. 23).

[51] A. Natarajan, P. Ning, Y. Liu, S. Jajodia, and S. E. Hutchinson. "NSDMiner: Automated discovery of network service dependencies". In: *Proceedings - IEEE INFOCOM*. IEEE, 2012, pp. 2507–2515. ISBN: 9781467307758. DOI: 10.1109/INFCOM.2012.6195642 (see p. 23).

[52] A. Kott. "Towards fundamental science of cyber security". In: *Advances in Information Security* 55 (2014), pp. 1–13. ISSN: 15682633. DOI: 10.1007/978-1-4614-7597-2_1 (see p. 23).

[53] A. Kott. "Science of Cyber Security as a System of Models and Problems". In: Lemnios 2011 (Nov. 2015). arXiv: 1512.00407 (see p. 23).

[54] *Cyber-security of SCADA and Other Industrial Control Systems*. Vol. 66. Advances in Information Security. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-32123-3. DOI: 10.1007/978-3-319-32125-7 (see p. 23).

[55] Z. A. Collier, M. Panwar, A. A. Ganin, A. Kott, and I. Linkov. "Security Metrics in Industrial Control Systems". In: *Cyber-security of SCADA and Other Industrial Control Systems, Advances in Information Security*. Vol. 66. Springer, Cham, 2016, pp. 167–185. ISBN: 978-3-319-32123-3. DOI: 10.1007/978-3-319-32125-7 (see p. 23).

[56] E. Colbert, D. T. Sullivan, and A. Kott. "Cyber Wargaming on SCADA Systems". In: *Proceedings of the 12th International Conference on Cyber Warfare and Security, ICCWS 2017*. Academic Conferences and publishing limited. 2017, pp. 96–104. ISBN: 9781911218258. DOI: 9781911218265 (see p. 23).

[57] I. Linkov, D. A. Eisenberg, K. Plourde, T. P. Seager, J. Allen, and A. Kott. "Resilience metrics for cyber systems". In: *Environment Systems and Decisions* 33.4 (Dec. 2013), pp. 471–476. ISSN: 21945403. DOI: 10.1007/s10669-013-9485-y (see p. 23).

[58] A. Kott and T. Abdelzaher. "Resiliency and Robustness of Complex Systems and Networks". In: *Adaptive, Dynamic, and Resilient Systems* 67 (2014), pp. 67–86 (see p. 23).

[59] N. O. Leslie, R. E. Harang, L. P. Knachel, and A. Kott. "Statistical models for the number of successful cyber intrusions". In: *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* (June 2017), p. 154851291771534. ISSN: 1548-5129. DOI: 10.1177/1548512917715342 (see p. 23).

[60] E. Colbert, A. Kott, L. P. Knachel, and D. T. Sullivan. *Modeling Cyber Physical War Gaming*. Tech. rep. US Army Research Laboratory Aberdeen Proving Ground United States, 2017, p. 46. URL: http://www.dtic.mil/docs/citations/AD1038105 (see p. 23).

[61] M. Ownby and A. Kott. "Predicting Enemy's Actions Improves Commander Decision-Making". In: (July 2016). arXiv: 1607.06759 (see p. 23).

[62] J. Brynielsson, U. Franke, M. Adnan Tariq, and S. Varga. "Using cyber defense exercises to obtain additional data for attacker profiling". In: *IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016*. IEEE, Sept. 2016, pp. 37–42. ISBN: 9781509038657. DOI: 10.1109/ISI.2016.7745440 (see p. 24).

[63] M. A. Tariq, J. Brynielsson, and H. Artman. "The security awareness paradox: A case study". In: *ASONAM 2014 - Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2014, pp. 704–711. ISBN: 9781479958771. DOI: 10.1109/ASONAM.2014.6921663 (see p. 24).

[64] H. Du, C. Wang, T. Zhang, S. J. Yang, J. Choi, and P. Liu. "Cyber Insider Mission Detection for Situation Awareness". In: *Studies in Computational Intelligence*. Vol. 563. Springer, Cham, 2015, pp. 201–217. DOI: 10.1007/978-3-319-08624-8_9 (see p. 24).

[65] J. Holsopple, M. Sudit, and S. J. Yang. "Impact Assessment". In: *Cyber Defense and Situational Awareness*. Springer, Cham, 2014, pp. 219–238. DOI: 10.1007/978-3-319-11391-3_11 (see p. 24).

[66] J. Holsopple, M. Sudit, M. Nusinov, D. Liu, H. Du, and S. Yang. "Enhancing situation awareness via automated situation assessment". In: *IEEE Communications Magazine* 48.3 (Mar. 2010), pp. 146–152. ISSN: 0163-6804. DOI: 10.1109/MCOM.2010.5434386 (see p. 24).

[67] S. J. Yang, H. Du, J. Holsopple, and M. Sudit. "Attack projection". In: *Cyber Defense and Situational Awareness*. Vol. 62. Springer, Cham, 2014, pp. 239–261. ISBN: 3319113909. DOI: 10.1007/978-3-319-11391-3_12 (see p. 24).

[68] A. Okutan, S. J. Yang, and K. McConky. "Predicting cyber attacks with bayesian networks using unconventional signals". In: *Proceedings of the 12th Annual Conference on Cyber and Information Security Research - CISRC '17*. New York, New York, USA: ACM Press, 2017, pp. 1–4. ISBN: 9781450348553. DOI: 10.1145/3064814.3064823 (see p. 24).

[69] G. Werner, S. Yang, and K. McConky. "Time series forecasting of cyber attack intensity". In: *Proceedings of the 12th Annual Conference on Cyber and Information Security Research - CISRC '17*. New York, New York, USA: ACM Press, 2017, pp. 1–3. ISBN: 9781450348553. DOI: 10.1145/3064814.3064831 (see p. 24).

[70] A. L. Krall, M. E. Kuhl, S. F. Moskal, and S. J. Yang. "Assessing the likelihood of cyber network infiltration using rare-event simulation". In: *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Dec. 2016, pp. 1–7. ISBN: 978-1-5090-4240-1. DOI: 10.1109/SSCI.2016.7849913 (see p. 24).

[71] H. Du, D. F. Liu, J. Holsopple, and S. J. Yang. "Toward Ensemble Characterization and Projection of Multistage Cyber Attacks". In: *2010 Proceedings of 19th International Conference on Computer Communications and Networks*. IEEE, Aug. 2010, pp. 1–8. ISBN: 978-1-4244-7114-0. DOI: 10.1109/ICCCN.2010.5560087 (see p. 24).

[72] S. Moskal, B. Wheeler, D. Kreider, M. E. Kuhl, and S. J. Yang. "Context model fusion for multistage network attack simulation". In: *IEEE Military Communications Conference MILCOM*. IEEE, Oct. 2014, pp. 158–163. ISBN: 9781479967704. DOI: 10.1109/MILCOM.2014.32 (see p. 24).

[73] T. Pahi, M. Leitner, and F. Skopik. "Analysis and Assessment of Situational Awareness Models for National Cyber Security Centers". In: *Proceedings of the 3rd International Conference on Information Systems Security and Privacy*. 2017, pp. 334–345. ISBN: 978-989-758-209-7. DOI: 10.5220/0006149703340345 (see p. 24).

[74] R. Graf, F. Skopik, and K. Whitebloom. "A decision support model for situational awareness in National Cyber Operations Centers". In: *2016 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (CyberSA)*. London, UK: IEEE, June 2016, pp. 1–6. ISBN: 978-1-5090-0703-5. DOI: 10.1109/CyberSA.2016.7503281 (see p. 24).

[75] Y. Shovgenya, F. Skopik, and K. Theuerkauf. "On demand for situational awareness for preventing attacks on the smart grid". In: *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, June 2015, pp. 1–4. ISBN: 978-0-9932-3380-7. DOI: 10.1109/CyberSA.2015.7166133 (see p. 24).

[76] F. Skopik, Z. Ma, P. Smith, and T. Bleier. "Designing a cyber attack information system for national situational awareness". In: *Communications in Computer and Information Science*. Vol. 318 CCIS. Bonn, Germany, 2012, pp. 277–288. ISBN: 9783642331602. DOI: 10.1007/978-3-642-33161-9_42 (see p. 24).

[77] J. Dressler, C. L. Bowen, W. Moody, and J. Koepke. "Operational data classes for establishing situational awareness in cyberspace". In: *2014 6th International Conference On Cyber Conflict (CyCon 2014)*. IEEE, June 2014, pp. 175–186. ISBN: 978-9949-9544-1-4. DOI: 10.1109/CYCON.2014.6916402 (see pp. 24, 26).

[78] K. Chandrasekar, G. Cleary, O. Cox, H. Lau, B. Nahorney, B. O. Gorman, D. O'Brien, S. Wallace, P. Wood, and C. Wueest. *Internet Security Threat Report - ISTR*. 2017. URL: https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf%20https://digitalhubshare.symantec.com/content/dam/Atlantis/campaigns-and-launches/FY17/Threat%20Protection/ISTR22%7B%5C_%7DMain-FINAL-JUN8.pdf?aid=elq%7B%5C_%7D (visited on Oct. 23, 2017) (see p. 25).

[79] J. Dijcks. *Oracle: Big data for the enterprise*. Redwood Shores, 2012. URL: http://www.oracle.com/us/products/database/big-data-for-enterprise-519135.pdf (visited on Jan. 23, 2018) (see p. 25).

[80] J. S. Ward and A. Barker. "Undefined By Data: A Survey of Big Data Definitions". In: *Communications of the ACM* 58.7 (June 2013), pp. 56–68. ISSN: 00010782. DOI: 10.1145/2699414. arXiv: 1309.5821 (see p. 25).

[81] Cisco. *Cisco Visual Networking Index: Forecast and Methodology, 2016-2021*. 2017. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf (visited on Jan. 25, 2018) (see pp. 25, 37, 47, 58, 148).

[82] Statista. *Apple App Store: number of available apps 2017 | Statistic*. 2017. URL: https://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/ (visited on Jan. 25, 2018) (see p. 25).

[83] R. van der Meulen and C. Pettey. *Gartner Forecasts Worldwide Security Spending Will Reach $96 Billion in 2018, Up 8 Percent from 2017*. URL: https://www.gartner.com/newsroom/id/3836563 (visited on Jan. 25, 2018) (see pp. 25, 26).

[84] R. Howard. "Information Value Theory". In: *IEEE Transactions on Systems Science and Cybernetics* 2.1 (1966), pp. 22–26. ISSN: 0536-1567. DOI: 10.1109/TSSC.1966.300074 (see p. 26).

[85] H. Shiravi, A. Shiravi, and A. A. Ghorbani. "A Survey of Visualization Systems for Network Security". In: *IEEE Transactions on Visualization and Computer Graphics* 18.8 (Aug. 2012), pp. 1313–1329. ISSN: 1077-2626. DOI: 10.1109/TVCG.2011.144 (see p. 27).

[86] V. T. Guimaraes, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, and L. Z. Granville. "A Survey on Information Visualization for Network and Service Management". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 285–323. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2450538 (see p. 27).

[87] W. J. Matuszak, L. DiPippo, and Y. L. Sun. "CyberSAVe". In: *Proceedings of the Tenth Workshop on Visualization for Cyber Security - VizSec '13*. New York, New York, USA: ACM Press, 2013, pp. 25–32. ISBN: 9781450321730. DOI: 10.1145/2517957.2517961 (see p. 27).

[88] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. "Flow monitoring explained: From packet capture to data analysis with NetFlow and IPFIX". In: *IEEE Communications Surveys and Tutorials* 16.4 (2014), pp. 2037–2064. ISSN: 1553877X. DOI: 10.1109/COMST.2014.2321898 (see pp. 29, 30, 33, 34, 38, 39, 42–44, 47, 48, 125).

[89] A. S. K. Pathan. *The State of the Art in Intrusion Prevention and Detection*. 2014, p. 496. ISBN: 9781482203523 (see p. 30).

[90] J. Weber. *The Fundamentals of Passiv e Monitoring Access*. 2006. URL: https://www.nanog.org/meetings/nanog37/presentations/joy-weber.pdf (visited on Feb. 12, 2018) (see p. 31).

[91] J. Zhang and A. Moore. "Traffic Trace Artifacts due to Monitoring Via Port Mirroring". In: *2007 Workshop on End-to-End Monitoring Techniques and Services*. IEEE, May 2007, pp. 1–8. ISBN: 1-4244-1289-7. DOI: 10.1109/E2EMON.2007.375317 (see p. 31).

[92] D. Levi, P. Meyer, and B. Stewart. *Simple Network Management Protocol (SNMP) Applications*. RFC 3413 (INTERNET STANDARD). Dec. 2002. URL: http://www.ietf.org/rfc/rfc3413.txt (see p. 31).

[93] S. Waldbusser, R. Cole, C. Kalbfleisch, and D. Romascanu. *Introduction to the Remote Monitoring (RMON) Family of MIB Modules*. RFC 3577 (Informational). Aug. 2003. URL: `http://www.ietf.org/rfc/rfc3577.txt` (see p. 31).

[94] Internet Security Research Group. *Let's Encrypt Stats*. 2018. URL: `https://letsencrypt.org/stats/` (visited on Feb. 19, 2018) (see pp. 32, 37).

[95] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. "An Architecture for Large-scale Internet Measurement". In: *IEEE Communications Magazine* 36.8 (1998), pp. 48–54. ISSN: 01636804. DOI: `10.1109/35.707817` (see p. 32).

[96] C. Mills, D. Hirsh, and G. R. Ruth. *Internet Accounting: Background*. RFC 1272 (Informational). Nov. 1991. URL: `http://www.ietf.org/rfc/rfc1272.txt` (see pp. 32, 33).

[97] K. Claffy, H.-W. Braun, and G. Polyzos. "A parameterizable Methodology for Internet Traffic Flow Profiling". In: *IEEE Journal on Selected Areas in Communications* 13.8 (1995), pp. 1481–1494. ISSN: 07338716. DOI: `10.1109/49.464717` (see p. 33).

[98] N. Brownlee, C. Mills, and G. Ruth. *Traffic Flow Measurement: Architecture*. RFC 2722 (Informational). Oct. 1999. URL: `http://www.ietf.org/rfc/rfc2722.txt` (see p. 33).

[99] J. Quittek, T. Zseby, B. Claise, and S. Zander. *Requirements for IP Flow Information Export (IPFIX)*. RFC 3917 (Informational). Oct. 2004. URL: `http://www.ietf.org/rfc/rfc3917.txt` (see p. 33).

[100] S. Leinen. *Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)*. RFC 3955 (Informational). Oct. 2004. URL: `http://www.ietf.org/rfc/rfc3955.txt` (see p. 33).

[101] B. Claise. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information*. RFC 5101 (Proposed Standard). Jan. 2008. URL: `http://www.ietf.org/rfc/rfc5101.txt` (see p. 33).

[102] B. Claise, B. Trammell, and P. Aitken. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. RFC 7011 (INTERNET STANDARD). Sept. 2013. URL: `http://www.ietf.org/rfc/rfc7011.txt` (see pp. 34–36, 38, 41, 42).

[103] B. Claise and B. Trammell. *Information Model for IP Flow Information Export (IPFIX)*. RFC 7012 (Proposed Standard). Sept. 2013. URL: `http://www.ietf.org/rfc/rfc7012.txt` (see p. 34).

[104] S. D'Antonio, T. Zseby, C. Henke, and L. Peluso. *Flow Selection Techniques*. RFC 7014 (Proposed Standard). Sept. 2013. URL: `http://www.ietf.org/rfc/rfc7014.txt` (see pp. 34, 41).

[105] B. Trammell, A. Wagner, and B. Claise. *Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol*. RFC 7015 (Proposed Standard). Sept. 2013. URL: `http://www.ietf.org/rfc/rfc7015.txt` (see p. 34).

[106] B. Trammell. *Textual Representation of IP Flow Information Export (IPFIX) Abstract Data Types*. RFC 7373 (Proposed Standard). Sept. 2014. URL: `http://www.ietf.org/rfc/rfc7373.txt` (see p. 34).

[107] B. Claise. *Cisco Systems NetFlow Services Export Version 9*. RFC 3954 (Informational). Oct. 2004. URL: `http://www.ietf.org/rfc/rfc3954.txt` (see p. 34).

[108] L. Deri, E. Chou, Z. Cherian, K. Karmarkar, and M. Patterson. "Increasing Data Center Network Visibility with Cisco NetFlow-Lite". In: *Proceedings of the 7th International Conference on Network and Services Management*. Laxenburg, Austria, Austria: International Federation for Information Processing, 2011, pp. 274–279. ISBN: 978-3-901882-44-9 (see p. 34).

[109] Cisco Systems Inc. *NetFlow Services Solutions Guide*. 2007. URL: `https://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products%7B%5C_%7Dimplementation%7B%5C_%7Ddesign%7B%5C_%7Dguide09186a00800d6a11.html%7B%5C#%7Dwp1093125` (visited on Feb. 26, 2018) (see p. 34).

[110] Cisco Systems Inc. *NetFlow Lite Configuration Guide, Cisco IOS Release 15.2(2)E (Catalyst 2960-X Switch)*. San Jose, 2014. URL: `https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960x/software/15-2%7B%5C_%7D2%7B%5C_%7De/fnf/configuration%7B%5C_%7Dguide/b%7B%5C_%7Dfnf%7B%5C_%7D1522e%7B%5C_%7D2960x%7B%5C_%7Dcg.pdf` (visited on Feb. 26, 2018) (see p. 34).

[111] Juniper Networks. *JUNIPER Flow Monitoring*. 2011. URL: `https://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf` (visited on Feb. 26, 2018) (see p. 34).

[112] P. Phaal, S. Panchen, and N. McKee. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*. RFC 3176 (Informational). Sept. 2001. URL: `http://www.ietf.org/rfc/rfc3176.txt` (see p. 34).

[113] Open Networking Foundation. *OpenFlow Switch Specification*. 2015. URL: `https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf` (visited on Feb. 27, 2018) (see p. 34).

[114] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: enabling innovation in campus network". In: *ACM SIGCOMM Computer Communication Review* 38.2 (Mar. 2008), p. 69. ISSN: 01464833. DOI: `10.1145/1355734.1355746` (see p. 35).

[115] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha. "FlowSense: Monitoring network utilization with zero measurement cost". In: *Lecture Notes in Computer Science*. Vol. 7799 LNCS. Springer, Berlin, Heidelberg, 2013, pp. 31–41. ISBN: 9783642365157. DOI: `10.1007/978-3-642-36516-4-4` (see p. 35).

[116] V. Pus, P. Velan, L. Kekely, J. Korenek, and P. Minarik. "Hardware accelerated flow measurement of 100 Gb ethernet". In: *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, May 2015, pp. 1147–1148. ISBN: 978-1-4799-8241-7. DOI: `10.1109/INM.2015.7140452` (see pp. 36, 58).

[117] Y. Lee and Y. Lee. "Toward scalable internet traffic measurement and analysis with Hadoop". In: *ACM SIGCOMM Computer Communication Review* 43.1 (Jan. 2012), p. 5. ISSN: 01464833. DOI: `10.1145/2427036.2427038` (see p. 37).

[118] S. Marchal, X. Jiang, R. State, and T. Engel. "A big data architecture for large scale security monitoring". In: *Proceedings - 2014 IEEE International Congress on Big Data, BigData Congress 2014*. IEEE, June 2014, pp. 56–63. ISBN: 9781479950577. DOI: `10.1109/BigData.Congress.2014.18` (see p. 37).

[119] A. Bar, A. Finamore, P. Casas, L. Golab, and M. Mellia. "Large-scale network traffic monitoring with DBStream, a system for rolling big data analysis". In: *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*. IEEE, Oct. 2015, pp. 165–170. ISBN: 9781479956654. DOI: `10.1109/BigData.2014.7004227` (see p. 37).

[120] I. Jolly. *Data protection in the United States: overview*. 2017. URL: `https://uk.practicallaw.thomsonreuters.com/6-502-0467?originationContext=document%7B%5C&%7DtransitionType=DocumentItem%7B%5C&%7DcontextData=(sc.Default)%7B%5C#%7Dco%7B%5C_%7Danchor%7B%5C_%7Da233354` (visited on Mar. 1, 2017) (see p. 37).

[121] Oracle Corporation. *Cloud Predictions 2017*. 2017. URL: `http://www.oracle.com/us/solutions/cloud/top-10-predictions-cloud-3436083.pdf` (visited on Mar. 1, 2018) (see p. 37).

[122] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek. *Architecture for IP Flow Information Export*. RFC 5470 (Informational). Mar. 2009. URL: `http://www.ietf.org/rfc/rfc5470.txt` (see pp. 38, 40).

[123] A.-C. Orgerie, P. Goncalves, M. Imbert, J. Ridoux, and D. Veitch. "Survey of Network Metrology Platforms". In: *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*. IEEE, July 2012, pp. 220–225. ISBN: 978-1-4673-2001-6. DOI: `10.1109/SAINT.2012.41` (see pp. 39, 45).

[124] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall. *Sampling and Filtering Techniques for IP Packet Selection*. RFC 5475 (Proposed Standard). Mar. 2009. URL: `http://www.ietf.org/rfc/rfc5475.txt` (see p. 39).

[125] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. "An Overview of IP Flow-Based Intrusion Detection". In: *IEEE Communications Surveys & Tutorials* 12.3 (2010), pp. 343–356. ISSN: 1553-877X. DOI: `10.1109/SURV.2010.032210.00054` (see pp. 40, 41, 51, 125).

[126] C. Estan and G. Varghese. "New directions in traffic measurement and accounting". In: *ACM SIGCOMM Computer Communication Review*. Vol. 32. 4. New York, USA: ACM Press, Oct. 2002, p. 323. ISBN: 158113570X. DOI: `10.1145/964725.633056` (see p. 41).

[127] N. Duffield, C. Lund, and M. Thorup. "Charging from sampled network usage". In: *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement - IMW '01*. New York, New York, USA: ACM Press, 2001, p. 245. ISBN: 1581134355. DOI: `10.1145/505202.505232` (see p. 41).

[128] N. Duffield, C. Lund, and M. Thorup. "Flow sampling under hard resource constraints". In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 32. 1. ACM, June 2004, p. 85. ISBN: 1581138733. DOI: `10.1145/1012888.1005699` (see p. 41).

[129] B. Trammell, E. Boschi, L. Mark, T. Zseby, and A. Wagner. *Specification of the IP Flow Information Export (IPFIX) File Format*. RFC 5655 (Proposed Standard). Oct. 2009. URL: `http://www.ietf.org/rfc/rfc5655.txt` (see p. 42).

[130] L. Deri. "nProbe: an Open Source NetFlow Probe for Gigabit Networks". In: *TERENA Networking Conference (TNC 2003), Zagreb, Croatia*. 2003, pp. 1–4 (see pp. 43, 65).

[131] R. Hofstede, A. Sperotto, T. Fioreze, and A. Pras. "The Network Data Handling War: MySQL vs. NfDump". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6164 LNCS. Springer, Berlin, Heidelberg, 2010, pp. 167–176. ISBN: 3642139701. DOI: `10.1007/978-3-642-13971-0_16` (see pp. 44, 53).

[132] P. Velan. "Practical experience with IPFIX flow collectors". In: *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management, IM 2013*. 2013, pp. 1021–1026. ISBN: 9783901882517 (see pp. 44, 53).

[133] M. Zadnik, P. Krobot, and J. Wrona. *Experience with Big Data Frameworks for IP Flow Collector*. Working paper. 2013. URL: `https://www.liberouter.org/wp-content/uploads/2013/02/benchmark-collector.pdf` (visited on Mar. 28, 2018) (see pp. 44, 146).

[134] D. Moore and K. Claffy. *Summary of Anonymization Best Practice Techniques*. 2016. URL: `http://www.caida.org/projects/predict/anonymization/` (visited on Mar. 22, 2018) (see p. 44).

[135] CAIDA. *Annotated Bibliography of Networking Papers*. Web page. 2013. URL: `https://www.caida.org/publications/bib/networking/bytopic/index.xml%7B%5C#%7Danonymization` (visited on Mar. 22, 2018) (see p. 44).

[136] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. "Prefix-preserving IP address anonymization: measurement-based security evaluation and a new cryptography-based scheme". In: *Computer Networks* 46.2 (Oct. 2004), pp. 253–272. ISSN: 13891286. DOI: `10.1016/j.comnet.2004.03.033` (see p. 44).

[137] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner. "The role of network trace anonymization under attack". In: *ACM SIGCOMM Computer Communication Review* 40.1 (Jan. 2010), p. 5. ISSN: 01464833. DOI: `10.1145/1672308.1672310` (see p. 44).

[138] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter. "On Web Browsing Privacy in Anonymized NetFlows". In: *SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*. USENIX Association, 2007, pp. 1–14. ISBN: 111-333-5555-77-9 (see p. 44).

[139] S. E. Coull, F. Monrose, M. K. Reiter, and M. Bailey. "The Challenges of Effectively Anonymizing Network Data". In: *2009 Cybersecurity Applications & Technology Conference for Homeland Security*. IEEE, Mar. 2009, pp. 230–236. ISBN: 978-0-7695-3568-5. DOI: 10.1109/CATCH.2009.27 (see pp. 44, 54).

[140] M. F. Umer, M. Sher, and Y. Bi. "Flow-based intrusion detection: Techniques and challenges". In: *Computers and Security* 70 (2017), pp. 238–254. ISSN: 01674048. DOI: 10.1016/j.cose.2017.05.009 (see pp. 47, 51, 53).

[141] R. Hofstede, I. Drago, A. Sperotto, R. Sadre, and A. Pras. "Measurement Artifacts in NetFlow Data". In: *International Conference on Passive and Active Network Measurement (PAM 2013)*. Springer, Berlin, Heidelberg, 2013, pp. 1–10. DOI: 10.1007/978-3-642-36516-4_1 (see p. 47).

[142] B. Li, J. Springer, G. Bebis, and M. Hadi Gunes. "A survey of network flow applications". In: *Journal of Network and Computer Applications* 36.2 (2013), pp. 567–581. ISSN: 10848045. DOI: 10.1016/j.jnca.2012.12.020 (see pp. 48, 50).

[143] A. Moore, D. Zuev, and M. Crogan. "Discriminators for Use in Flow-based Classification". In: *Queen Mary and Westfield College, Department of Computer Science* 16.August (2005), p. 2005. ISSN: 1470-5559. DOI: 10.1.1.101.7450 (see p. 49).

[144] G. Moura, R. Sadre, and A. Pras. "Bad neighborhoods on the internet". In: *IEEE Communications Magazine* 52.7 (July 2014), pp. 132–139. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6852094 (see p. 51).

[145] J. Vykopal, M. Drasar, and P. Winter. "Flow-based Brute-force Attack Detection". In: *Advances in IT Early Warning*. Stuttgart, Germany: Fraunhofer Verlag, 2013, p. 12. ISBN: 9783839604748 (see p. 51).

[146] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. "Network Anomaly Detection: Methods, Systems and Tools". In: *IEEE Communications Surveys & Tutorials* 16.1 (2014), pp. 303–336. ISSN: 1553-877X. DOI: 10.1109/SURV.2013.052213.00046. arXiv: 878 (see pp. 51–53, 59).

[147] A. Patcha and J. M. Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends". In: *Computer Networks* 51.12 (Aug. 2007), pp. 3448–3470. ISSN: 13891286. DOI: 10.1016/j.comnet.2007.02.001 (see p. 51).

[148] V. Chandola, A. Banerjee, and V. Kumar. "Anomaly detection: A survey". In: *ACM Computing Surveys* 41.3 (July 2009), pp. 1–58. ISSN: 03600300. DOI: 10.1145/1541880.1541882 (see p. 51).

[149] T. T. Nguyen and G. Armitage. "A survey of techniques for internet traffic classification using machine learning". In: *IEEE Communications Surveys & Tutorials* 10.4 (2008), pp. 56–76. ISSN: 1553-877X. DOI: 10.1109/SURV.2008.080406 (see p. 51).

[150] A. L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: *IEEE Communications Surveys & Tutorials* 18.2 (2016), pp. 1153–1176. ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2494502 (see pp. 51, 54).

[151] Y. Xue, L. Zhang, and D. Wang. "Traffic classification: Issues and challenges". In: *International Conference on Computing, Networking and Communications (ICNC)*. Vol. 8. 4. IEEE, Jan. 2013, pp. 28–31. ISBN: 978-1-4673-5287-1. DOI: 10.1109/ICCNC.2013.6504144 (see p. 52).

[152] S. M. Sangve and R. Thool. "A formal assessment of anomaly network intrusion detection methods and techniques using various datasets". In: *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*. IEEE, Oct. 2015, pp. 267–272. ISBN: 978-1-4673-9223-5. DOI: 10.1109/ICATCCT.2015.7456894 (see p. 53).

[153] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. "An Evaluation Framework for Intrusion Detection Dataset". In: *2016 International Conference on Information Science and Security (ICISS)*. IEEE, Dec. 2016, pp. 1–6. ISBN: 978-1-5090-5493-0. DOI: `10.1109/ICISSEC.2016.7885840` (see p. 53).

[154] E. K. Viegas, A. O. Santin, and L. S. Oliveira. "Toward a reliable anomaly-based intrusion detection in real-world environments". In: *Computer Networks* 127 (Nov. 2017), pp. 200–216. ISSN: 13891286. DOI: `10.1016/j.comnet.2017.08.013` (see p. 53).

[155] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani. "Toward credible evaluation of anomaly-based intrusion-detection methods". In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 40.5 (Sept. 2010), pp. 516–524. ISSN: 10946977. DOI: `10.1109/TSMCC.2010.2048428` (see p. 53).

[156] S. Bose. "Network data anonymization and its effect on security analysis". PhD thesis. University of California, 2017, p. 231. URL: `https://search.proquest.com/openview/da21f849a1e2e3295d881331d7fa2578/1?pq-origsite=gscholar%7B%5C&%7Dcbl=18750%7B%5C&%7Ddiss=y` (see p. 54).

[157] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard). June 1999. URL: `http://www.ietf.org/rfc/rfc2616.txt` (see pp. 65, 66).

[158] I. Cisco Systems. *Application Visibility and Control*. Web page. Apr. 2013. URL: `http://www.cisco.com/go/avc` (visited on Apr. 12, 2018) (see p. 65).

[159] C. M. Inacio and B. Trammell. "YAF: Yet Another Flowmeter". In: *Proceedings of the 24th international conference on Large installation system administration*. LISA'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–16 (see pp. 65, 68).

[160] *The GNU C Library (glibc)*. Web page. 2012. URL: `http://www.gnu.org/software/libc/` (visited on Apr. 20, 2018) (see p. 65).

[161] *PCRE - Perl Compatible Regular Expressions*. Web page. 2012. URL: `http://www.pcre.org/` (visited on Apr. 22, 2018) (see p. 65).

[162] Open Information Security Foundation. *Suricata – network IDS, IPS and network security monitoring engine*. Web page. Apr. 2013. URL: `http://www.suricata-ids.org` (visited on Apr. 22, 2018) (see p. 65).

[163] M. Roesch. "Snort - Lightweight Intrusion Detection for Networks". In: *Proceedings of the 13th USENIX conference on System administration*. LISA '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238 (see p. 65).

[164] I. Qualys. *LibHTP – security-aware parser for the HTTP protocol*. Web page. Apr. 2013. URL: `http://github.com/ironbee/libhtp` (visited on Apr. 26, 2018) (see p. 65).

[165] J. Bittel. *httpry - HTTP logging and information retrieval tool*. Web page. Apr. 2013. URL: `http://github.com/jbittel/httpry` (visited on Apr. 24, 2018) (see p. 65).

[166] V. Paxson. "Bro: a system for detecting network intruders in real-time". In: *Comput. Netw.* 31.23-24 (Dec. 1999), pp. 2435–2463. ISSN: 1389-1286. DOI: `10.1016/S1389-1286(99)00112-7` (see p. 65).

[167] R. Pang, V. Paxson, R. Sommer, and L. Peterson. "binpac: A yacc for Writing Application Protocol Parsers". In: *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. IMC '06. New York, NY, USA: ACM, 2006, pp. 289–300. ISBN: 1-59593-561-4. DOI: `10.1145/1177080.1177119` (see p. 65).

[168] J. Levine and L. John. *Flex & Bison*. 1st. O'Reilly Media, Inc., 2009. ISBN: 0596155972, 9780596155971 (see p. 65).

[169] M. E. Lesk and E. Schmidt. *Lex – a Lexical Analyzer Generator*. Tech. rep. Bell Laboratories, 1975 (see p. 65).

[170] R. McNaughton and H. Yamada. "Regular Expressions and State Graphs for Automata". In: *Electronic Computers, IRE Transactions on* EC-9.1 (1960), pp. 39–47. ISSN: 0367-9950. DOI: `10.1109/TEC.1960.5221603` (see p. 65).

[171] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. "The New Web: Characterizing AJAX Traffic". In: *Proceedings of the 9th international conference on Passive and active network measurement*. PAM'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 31–40. ISBN: 3-540-79231-7, 978-3-540-79231-4 (see p. 66).

[172] L. M. Torres, E. Magana, M. Izal, and D. Morato. "Identifying Sessions to Websites as an Aggregation of Related Flows". In: *Telecommunications Network Strategy and Planning Symposium (NETWORKS), 2012 XVth International*. 2012, pp. 1–6. DOI: `10.1109/NETWKS.2012.6381703` (see p. 66).

[173] L. M. Torres, E. Magana, M. Izal, and D. Morato. "Strategies for Automatic Labelling of Web Traffic Traces". In: *37th Annual IEEE Conference on Local Computer Networks* 0 (2012), pp. 196–199. ISSN: 0742-1303. DOI: `http://doi.ieeecomputersociety.org/10.1109/LCN.2012.6423605` (see p. 66).

[174] V. Gehlen, A. Finamore, M. Mellia, and M. M. Munafò. "Uncovering the Big Players of the Web". In: *Proceedings of the 4th international conference on Traffic Monitoring and Analysis*. TMA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 15–28. ISBN: 978-3-642-28533-2. DOI: `10.1007/978-3-642-28534-9_2` (see p. 66).

[175] A. Mahanti, C. Williamson, N. Carlsson, M. Arlitt, and A. Mahanti. "Characterizing the file hosting ecosystem: A view from the edge". In: *Perform. Eval.* 68.11 (Nov. 2011), pp. 1085–1102. ISSN: 0166-5316. DOI: `10.1016/j.peva.2011.07.016` (see p. 66).

[176] Flowmon Networks. *FlowMon Exporter – Community Program*. 2013. URL: `https://www.flowmon.com/en` (visited on May 3, 2018) (see pp. 67, 101).

[177] T. Sima, P. Velan, and P. Celeda. *FlowMon - Plugins for HTTP Monitoring*. Apr. 2013. URL: `http://dior.ics.muni.cz/%7B~%7Dvelan/flowmon-input-http/` (visited on Apr. 26, 2018) (see p. 67).

[178] B. Möller, T. Duong, and K. Kotowicz. *This POODLE Bites: Exploiting The SSL 3.0 Fallback*. PDF online. 2014. URL: `https://www.openssl.org/%7B~%7Dbodo/ssl-poodle.pdf` (visited on Apr. 19, 2018) (see pp. 74, 83).

[179] P. Velan, M. Cermak, P. Celeda, and M. Drasar. "A survey of methods for encrypted traffic classification and analysis". In: *Internation Journal of Network Management* 25.October 2005 (2007), pp. 17–31. ISSN: 10557148. DOI: `10.1002/nem` (see p. 74).

[180] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. "Web Tracking: Mechanisms, Implications, and Defenses". In: *CoRR* abs/1507.0 (2015) (see pp. 75, 77, 121).

[181] I. Zeifman. *Was that really a Google bot crawling my site?* 2012. URL: `https://www.incapsula.com/blog/was-that-really-a-google-bot-crawling-my-site.html` (visited on Apr. 29, 2018) (see p. 75).

[182] E. Raftopoulos and X. Dimitropoulos. "Understanding network forensics analysis in an operational environment". In: *Security and Privacy Workshops (SPW), 2013 IEEE*. IEEE. 2013, pp. 111–118 (see pp. 76, 78).

[183] P. Wang, S. Sparks, and C. C. Zou. "An Advanced Hybrid Peer-to-Peer Botnet". In: *Dependable and Secure Computing, IEEE Transactions on* 7.2 (Apr. 2010), pp. 113–127. ISSN: 1545-5971 (see p. 76).

[184] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Aug. 2008. URL: `http://www.ietf.org/rfc/rfc5246.txt` (see pp. 76, 77).

[185] A. Freier, P. Karlton, and P. Kocher. *The Secure Sockets Layer (SSL) Protocol Version 3.0*. RFC 6101 (Historic). Aug. 2011. URL: `http://www.ietf.org/rfc/rfc6101.txt` (see p. 76).

[186] E. Rescorla. *HTTP Over TLS*. RFC 2818 (Informational). May 2000. URL: `http://www.ietf.org/rfc/rfc2818.txt` (see p. 76).

[187] IANA – Internet Assigned Numbers Authority. *Protocol Registries*. Web page. 2014. URL: `http://www.iana.org/protocols` (visited on Apr. 29, 2018) (see p. 76).

[188] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280 (Proposed Standard). May 2008. URL: `http://www.ietf.org/rfc/rfc5280.txt` (see p. 76).

[189] C. Meyer. "20 Years of SSL/TLS Research: An Analysis of the Internet's Security Foundation". PhD thesis. Ruhr-University Bochum, Feb. 2014. URL: `http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/MeyerChristopher/diss.pdf` (see p. 76).

[190] O. Levillain, A. Ébalard, B. Morin, and H. Debar. "One Year of SSL Internet Measurement". In: *Proceedings of the 28th Annual Computer Security Applications Conference*. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 11–20. ISBN: 978-1-4503-1312-4 (see pp. 77, 83).

[191] R. Holz, L. Braun, N. Kammenhuber, and G. Carle. "The SSL Landscape: A Thorough Analysis of the x.509 PKI Using Active and Passive Measurements". In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*. IMC '11. New York, NY, USA: ACM, 2011, pp. 427–444. ISBN: 978-1-4503-1013-0 (see p. 77).

[192] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman. "Analysis of the HTTPS Certificate Ecosystem". In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. New York, NY, USA: ACM, 2013, pp. 291–304. ISBN: 978-1-4503-1953-9 (see p. 77).

[193] Qualys SSL Lab. *HTTP Client Fingerprinting Using SSL Handshake Analysis*. Web page. 2014. URL: `https://www.ssllabs.com/projects/client-fingerprinting/` (visited on Apr. 30, 2018) (see p. 77).

[194] I. Ristić. *Passive SSL client fingerprinting using handshake analysis*. GitHub repository. 2014. URL: `https://github.com/ssllabs/sslhaf` (visited on Apr. 30, 2018) (see pp. 77, 79).

[195] J. B. Ullrich. *Browser Fingerprinting via SSL Client Hello Messages*. 2014. URL: `https://isc.sans.edu/forums/diary/Browser+Fingerprinting+via+SSL+Client+Hello+Messages/17210` (visited on Apr. 30, 2018) (see p. 77).

[196] M. Majkowski. *SSL fingerprinting for p0f*. Web page. June 2012. URL: `https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f` (visited on Apr. 30, 2018) (see p. 77).

[197] L. Bernaille and R. Teixeira. "Early Recognition of Encrypted Applications". In: *Passive and Active Network Measurement*. Vol. 4427. Lecture Notes in Computer Science. Heidelberg: Springer Berlin Heidelberg, 2007, pp. 165–175. ISBN: 978-3-540-71616-7 (see p. 77).

[198] T. Matsunaka, A. Yamada, and A. Kubota. "Passive OS Fingerprinting by DNS Traffic Analysis". In: *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, Mar. 2013, pp. 243–250. ISBN: 978-1-4673-5550-6. DOI: `10.1109/AINA.2013.119` (see pp. 77, 110).

[199] M. Zalewski. *p0f v3*. 2014. URL: `http://lcamtuf.coredump.cx/p0f3/` (visited on Apr. 30, 2018) (see p. 77).

[200] T. Kohno, A. Broido, and K. Claffy. "Remote physical device fingerprinting". In: *Security and Privacy, 2005 IEEE Symposium on*. May 2005, pp. 211–225 (see p. 77).

[201] T. Karagiannis, K. Papagiannaki, N. Taft, and M. Faloutsos. "Profiling the End Host". In: *Passive and Active Network Measurement*. Vol. 4427. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 186–196. ISBN: 978-3-540-71616-7 (see p. 77).

[202] H. J. Abdelnur, R. State, and O. Festor. "Advanced Network Fingerprinting". In: *Recent Advances in Intrusion Detection*. Vol. 5230. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 372–389. ISBN: 978-3-540-87402-7 (see p. 77).

[203] T. Unger, M. Mulazzani, D. Fruhwirt, M. Huber, S. Schrittwieser, and E. Weippl. "SHPF: Enhancing HTTP(S) Session Security with Browser Fingerprinting". In: *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*. Sept. 2013, pp. 255–261 (see p. 78).

[204] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, and E. Weippl. *Fast and Reliable Browser Identification with JavaScript Engine Fingerprinting*. 2013. URL: `http://www.ieee-security.org/TC/W2SP/2013/papers/s2p1.pdf` (visited on May 1, 2018) (see p. 78).

[205] P. Eckersley. "How Unique is Your Web Browser?" In: *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 1–18. ISBN: 3-642-14526-4, 978-3-642-14526-1 (see p. 78).

[206] Proxy4Free.com. *Free Proxy Servers - Protect Your Online Privacy with Our Proxy List*. Web page. 2015. URL: `http://www.proxy4free.com/` (visited on May 2, 2018) (see p. 78).

[207] R. Dingledine, N. Mathewson, and P. Syverson. "Tor: The Second-generation Onion Router". In: *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, p. 21 (see p. 78).

[208] Y. Gokcen, V. A. Foroushani, and A. Heywood. "Can we identify NAT behavior by analyzing Traffic Flows?" In: *Security and Privacy Workshops (SPW), 2014 IEEE*. IEEE. 2014, pp. 132–139 (see p. 78).

[209] V. Krmicek, J. Vykopal, and R. Krejci. "Netflow Based System for NAT Detection". In: *Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies*. Co-Next Student Workshop '09. New York, NY, USA: ACM, 2009, pp. 23–24. ISBN: 978-1-60558-751-6 (see p. 78).

[210] cURL Contributors. *cURL - command line tool and library for transferring data with URL syntax*. 2015. URL: `http://curl.haxx.se/` (visited on June 22, 2018) (see p. 81).

[211] X. Cao, W. Feng, Y. Dou, Z. Lei, and H. Yu. "A space-saving method for aggregate Top-N flow statistics with high accuracy". In: *Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on*. Oct. 2011, pp. 407–411. DOI: `10.1109/ICBNMT.2011.6155966` (see pp. 91, 92).

[212] Cisco Systems Inc. *Cisco IOS Master Command List , All Releases*. Online. Jan. 2014. URL: `https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mcl/allreleasemcl/all-book.pdf` (visited on June 22, 2018) (see p. 91).

[213] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *Commun. ACM* 51.1 (Jan. 2008), pp. 107–113. ISSN: 0001-0782. DOI: `10.1145/1327452.1327492` (see p. 92).

[214] A. Moore, D. Zuev, and M. Crogan. *Discriminators for use in flow-based classification*. Tech. rep. August. Queen Mary, University of London, 2005. DOI: `10.1.1.101.7450`. URL: `http://www.cl.cam.ac.uk/%7B~%7Dawm22/publications/moore2005discriminators.pdf` (see p. 92).

[215] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Vol. 1. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321321367 (see p. 95).

[216] K. Claffy. "Tracking IPv6 evolution". In: *ACM SIGCOMM Computer Communication Review* 41.3 (July 2011), p. 43. ISSN: 01464833. DOI: `10.1145/2002250.2002258` (see p. 98).

[217] M. Aazam, I. Khan, M. Alam, and A. Qayyum. "Comparison of ipv6 tunneled traffic of Teredo and ISATAP over test-bed setup". In: *2010 International Conference on Information and Emerging Technologies*. IEEE, June 2010, pp. 1–4. ISBN: 978-1-4244-8001-2. DOI: `10.1109/ICIET.2010.5625689` (see p. 99).

[218] S. Zander, L. L. Andrew, G. Armitage, G. Huston, and G. Michaelson. "Investigating the IPv6 teredo tunnelling capability and performance of internet clients". In: *ACM SIGCOMM Computer Communication Review* 42.5 (Sept. 2012), p. 13. ISSN: 01464833. DOI: `10.1145/2378956.2378959` (see p. 99).

[219] N. Bahaman, E. Hamid, and A. S. Prabuwono. "Network performance evaluation of 6to4 tunneling". In: *2012 International Conference on Innovation Management and Technology Research*. IEEE, May 2012, pp. 263–268. ISBN: 978-1-4673-0654-6. DOI: `10.1109/ICIMTR.2012.6236400` (see p. 99).

[220] S. Krishnan, D. Thaler, and J. Hoagland. *Security Concerns with IP Tunneling*. RFC 6169 (Informational). Apr. 2011. URL: `http://www.ietf.org/rfc/rfc6169.txt` (see p. 99).

[221] N. Sarrar, G. Maier, B. Ager, R. Sommer, and S. Uhlig. "Investigating IPv6 Traffic". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7192 LNCS. PAM'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 11–20. ISBN: 9783642285363. DOI: `10.1007/978-3-642-28537-0_2` (see p. 99).

[222] C. Huitema. *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*. RFC 4380 (Proposed Standard). Feb. 2006. URL: `http://www.ietf.org/rfc/rfc4380.txt` (see p. 100).

[223] B. Carpenter and K. Moore. *Connection of IPv6 Domains via IPv4 Clouds*. RFC 3056 (Proposed Standard). Feb. 2001. URL: `http://www.ietf.org/rfc/rfc3056.txt` (see p. 100).

[224] M. Elich, M. Gregr, and P. Celeda. "Monitoring of tunneled IPv6 traffic using packet decapsulation and IPFIXx (short paper)". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 6613 LNCS. TMA'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 64–71. ISBN: 9783642203046. DOI: `10.1007/978-3-642-20305-3_6` (see pp. 100, 108).

[225] M. Elich, P. Velan, and P. Celeda. *FlowMon IPv6 Tunnel Monitoring Plugin*. 2013. (Visited on May 5, 2018) (see pp. 100, 101).

[226] MaxMind. *MaxMind GeoIP services*. 2013. URL: `http://www.maxmind.com` (visited on May 4, 2018) (see p. 101).

[227] P. Velan and R. Krejci. "Flow Information Storage Assessment Using IPFIXcol." In: *AIMS*. Vol. 7279. Lecture Notes in Computer Science. Springer, 2012, pp. 155–158. ISBN: 978-3-642-30632-7 (see pp. 101, 144).

[228] S. Siby. *Default TTL (Time To Live) Values of Different OS*. 2014. URL: `https://subinsb.com/default-device-ttl-values/` (visited on Oct. 2, 2018) (see p. 102).

[229] C. Ç. Iflikli, A. Gezer, and A. T. Özşahin. "Packet traffic features of IPv6 and IPv4 protocol traffic". In: *Turkish Journal of Electrical Engineering and Computer Sciences* 20.5 (2012), pp. 727–749. ISSN: 13000632. DOI: `10.3906/elk-1008-696` (see p. 105).

[230] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. *Neighbor Discovery for IP version 6 (IPv6)*. RFC 4861 (Draft Standard). Sept. 2007. URL: `http://www.ietf.org/rfc/rfc4861.txt` (see p. 106).

[231] Google Inc. *Dashboards: Platform Versions*. URL: `https://developer.android.com/about/dashboards/index.html` (visited on Apr. 12, 2018) (see p. 109).

[232] A. Kott, C. Wang, and R. F. Erbacher. *Cyber defense and situational awareness*. Vol. 62. Springer, 2015 (see p. 109).

[233] R. Lippmann and D. Fried. "Passive operating system identification from TCP/IP packet headers". In: *Workshop on Data Mining for Computer Security*. 2003, p. 40 (see p. 110).

[234] R. Tyagi, T. Paul, B. Manoj, and B. Thanudas. "Packet Inspection for Unauthorized OS Detection in Enterprises". In: *IEEE Security & Privacy* 13.4 (July 2015), pp. 60–65. ISSN: 1540-7993. DOI: `10.1109/MSP.2015.86` (see p. 110).

[235] A. Aksoy, S. Louis, and M. H. Gunes. "Operating system fingerprinting via automated network traffic analysis". In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. June 2017, pp. 2502–2509. DOI: 10.1109/CEC.2017.7969609 (see p. 110).

[236] D. W. Richardson, S. D. Gribble, and T. Kohno. "The limits of automatic OS fingerprint generation". In: *Proceedings of the 3rd ACM workshop on Artificial intelligence and security - AISec '10*. ACM. New York, New York, USA: ACM Press, 2010, p. 24. ISBN: 9781450300889. DOI: 10.1145/1866423.1866430 (see p. 110).

[237] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231 (Proposed Standard). June 2014. URL: http://www.ietf.org/rfc/rfc7231.txt (see p. 111).

[238] Flowmon Networks. *Netflow, a new era of network traffic monitoring*. URL: https://www.flowmon.com/en/solutions/use-case/netflow-ipfix (visited on May 22, 2018) (see p. 111).

[239] P. Matoušek, O. Ryšavý, M. Grégr, and M. Vymlátil. "Towards Identification of Operating Systems from the Internet Traffic - IPFIX Monitoring with Fingerprinting and Clustering". In: *Proceedings of the 5th International Conference on Data Communication Networking*. SCITEPRESS - Science, Aug. 2014, pp. 21–27. ISBN: 978-989-758-042-0. DOI: 10.5220/0005099500210027 (see p. 112).

[240] M. Sokolova and G. Lapalme. "A systematic analysis of performance measures for classification tasks". In: *Information Processing & Management* 45.4 (2009), pp. 427–437. ISSN: 03064573. DOI: 10.1016/j.ipm.2009.03.002 (see p. 115).

[241] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W.-T. Chang, and Z. Shi. "The QUIC Transport Protocol: Design and Internet-Scale Deployment". In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 183–196. ISBN: 978-1-4503-4653-5. DOI: 10.1145/3098822.3098842 (see p. 117).

[242] M. Belshe, R. Peon, and M. Thomson. *Hypertext Transfer Protocol Version 2 (HTTP/2)*. RFC 7540 (Proposed Standard). May 2015. URL: http://www.ietf.org/rfc/rfc7540.txt (see p. 117).

[243] IETF HTTP Working Group. *HTTP/2 Frequently Asked Questions*. URL: https://http2.github.io/faq/ (visited on June 13, 2018) (see p. 117).

[244] O. Alders. *HTTP::BrowserDetect*. 2018-06-05. 2015. URL: https://github.com/oalders/http-browserdetect (visited on Mar. 22, 2018) (see p. 118).

[245] StatCounter. *StatCounter Global Stats*. 2015. URL: http://gs.statcounter.com/%7B%5C#%7Dall-browser-ww-monthly-201506-201506-map (visited on Mar. 7, 2018) (see p. 120).

[246] A. Cortesi. *mitmproxy*. 2014. URL: https://mitmproxy.org/ (visited on Apr. 9, 2018) (see p. 121).

[247] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. "Models and issues in data stream systems". In: *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '02*. New York, New York, USA: ACM Press, 2002, p. 1. ISBN: 1581135076. DOI: 10.1145/543613.543615. arXiv: arXiv:1011.1669v3 (see pp. 125–127).

[248] D. McCarthy and U. Dayal. "The Architecture of an Active Database Management System". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Vol. 18. 2. New York, New York, USA: ACM Press, 1989, pp. 215–224. ISBN: 0897913175. DOI: 10.1145/67544.66946 (see p. 126).

[249] G. Cugola and A. Margara. "Processing Flows of Information: From Data Stream to Complex Event Processing". In: *ACM Computing Surveys* 44.3 (June 2012), pp. 1–62. ISSN: 03600300. DOI: 10.1145/2187671.2187677 (see pp. 126, 128).

[250] M. Hirzel, R. Soulé, S. Schneider, B. Gedik, and R. Grimm. "A Catalog of Stream Processing Optimizations". In: *ACM Computing Surveys* 46.4 (Mar. 2014), pp. 1–34. ISSN: 03600300. DOI: 10.1145/2528412 (see pp. 127, 140).

[251] J. Gama and P. P. Rodrigues. "Data Stream Processing". In: *Learning from Data Streams*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 25–40. ISBN: 9783540736783. DOI: 10.1007/3-540-73679-4_3 (see p. 128).

[252] R. Lu, G. Wu, B. Xie, and J. Hu. "StreamBench: Towards Benchmarking Modern Distributed Stream Computing Frameworks". In: *Proceedings - 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, UCC 2014*. IEEE, Dec. 2014, pp. 69–78. ISBN: 9781479978816. DOI: 10.1109/UCC.2014.15 (see p. 128).

[253] M. Cermak, D. Tovarnak, M. Lastovicka, and P. Celeda. "A performance benchmark for NetFlow data analysis on distributed stream processing systems". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, Apr. 2016, pp. 919–924. ISBN: 978-1-5090-0223-8. DOI: 10.1109/NOMS.2016.7502926 (see pp. 128–130, 132, 151).

[254] L. Strzalkowski. *Queues*. 2016. URL: http://queues.io/ (visited on June 7, 2018) (see pp. 130, 131, 144).

[255] J. Kreps. *Benchmarking Apache Kafka: 2 Million Writes Per Second (On Three Cheap Machines)*. 2014. URL: https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines (visited on June 21, 2018) (see p. 132).

[256] N. Marz and J. Warren. *Big Data: Principles and best practices of scalable realtime data systems*. Vol. 1. Manning Publications, 2015, p. 328. ISBN: 978-1617290343 (see pp. 139, 141–143).

[257] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". In: *Communications of the ACM*. Vol. 51. OSDI'04 1. Berkeley, CA, USA: USENIX Association, Jan. 2008, p. 107. ISBN: 9781595936868. DOI: 10.1145/1327452.1327492. arXiv: 10.1.1.163.5292 (see pp. 140, 151).

[258] X. Liu, N. Iftikhar, and X. Xie. "Survey of real-time processing systems for big data". In: *Proceedings of the 18th International Database Engineering & Applications Symposium on - IDEAS '14*. 2014, pp. 356–361. ISBN: 9781450326278. DOI: 10.1145/2628194.2628251 (see pp. 141, 142).

[259] U. Suthakar, L. Magnoni, D. R. Smith, and A. Khan. "Optimised lambda architecture for monitoring WLCG using spark and spark streaming". In: *2016 IEEE Nuclear Science Symposium, Medical Imaging Conference and Room-Temperature Semiconductor Detector Workshop (NSS/MIC/RTSD)*. IEEE, Oct. 2016, pp. 1–2. ISBN: 978-1-5090-1642-6. DOI: 10.1109/NSSMIC.2016.8069637 (see p. 141).

[260] M. Strohbach, H. Ziekow, V. Gazis, and N. Akiva. "Towards a Big Data Analytics Framework for IoT and Smart City Applications". In: *Modeling and Processing for Next-Generation Big-Data Technologies. Modeling and Optimization in Science and Technologies*. 4th ed. Springer, Cham, 2015, pp. 257–282. ISBN: 9783319091778. DOI: 10.1007/978-3-319-09177-8_11 (see p. 141).

[261] X. Liu, N. Iftikhar, P. S. Nielsen, and A. Heller. "Online anomaly energy consumption detection using Lambda architecture". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9829 LNCS. Springer, Cham, 2016, pp. 193–209. ISBN: 9783319439457. DOI: 10.1007/978-3-319-43946-4_13 (see p. 141).

[262] V. Dorokhov. *Applying Lambda Architecture on Azure*. Web page. 2017. URL: `https://www.codeproject.com/Articles/1171443/Applying-Lambda-Architecture-on-Azure` (visited on July 20, 2018) (see p. 143).

[263] S. Jajodia, S. Noel, P. Kalapa, M. Albanese, and J. Williams. "Cauldron Mission-centric Cyber Situational Awareness with Defense in Depth". In: *2011 - MILCOM 2011 Military Communications Conference*. Nov. 2011, pp. 1339–1344. DOI: `10.1109/MILCOM.2011.6127490` (see pp. 147, 148).

[264] F. De Gaspari, S. Jajodia, L. V. Mancini, and A. Panico. "AHEAD: A New Architecture for Active Defense". In: *Proceedings of the 2016 ACM Workshop on Automated Decision Making for Active Cyber Defense*. SafeConfig '16. New York, NY, USA: ACM, 2016, pp. 11–16. ISBN: 978-1-4503-4566-8. DOI: `10.1145/2994475.2994481` (see p. 148).

[265] D. Tovarnak. "Normalization of Unstructured Log Data into Streams of Structured Event Objects". PhD thesis. Masaryk University, Czech republic, 2018, p. 200. URL: `https://is.muni.cz/th/rjfzq/` (see p. 150).

A
List of Authored Publications

Impacted Journals

[1] T. Jirsik, M. Cermak, D. Tovarnak, and P. Celeda. "Toward Stream-based IP Flow Analysis". In: *IEEE Communications Magazine* 55.7 (2017), pp. 70–76. ISSN: 01636804. DOI: `10.1109/MCOM.2017.1600972`
- Impact Factor: **9.27**, WoS Ranking: D1, Contribution: **40 %**

[2] M. Husak, M. Cermak, T. Jirsik, and P. Celeda. "HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting". In: *Eurasip Journal on Information Security* 2016.1 (2016). ISSN: 2510523X. DOI: `10.1186/s13635-016-0030-7`
- SJR ranking: **0.315**, Contribution: **20 %**

Conference Proceedings

[1] T. Jirsik, S. Trcka, and P. Celeda. "Quality of Service Forecasting with LSTM Neural Network". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 1–8. Accepted for publication
- CORE ranking: A, Technical Session, Contribution: **65 %**

[2] M. Cermak, T. Jirsik, P. Velan, J. Komarkova, S. Spacek, M. Drasar, and T. Plesnik. "Towards Provable Network Traffic Measurement and Analysis via Semi-Labeled Trace Datasets". In: *2018 Network Traffic Measurement and Analysis Conference (TMA)*. Vienna, Austria, 2018: IEEE, June 2018, pp. 1–8. ISBN: 978-3-903176-09-6. DOI: `10.23919/TMA.2018.8506498`
- CORE ranking: N/A, Main session, Contribution: **15 %**

[3] T. Jirsik and P. Celeda. "Toward Real-time Network-wide Cyber Situational Awareness". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, 2018, p. 7. DOI: `10.1109/NOMS.2018.8406166`
- CORE ranking: **B**, Mini conference, Contribution: **90 %**

[4] T. Jirsik. "Stream4Flow: Real-time IP Flow Host Monitoring using Apache Spark". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, 2018, p. 2. DOI: `10.1109/NOMS.2018.8406132`
- CORE ranking: **B**, Demo paper, Contribution: **100 %**

[5] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky. "Passive OS Fingerprinting Methods in the Jungle of Wireless Networks". In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. Taipei, Taiwan: IEEE, Apr. 2018, pp. 1–9. ISBN:

978-1-5386-3416-5. doi: `10.1109/NOMS.2018.8406262`
- CORE ranking: **B**, Technical session, Contribution: **10 %**

[6] P. Velan, J. Medkova, T. Jirsik, and P. Celeda. “Network Traffic Characterisation using Flow-based Statistics”. In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 907–912. isbn: 9781509002238. doi: `10.1109/NOMS.2016.7502924`
- CORE ranking: **B**, Experience session, Contribution: **15 %**

[7] M. Cermak, T. Jirsik, and M. Lastovicka. “Real-time Analysis of NetFlow Data for Generating Network Traffic Statistics using Apache Spark”. In: *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*. 2016, pp. 1019–1020. isbn: 9781509002238. doi: `10.1109/NOMS.2016.7502952`
- CORE ranking: **B**, Demo paper, Contribution: **30 %**

[8] T. Jirsik, M. Cermak, and P. Celeda. “On Information Value of Top N Statistics”. In: *2016 6th International Conference on IT Convergence and Security, ICITCS 2016*. 2016, pp. 1–5. isbn: 9781509037643. doi: `10.1109/ICITCS.2016.7740357`
- CORE ranking: **N/A**, Main session, Contribution: **70 %**

[9] M. Husak, M. Cermak, T. Jirsik, and P. Celeda. “Network-based HTTPS Client Identification using SSL/TLS Fingerprinting”. In: *Proceedings of 10th International Conference on Availability, Reliability and Security, ARES 2015*. 2015, pp. 389–396. isbn: 9781467365901. doi: `10.1109/ARES.2015.35`
- CORE ranking: **N/A**, IWCC Workshop associated with ARES conference (CORE ranking: B), Contribution: **20 %**

[10] D. Kouril, T. Rebok, T. Jirsik, J. Cegan, M. Drasar, M. Vizvary, and J. Vykopal. “Cloud-based Testbed for Simulation of Cyber Attacks”. In: *Proceedings of IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*. 2014, pp. 1–6. isbn: 9781479909131. doi: `10.1109/NOMS.2014.6838298`
- CORE ranking: **B**, Experience session, Contribution: **14 %**

[11] T. Jirsik, M. Husak, P. Celeda, and Z. Eichler. “Cloud-based Security Research Testbed: A DDoS Use Case”. In: *Proceedings of IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*. 2014, pp. 1–2. isbn: 9781479909131. doi: `10.1109/NOMS.2014.6838272`
- CORE ranking: **B**, Demo paper, Contribution: **35 %**

[12] M. Drasar, T. Jirsik, and M. Vizvary. “Enhancing Network Intrusion Detection by Correlation of Modularly Hashed Sketches”. In: *Monitoring and Securing Virtualized Networks and Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 160–172. isbn: 978-3-662-43862-6
- CORE ranking: N/A, Main session, Contribution: **20 %**

[13] T. Jirsik and P. Celeda. “Enhancing Network Security: Host Trustworthiness Estimation”. In: *Monitoring and Securing Virtualized Networks and Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 63–68. isbn: 978-3-662-43862-6
- CORE ranking: N/A, Ph.D. session, Contribution: **80 %**

[14] T. Jirsik and P. Celeda. “Identifying Operating System Using Flow-Based Traffic Fingerprinting”. In: *Advances in Communication Networking*. Cham: Springer International Publishing, 2014, pp. 70–73. isbn: 978-3-319-13488-8
- CORE ranking: N/A, Main session, Contribution: **80 %**

[15] P. Velan, T. Jirsik, and P. Celeda. “Design and Evaluation of HTTP Protocol Parsers for IPFIX Measurement”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 8115. Springer Berlin Heidelberg, 2013, pp. 136–147. isbn: 978-3-642-40552-5. doi: `10.1007/978-3-642-40552-5_13`
• CORE ranking: N/A, Main session, Contribution: **30 %**

[16] M. Elich, P. Velany, T. Jirsik, and P. Celeda. “An Investigation into Teredo and 6to4 Transition Mechanisms: Traffic Analysis”. In: *Proceedings of Conference on Local Computer Networks, LCN*. 2013, pp. 1018–1024. doi: `10.1109/LCNW.2013.6758546`
• CORE ranking: N/A, WNM Workshop associated with LCN conference (CORE ranking: A), Contribution: **20 %**