IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 15, 2020, accepted June 3, 2020, date of publication June 11, 2020, date of current version June 29, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3001768

ASNM Datasets: A Collection of Network Attacks for Testing of Adversarial Classifiers and Intrusion Detectors

IVAN HOMOLIAK, KAMIL MALINKA, AND PETR HANACEK
Centre of Excellence IT4Innovations, Faculty of Information Technology, Brno University of Technology, 612 00 Brno, Czech Republic

Corresponding author: Ivan Homoliak (ihomoliak@fit.vutbr.cz)

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science–LQ1602.

ABSTRACT In this paper, we present three datasets that have been built from network traffic traces using ASNM (Advanced Security Network Metrics) features, designed in our previous work. The first dataset was built using a state-of-the-art dataset CDX 2009 that was collected during a cyber defense exercise, while the remaining two datasets were collected by us in 2015 and 2018 using publicly available network services containing buffer overflow and other high severity vulnerabilities. These two datasets contain several adversarial obfuscation techniques that were applied onto malicious as well as legitimate traffic samples during ''the execution'' of their TCP network connections. Adversarial obfuscation techniques were used for evading machine learning-based network intrusion detection classifiers. We show that the performance of such classifiers can be improved when partially augmenting their training data by samples obtained from obfuscation techniques. In detail, we utilized tunneling obfuscation in HTTP(S) protocol and non-payload-based obfuscations modifying various properties of network traffic by, e.g., TCP segmentation, re-transmissions, corrupting and reordering of packets, etc. To the best of our knowledge, this is the first collection of network traffic data that contains adversarial techniques and is intended for non-payload-based network intrusion detection and adversarial classification. Provided datasets enable testing of the evasion resistance of arbitrary machine learning-based classifiers.

INDEX TERMS Dataset, network intrusion detection, adversarial classification, evasions, ASNM features, buffer overflow, non-payload-based obfuscations, tunneling obfuscations.

I. INTRODUCTION

Network intrusions are one of the most dangerous threats in the domain of information security [1], [2]. Traditionally, there exist two orthogonal approaches to building intrusion detectors according to the input training data: (1) *misuse-based* (a.k.a., knowledge-based) detection, which models characteristics of malicious intrusions and then match them, and (2) *anomaly-based* detection, which models normal behaviors and detects deviations from them [3]. Nevertheless, an increasingly popular approach in recent research (e.g., [4]–[9]) is to build an intrusion detector that models both malicious and legitimate behaviors at the same time, and we refer to it as (supervised) *classification-based* approach.[1]

Each of these approaches has its respective pros and cons (see Table 1). Due to increasing sophistication in the techniques used by attackers, misuse-based intrusion detection suffers from undetected attacks such as zero-day attacks or polymorphism, enabling an exploit-code to avoid positive signature matching of the packet payload data.

Therefore, researchers and developers are motivated to design new methods to detect various versions of the modified network attacks including the zero-day ones. These goals motivate the popularity of Anomaly Detection Systems (ADS) and also the classification-based approaches in the

The associate editor coordinating the review of this manuscript and approving it for publication was Chi-Yuan Chen.

[1]Note that classifiers might be utilized even in the setting of a one-class classification. However, a trend is to leverage the maximum information from both legitimate and malicious samples (a.k.a., binary classification).

Table 1. Approaches to intrusion detection.

Detection Principle	Training Data	Pros	Cons
Misused-Based	malicious	• low FPR • high throughput	• high FNR in the case of evasions by unknown/zero-day attacks or polymorphism of existing attacks
Anomaly-Based	legitimate	• detection of unknown attacks	• high FPR, causing a lot of effort for manual investigation • long time required for profiling • re-profiling once in a while • susceptibility to many obfuscations
Classification-Based	malicious + legitimate	• detection of unknown attacks similar to known ones • lower FPR than anomaly-based	• long time required for training • re-training once in a while • susceptibility to some obfuscations • low throughput

context of intrusion detection. Anomaly-based approaches are based on building profiles of normal users, and they try to detect anomalies deviating from these profiles [3], which might lead to the detection of unknown intrusions, but on the other hand it might also generate many false positives. In contrast, the classification-based approaches take advantage of both misuse-based and anomaly-based models in order to leverage their respective advantages. The classification-based detection methods first build a model based on the labeled samples from both classes – intrusions and the legitimate instances. Second, they compare a new input to the model and select the more similar class as the predicted label. Classification and anomaly-based approaches are capable of detecting some unknown intrusions, but at the same time, they may be susceptible to evasion by obfuscation techniques.

In this paper, we present Advanced Security Network Metrics (ASNM) datasets, a collection of malicious and benign network traffic data. ASNM datasets include records consisting of several features that express miscellaneous properties and characteristics of TCP communications (i.e., aggregated bidirectional flows). These features are called Advanced Security Network Metrics (ASNM) and were designed in our previous work [10] with the intention to distinguish between legitimate and malicious TCP connections (i.e., intrusions and C&C channels of malware). ASNM features are extracted from tcpdump [11] traces and do not perform deep packet inspection during their computation, which makes them suitable for passive monitoring of (potentially encrypted) network traffic.[2]

A simplified overview of constructing ASNM datasets is depicted in Figure 1. We performed ASNM feature extraction over three different subsets of network traffic collections, resulting in three sub-datasets that we introduce:

- **ASNM-CDX-2009 Dataset**: was created from tcpdump traces of CDX 2009 dataset [12]. The dataset misses a few newer ASNM features and does not contain any obfuscations of the network traffic (see Section IV-A).
- **ASNM-TUN Dataset:** was created with the intention to evade and improve machine learning classifiers, and besides legitimate network traffic samples, it contains tunneling obfuscation technique [13] applied onto malicious network traffic, in which several vulnerable network services were exploited (see Section IV-B).
- **ASNM-NPBO Dataset:** Similar to the previous dataset, the current dataset was created with the intention to evade and improve machine learning classifiers. The dataset contains non-payload-based obfuscation techniques (modifying the properties of network flows) applied onto malicious traffic and onto several samples of legitimate traffic (see Section IV-C).

We made all ASNM datasets available for download at http://www.fit.vutbr.cz/~ihomoliak/asnm/.

A. CONTRIBUTIONS

We made several contributions in this work, which are summarized as follows:

- We formally define the process of ASNM feature extraction from network traces, and we briefly describe categories of ASNM features.
- We introduce ASNM datasets one-by-one, while in each dataset we describe underlying network infrastructure and vulnerable network services that were exploited during the capture of network traces.
- In the case of ASNM-TUN and ASNM-NPBO datasets, we also describe the methodology of executing the obfuscations, which, first serve as a means to evade (and improve) a detection by existing methods.
- To verify the effect of obfuscations, we test two signature-based NIDSs on obfuscated attacks from

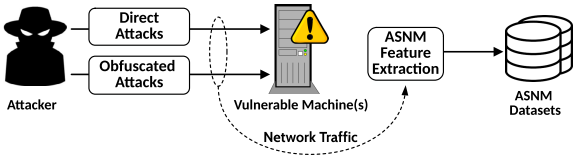


Figure 1. An overview of constructing ASNM datasets.

[2]Note that ASNM features can be directly extracted from the network traffic, and they do not perform a decryption since it would be a subject to privacy issues.

ASNM-TUN and ASNM-NPBO datasets, and we observe that they are successfully evaded in some cases.
- We benchmark the datasets in the experiments aimed at the adversarial classification with a few supervised binary classifiers, where we demonstrate the negative effect of obfuscations on the classification performance.
- Finally, we show that augmenting the training data of classifiers by a subset of obfuscated samples improves their performance.

B. ORGANIZATION

The rest of the paper is organized as follows: In Section II, we define the classification problem in intrusion detection and describe preliminaries and terms used throughout the paper. In Section III, we formally define ASNM features and describe them. We introduce particular ASNM datasets in Section IV, and then we perform statistical analysis of the datasets in Section V. Consequently, we perform the benchmarking of the datasets in Section VI. In Section VII we discuss limitations of the proposed datasets as well as directions for future work. Then, in Section VIII, we summarize existing datasets applicable in the network intrusion detection, compare them to the ASNM datasets, and finally in Section IX we conclude the paper.

II. PROBLEM DEFINITION AND PRELIMINARIES

First, we define the scope of our work by introducing the network connection as an elementary data object that is used for building our datasets. Second, we describe the feature extraction process over a network connection object, which forms a sample/data record in our datasets. Then, we describe the intrusion detection classification task, representing the problem that is addressed by an arbitrary binary classifier given a dataset containing 2-class labels. This problem represents the main challenge of ASNM datasets, but the application of ASNM datasets can be straightforwardly extended to a multi-class classification problem in sub-datasets containing multi-class labels. The full description of preliminaries is available in the dissertation thesis [14] and [10].

A. TCP CONNECTION

Consider a session of a protocol at the application layer of the TCP/IP stack that serves for data transfer between the client/server based application. The interpretation of application data exchanges between client and server can be formulated, considering the TCP/IP stack up to the transport layer, by connection c that is constrained to the connection-oriented protocol TCP at L4, Internet protocol IP at L3, and Ethernet protocol at L2. The TCP connection c is represented by the tuple

$$c = (t_s, t_e, p_c, p_s, ip_c, ip_s, P_c, P_s),$$

which consists of the start and end timestamps t_s and t_e, ports of the client and the server p_c and p_s, IP addresses of the client and the server ip_c and ip_s, sets of packets sent by the client P_c, and by the server P_s, respectively (see details in Table 24 of Appendix). Sets P_c and P_s contain a number of packets, where each of them can be interpreted by the packet tuple

$$p = (t, size, eth_{src}, eth_{dst}, ip_{off}, ip_{ttl}, ip_p, ip_{sum}, ip_{src}, ip_{dst}, ip_{dscp}, tcp_{sport}, tcp_{dport}, tcp_{sum}, tcp_{seq}, tcp_{ack}, tcp_{off}, tcp_{flags}, tcp_{win}, tcp_{urp}, data).$$

The symbols of the tuple are described in Table 25 of Appendix. We assume that the payload of P_s and P_c is encrypted, and thus data of these packet sets are not accessible.

Each TCP connection has its beginning that is represented by a three-way handshake, in which, three packets that contain the same IP addresses (ip_s, ip_d), ports (p_s, p_d), and sequence/acknowledgment numbers (tcp_{seq}, tcp_{ack}) conforming to the specification of RFC 793[3] must be found. Similarly, each TCP connection has its end, which is defined by a three-way endshake or by an inactivity timeout.[4]

B. FEATURE EXTRACTION

At this time, we can express the characteristics of a TCP connection by network connection features. The features extraction process is defined as a function that maps a connection c into space of features F:

$$f(c) \mapsto F, \quad F = (F_1, F_2, \ldots, F_n), \tag{1}$$

where n represents the number of defined features. Each function f_i that extracts feature i is defined as a mapping of a connection c into feature space F_i:

$$f_i(c) \mapsto F_i, \quad i \in \{1, \ldots, n\}, \tag{2}$$

and each element[5] of codomain F_i is defined as

$$e = (e_0, \ldots, e_n),\ n \in \mathbb{N}_0, \quad e_i \in \mathbb{N} \mid e_i \in \mathbb{R} \mid e_i \in \Gamma^+, \quad i \in \{0, \ldots, n\}, \quad \Gamma = \{a - z, A - Z, 0 - 9\}, \tag{3}$$

where Γ^+ denotes positive iteration of the set Γ. Note that for demonstration purposes, we abstract in our formalization from the fact that some features of a network connection c can be extracted not only from c itself but in addition from metadata of c that are not part of c. For example, such metadata may represent ''neighboring'' network connections of c, which we later refer to as *a context* of c (see Section III).

In general, network connection features can be instantiated, for example, by discriminators of A. Moore [15], Kyoto 2006+ features [16], basic and traffic features[6] of KDD Cup'99 dataset [17], NetFlow features [18], or ASNM features [10], CICFlowMeter features [19], multi-layered network traffic features from BGU [20], or connection-less features [21].

[3] URL http://www.ietf.org/rfc/rfc793.txt, page 30.
[4] E.g., in Unix-based systems, such a timeout is equal to five days.
[5] Representing a particular dimension of a feature.
[6] Not content features, which work over payload of the network data.

C. INTRUSION DETECTION CLASSIFICATION TASK

A data sample of the dataset D_{tr} refers to the vector of the network connection features, defined in Section II-B. Then, referring to [22] and [23], let $X = V \times Y$ be the space of labeled samples, where V represents the space of unlabeled samples and Y represents the space of possible labels. Let $D_{tr} = \{x_1, x_2, \ldots, x_n\}$ be a training dataset consisting of n labeled samples, where

$$x_i = (v_i \in V,\ y_i \in Y). \tag{4}$$

Consider classifier C which maps unlabeled sample $v \in V$ to a label $y \in Y$:

$$y = C(v), \tag{5}$$

and learning algorithm A which maps the given dataset D to a classifier C:

$$C = A(D_{tr}). \tag{6}$$

The notation $y_{predict} = A(D_{tr}, v)$ denotes the label assigned to an unlabeled sample v by the classifier C, build by learning algorithm A on the dataset D_{tr}. Then, all extracted features $f()$ of an unknown connection c can be used as an input of the trained classifier C that predicts the target label:

$$y_{predict} = A\big(D_{tr}, f(c)\big), \tag{7}$$

where

$$y_{predict} \in \{Intrusion, Legitimate\}. \tag{8}$$

D. ADVERSARIAL OBFUSCATIONS & EVASION OF THE CLASSIFIER

Assume a connection c_m representing a malicious communication executed without any obfuscation. Then, c_m can be expressed by network connection features

$$f(c_m) \mapsto F^m = (F_1^m, F_2^m, \ldots, F_n^m) \tag{9}$$

that are delivered to the previously trained classifier C. Assume that C can correctly predict the target label as a malicious one, because its knowledge base is derived from training dataset D_{tr} containing features of malicious connections having similar (or the same) behavioral characteristics as c_m.

Now, consider connection c'_m that represents the malicious communication c_m executed by the employment of an obfuscation technique that is aimed at modification of network behavioral properties of the connection c_m. An obfuscation technique can modify P_c and P_s packet sets of the original connection c_m as well as IP addresses (ip_s, ip_d) and ports (p_s, p_d) of the original connection c_m.

Hence, network connection features extracted for c'_m are represented by

$$f(c'_m) \mapsto F^{m'} = (F_1^{m'}, F_2^{m'}, \ldots, F_n^{m'}) \tag{10}$$

and have different values in comparison to features F^m of the connection c_m. Therefore, we conjecture that the likelihood of a correct prediction of c'_m connection's features $F^{m'}$ by the previously assumed classifier C is lower than in the case of connection c_m, which might cause an evasion of the detection. Also, we conjecture that the classifier C' trained by learning algorithm A on training dataset D'_{tr}, containing obfuscated malicious instances, will be able to correctly predict higher number of unknown obfuscated malicious connections than classifier C. We will demonstrate the correctness of these assumptions in Section VI on two of our datasets.

III. ASNM FEATURES AND CONTEXT ANALYSIS

ASNM features [10] are network connection features that describe various properties of TCP connections and were designed with the intention to distinguish between legitimate traffic and remote buffer overflow attacks.[7] We studied behavioral characteristics of remote buffer overflow attacks in our previous work [24], and our findings inspired the design of ASNM features. We can interpret ASNM features like an extended protocol NetFlow [18] but describing more than statistical properties of network connections. In addition to NetFlow features, ASNM features represent dynamical, localization, and, most importantly, the behavioral properties of network connections. Moreover, some of the features utilize the context of an analyzed connection c, which represents ''neighboring'' connection objects (see Section III-A).

In the following, we assume an input dataset of network traffic traces, which is used for identification of network TCP connection objects $C = \{c_1, \ldots, c_M\}$, where M is a count of TCP connections in the dataset.

A. CONTEXT DEFINITION

We assume a dataset of TCP connection objects (as described in Section II) Considering analyzed TCP connection c_k, we define a sliding window sw of length τ as a set of TCP connections W_k that are delimited by $\pm\frac{\tau}{2}$:

$$\begin{aligned} sw(c_k, \tau) &= W_k \\ W_k &\subseteq C, \\ W_k &= \{c_j\}, \end{aligned} \tag{11}$$

where each TCP connection c_j must satisfy the following:

$$\begin{aligned} c_j[t_s] > c_k[t_e] - \frac{\tau}{2} \ \wedge\ c_j[t_s] \le c_k[t_f] + \frac{\tau}{2}, \\ c_j[t_e] > c_k[t_s] - \frac{\tau}{2} \ \wedge\ c_j[t_e] \le c_k[t_e] + \frac{\tau}{2}, \end{aligned} \tag{12}$$

which means that sliding window contains the union of: (1) all TCP connections c_j that contain their hand-shake within the boundaries of sw, and (2) all TCP connections c_j that contain their end-shake within the boundaries of sw.

The next fact about each particular TCP connection c_k is an unambiguous association of it to particular sliding window W_k. We can interpret the start time t_s of the TCP connection c_k as a center of the sliding window W_k. Then, we can denote a shift of the sliding window $\Delta(W_j)$ which

[7] See Appendix D of [14] for the full list of ASNM features.

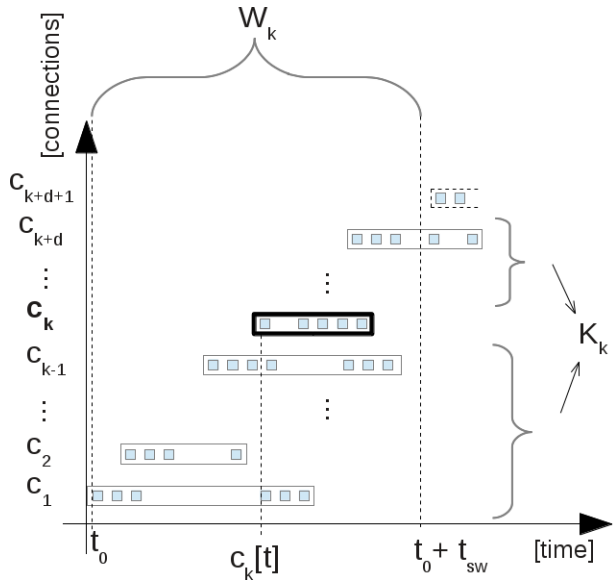


Figure 2. Sliding window and the context of the connection c_k [10].

is defined by start time differences of two consecutive TCP connections in C:

$$\Delta(W_j) = c_{j+1}[t_s] - c_j[t_s], \quad j \in \{1, \ldots, |C| - 1\}. \tag{13}$$

Next, we define the context K_k of the TCP connection c_k, which is a set of all connections in a particular sliding window W_k excluding analyzed TCP connection c_k:

$$K_k = \{c_1, \ldots, c_n\} = \{W_k \setminus c_k\}. \tag{14}$$

Defined terms are shown in Figure 2. In the figure, the x axis displays time, and the y axis represents TCP connections, which are shown in the order of their occurrences. Packets are represented by small squares, and TCP connections are represented by a rectangular boundary of particular packets. A bold line and bold font are used for depicting an analyzed TCP connection c_k, which has an associated sliding window W_k and context K_k. TCP connections, which are part of the sliding window W_k, are drawn by a full line boundary, and TCP connections, which are not part of this sliding window, are drawn by a dashed line boundary. We note that only a few features from the ASNM datasets utilize context; these features belong to the dynamic and behavioral categories (see Section III-C).

B. ASNM FEATURE EXTRACTION

In addition to a general definition of network connection feature extraction (see Section II-B), we incorporate the context of a TCP connection into the extraction process of ASNM features. The ASNM feature extraction is thus defined as a function that maps a connection c_k with its context $K_k = sw(c_k, \tau)$ into feature space F:

$$f(c_k, K_k) \mapsto F, \quad F = (F_1, F_2, \ldots, F_n), \tag{15}$$

where n represents the number of defined features, while the rest of the definition is inherited from Section II-B.

C. CATEGORIZATION OF ASNM FEATURES

The list of originally proposed ASNM feature set contains 167 features (see Master's thesis [25]), which are formally described in [10]. However, the ASNM feature set was later extended [14], resulting in 194 features. These 194 features are in many cases a result of reasonable parametrization of the base feature functions $f_i()$, We depict a categorization of our feature set in Table 2 together with their counts. We decided to determine the naming of particular categories of features according to their principles, not according to their data representation. In the following, we briefly describe each category.

1) STATISTICAL FEATURES

In this category of ASNM features, the statistical properties of TCP connections are identified. All packets of the TCP connection are considered in order to determine count, mode, median, mean, standard deviation, ratios of some header fields of packets, or the packets themselves. This category of features partially uses a time representation of packets occurrences, in contrast to the dynamic category (see below). Therefore, it includes particularly dynamic properties of the analyzed TCP connection, but without any context. Most of the features in this category also distinguish inbound and outbound packets of the analyzed TCP connection.

2) DYNAMIC FEATURES

Dynamic features were defined with the aim to examine dynamic properties of the analyzed TCP connection and transfer channel such as speed or an error rate. These properties can be real or simulated. Eighteen of the features consider the context of an analyzed TCP connection. The difference between some of the statistical and dynamic features from a dynamic view can be demonstrated on two instances of the same TCP connection, which performs the same packet transfers, but in different context conditions and with different packet retransmission and attempts to start or finish the TCP connection. Many of the defined features distinguish between the inbound and outbound direction of the packets and consider the statistical properties of the packets and their sizes, as mentioned in statistical features.

3) LOCALIZATION FEATURES

The main characteristic of the localization features category is that it contains static properties of the TCP connection.

Table 2. Categorization of ASNM features.

Category of ASNM Features	#
Statistical	77
Dynamic	32
Localization	8
Distributed	34
Behavioral	43

Table 3. A list of vulnerable servers in CDX 2009 dataset.

Service	OS	Internal IP	External IP
Postfix Email	FreeBSD	7.204.241.161	10.1.60.25
Apache Web Server	Fedora 10	154.241.88.201	10.1.60.187
OpenFire Chat	FreeBSD	180.242.137.181	10.1.60.73
BIND DNS	FreeBSD	65.190.233.37	10.1.60.5

These properties represent the localization of participating machines and their ports used for communication. In some features, the localization is expressed indirectly by a flag, which distinguishes whether participating machines lay in a local network or not. Features included in this category do not consider the context of the analyzed TCP connection, but they distinguish the direction of the analyzed TCP connection.

4) DISTRIBUTED FEATURES

The characteristic property of the distributed features category is the fact that they distribute packets or their lengths to a fixed number of intervals per unit of time, specified by a logarithmic scale (1s, 4s, 8s, 32s, 64s). The next principal property of this category is vector representation. All these features are supposed to work within the context of an analyzed TCP connection.

5) BEHAVIORAL FEATURES

Behavioral features represent properties associated with the behavior of an analyzed TCP connection. Examples include legal or illegal connection closing, the polynomial approximation of packet lengths in a time domain or an index of occurrence domain, count of new TCP connections after starting an analyzed TCP connection, coefficients of Fourier series with the distinguished direction of an analyzed TCP connection, etc.

IV. ASNM DATASETS

In this section, we detail three different datasets that have been built using ASNM features. The first of them was built using an existing dataset of network traffic traces, while the remaining two were collected by us, and they contain several adversarial obfuscation techniques that were applied onto malicious as well as legitimate samples during ''the execution'' of particular network connections. All intrusions contained in our dataset represent targeted attacks on a particular service running on a host, and they do not contain any Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks.

A. ASNM-CDX-2009 DATASET

ASNM-CDX-2009 dataset was build from CDX-2009 dataset [26], which was introduced by Sangster *et al.* [12] and it contains data in tcpdump format as well as SNORT [27] intrusion prevention logs, as relevant sources for our purpose.

Table 4. ASNM-CDX-2009 dataset distribution.

Network Service	Count of TCP Connections		
	Legitimate	Malicious	Summary
Apache	2911	37	2948
Postfix	179	7	186
Other Traffic	2637	–	2637
Summary	5727	44	5771

The CDX 2009 dataset was created during the network warfare competition, in which one of the goals was to generate a labeled dataset. By labeled dataset, the authors mean tcpdump traces of all simulated communications and SNORT log with information about occurrences of intrusions, deemed as the ground truth. Network infrastructure contained four servers with four vulnerable services (one per each server), while the authors provided two collections of network traces: 1) network traces captured outside the West Point network border and 2) network traces captured by National Security Agency (NSA). The services that run on the hosted servers together with IP addresses of the servers are listed in Table 3.

Two types of IP addresses are shown in this table:

- **Internal IP** addresses – corresponding to the SNORT log,
- **External IP** addresses – corresponding to a TCP network traces captured outside the West Point network border.

Note that specific versions of services described in [12] were not announced. We found out that SNORT log can be associated only with data capture outside of West point network border and only with significant timestamps differences – approximately 930 days. We have not found any association between SNORT log and data capture performed by NSA. We focused only on buffer overflow attacks found in SNORT log, and we performed a match with the packets contained in the West point network border capture.

Despite all the efforts, we matched only 44 buffer overflow attacks out of 65 entries in SNORT log. To correctly match SNORT entries, it was necessary to remap external IP addresses to internal ones, because SNORT detection was performed in external network and network traces contain entries with internal IP addresses. We found out that in CDX 2009 dataset, buffer overflow attacks were performed only on two services – Postfix Email and Apache Web Server.

The buffer overflow attacks that were matched with data capture have their content only in two files with network traces:

- *2009-04-21-07-47-35.dmp*
- *2009-04-21-07-47-35.dmp2*

Due to the high number of all packets (approx. 4 mil.) in all files of network traces, we decided to consider only these two files for the purpose of extraction of both malicious and legitimate samples (which together contain 1, 538, 182 packets). We also noticed that network data density was increased during the time when the attacks were performed. Consequently,

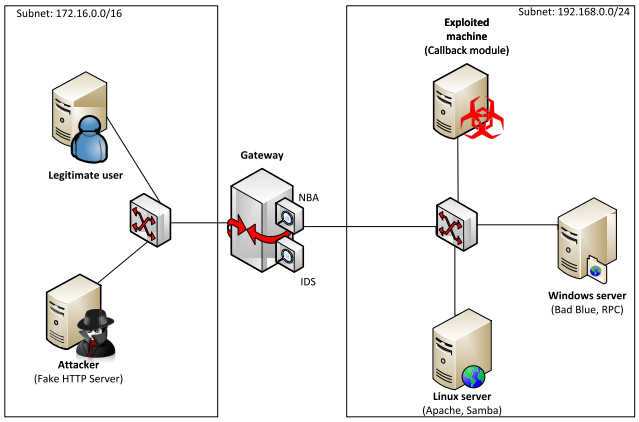


Figure 3. A setup of virtual network used in ASNM-TUN dataset.

we made another reduction of all packets considered so far, which filtered enough temporal neighborhood of all attacks occurrences, and at the same time, included a sufficient number of legitimate TCP connections. In the result, we used 204 953 packets for the extraction of ASNM features.

A distribution of malicious and legitimate samples within the obtained dataset is presented in Table 4. Beside two services that contained buffer overflow vulnerabilities, our dataset also contains samples representing other network traffic, which we consider as legitimate since no match of its metadata with SNORT log was determined.

1) LABELING

ASNM-CDX-2009 dataset contains two types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2**: is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or legitimate traffic.
- **label_poly**: is composed of two parts that are delimited by a separator: (a) a two-class label where legitimate and malicious communications are represented by symbols 0 and 1, respectively, and (b) an acronym of network service. This label represents the type of communication on a particular network service.

This dataset was for the first time used and evaluated in [10].

B. ASNM-TUN DATASET

ASNM-TUN[8] dataset was build in laboratory conditions[9] using a custom virtual network architecture (see Figure 3), where we simulated malicious TCP connections on a few selected vulnerable network services [13]. The selected vulnerabilities are presented in Table 5, which also contains Common Vulnerabilities and Exposures (CVE) IDs and Common Vulnerability Scoring System (CVSS) values. A selection of the vulnerable services was aimed at a high severity of their successful exploitation, namely a presence of buffer overflow vulnerabilities. The exploitation of such vulnerabilities usually led to a remote shellcode execution through an established backdoor connection, while as a consequence of successful exploitation, the attacker was able to get the root access. The details about each vulnerability and its exploitation are briefly described in the following listing:

[8]The name is derived from **TUN**elling obfuscations.

[9]Note that a part of the legitimate traffic samples was extracted from anonymized metadata collected from a real network.

Table 5. A list of vulnerable services in ASNM-TUN dataset.

Service	CVE	CVSS
Apache Tomcat	2002-0082	7.5
BadBlue	2007-6377	7.5
DCOM RPC	2003-0352	7.5
Samba	2003-0201	10.0

- **Apache web server with `mod_ssl` plugin 2.8.6:** This attack exploits a buffer overflow vulnerability in `mod_ssl` plugin of the Apache web server. The plugin does not correctly initialize memory in the `i2d_SSL_SESSION` function, which allows a remote attacker to exploit a buffer overflow vulnerability in order to execute arbitrary code via a large client certificate that is signed by a trusted Certification Authority, which produces a large serialized session [28]. This allows remote code execution and modification of any file on a compromised system [29]. The vulnerable versions of the plugin are in range 2.7.1-2.8.6.
- **BadBlue web server 2.72b:** The second attack exploits a stack-based buffer overflow vulnerability in `PassThru` functionality of `ext.dll` in BadBlue 2.72b and earlier [30]. In the attack performing phase, the specially crafted packet with a long header is sent, which leads to an overflow of processing buffer [31]
- **Microsoft DCOM RPC:** The third attack exploits a vulnerability in Microsoft Windows DCOM Remote Procedure Call (DCOM RPC) service of Microsoft Windows NT 4.0, 2000 (up to Service Pack 4), Server 2003, and XP [32]. This vulnerability allows a remote attacker to execute an arbitrary code after a buffer overflow in the DCOM interface. The vulnerability was originally found by the Last Stage of Delirium research group and has been widely exploited since then [33]. The vulnerability is well documented and was used, e.g., by the Blaster worm.
- **Samba service 2.2.7:** The last attack exploits a buffer overflow vulnerability in `call_ trans2open` function in `trans2.c` of Samba 2.2.x before 2.2.8a, 2.0.10, earlier versions than 2.0.x and Samba-TNG before 0.3.2 [34]. This vulnerability allows a remote attacker to execute arbitrary code. An exploit code sends malformed packets to a remote server in batches [35]. Packets differ in one shell-code address only because the return address depends on versions of Samba and host operating systems.

1) ADVERSARIAL MODIFICATIONS

We employed tunneling of malicious network traffic inside of HTTP and HTTPS protocols, serving as obfuscation

techniques when exploiting vulnerable services. The tunneling obfuscation modifies P_c and P_s packet sets (see Section II-A) of the original malicious connection c_m by wrapping each original packet into a new one. Assuming the background from Section II-D, the tunneling (i.e., wrapping) may cause fragmentation of IP packets, and thus it can also modify the number of packets in both packet sets P_c and P_s. Also, the obfuscation modifies IP addresses (ip_c, ip_s) and ports (p_c, p_s) of the original connection. Symbols of the packet tuple whose values are sensitive to the obfuscation include all defined fields, as tunneling obfuscation creates new TCP/IP stack with unique values of L2, L3, L4 headers as well as new content of application layer data. All these modifications, especially modifications of P_c and P_s of the connection c_m, cause alteration of the original network connection features' values (see Section II-D).

For the purpose of simulating real network conditions, we executed each malicious and legitimate network communication four times with four different network traffic modifications. Network traffic modifications differ in the alteration degree of the network traffic, and we divide them into four categories:

(a) **No Modification:** The first category represents reference output without any modification. All experiments ran on the same host machine to minimize deviations among different tests.
(b) **Traffic Shaping:** The second category is dedicated to the simulation of traffic shaping. Therefore, all packets were forwarded with higher time delays. For this purpose, the special gateway machine with a limited processor's performance was used. This machine was also fully loaded to emulate slower packets processing than in the first scenario.
(c) **Traffic Policing:** The third category is supposed to simulate traffic policing when some of the packets were dropped during the processing on the network gateway node. In this case, a custom packet dropper was used on the gateway node, and 25% of packets were dropped, resulting in output that contained re-transmitted packets.
(d) **Corrupted Traffic:** The fourth category represents transmission on an unreliable network channel; thus, 25% of packets were corrupted during processing on the network gateway node.

2) LEGITIMATE NETWORK TRAFFIC

Legitimate samples of the dataset were collected from two sources. The first source represents a legitimate traffic simulation in our virtual network architecture and also employed network traffic modifications for the purpose of simulating real network conditions. As the second source, common usage of selected services was captured in the campus network in accordance with policies in force. In the obtained data, no content of packets was captured, and all collected metadata was anonymized. Further, we filter out data matched on high severity alerts by signature-based Network Intrusion Detection Systems (NIDSs) Suricata [36] and SNORT [27] through Virus Total API. This step assured that legitimate traffic does not contain any malicious data. Note that SNORT was equipped with *Sourcefire VRT* ruleset, and SURICATA utilized *Emerging Threats ETPro* ruleset. The final composition of the dataset after extraction of ASNM features is depicted in Table 6.

Table 6. ASNM-TUN dataset distribution.

Network Service	Count of TCP Connections			
	Legitimate	Direct Attacks	Obfuscated Attacks	Summary
Apache	38	102	61	201
BadBlue	95	4	10	109
DCOM RPC	4	4	8	16
Samba	15	20	8	43
Other Traffic	25	–	–	25
Summary	177	130	87	394

3) LABELING

ASNM-TUN dataset contains four types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2:** is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or a legitimate communication.
- **label_3:** is a three-class label, which distinguishes among legitimate traffic (symbol 3), direct attacks (symbols 1), and obfuscated network attacks (symbol 2).
- **label_poly:** is a label that is composed of 2 parts: (a) a three-class label, and (b) acronym of a network service. This label represents a type of communication on a particular network service.
- **label_poly_s:** is composed of 3 parts: (a) a three-class label, (b) an acronym of network service, and (c) a network modification technique employed. This label has almost the same interpretation as the previous one, but in addition, it introduces a network traffic modification technique (identified by a letter from the previous listing).

4) TESTING WITH SIGNATURE-BASED NIDS

To investigate the effect of the tunneling obfuscation on signature-based NIDSs, we performed detection by SNORT and SURICATA through VirusTotal API [37]. SNORT was equipped with Sourcefire VRT ruleset, and SURICATA utilized Emerging Threats ETPro ruleset. The results of direct attacks' detection by both NIDSs are shown in Table 7. Note that high priority rules detected 93 direct attacks on Apache service in both NIDSs. In addition, 4 undetected direct attack instances (i.e., TCP connections) occurred almost at the same time as some of the detected attack instances, which means that these 4 TCP connections are part of the same attacks that were already detected (we verified this by checking the value

Table 7. Detection of direct attacks in ASNM-TUN dataset by SNORT and SURICATA.

	Direct Attacks		
	Detected	**Total**	**%**
Apache	93 +4	102	95.10%
BadBlue	4	4	100.00%
DCOM RPC	4	4	100.00%
Samba	20	20	100.00%
Overall Detection	125	130	96.15%
ADR* per Service			98.77%

*Average detection rate.

(a) SNORT

	Direct Attacks		
	Detected	**Total**	**%**
Apache	93 +4	102	95.10%
BadBlue	4	4	100.00%
DCOM RPC	4	4	100.00%
Samba	20	20	100.00%
Overall Detection	125	130	96.15%
ADR* per Service			98.77%

*Average detection rate.

(b) SURICATA

of the label (i.e., *label_poly*) that matched the attack on the same service than the other one already detected). In addition, we emphasize that an attack might consist of multiple TCP connections, and not all of them might raise an alert. Also, we can see that five instances of direct attacks were not detected by SNORT nor SURICATA. These five instances utilized network traffic modifications (c) and (d), which likely influenced the detection rate of both NIDSs; hence, they give an intuition for the adversarial obfuscation techniques utilized in the last ASNM dataset (see Section IV-C). The resulting detection rates of direct attacks look the same in both NIDSs, but there were differences in fired alerts during the exploitation of Apache service. Unlike SNORT, SURICATA had not detected any occurrence of buffer overflow, nor shellcode, nor remote command execution but instead fired high priority alerts related to potential corporate privacy violation:

- `ET POLICY Possible SSLv2 Negotiation in Progress Client Master Key SSL2_RC4_128_WITH_MD5,`

which we decided to consider as correct detection. If we were to not consider them as correctly detected, then SURICATA would not detect any direct attack on the Apache service.

Next, we have performed exploitation of each vulnerable service using the tunneling obfuscation, while scanning the network by aforementioned NIDSs. The obtained results are depicted in Table 8, which distinguishes between tunneling obfuscation performed through HTTP and HTTPS protocols. We can see that an average detection rate per service is significantly lower for obfuscated attacks than in the case of direct attacks, and thus tunneling obfuscation was partially capable of evading detection by utilized NIDSs. Regarding tunneling

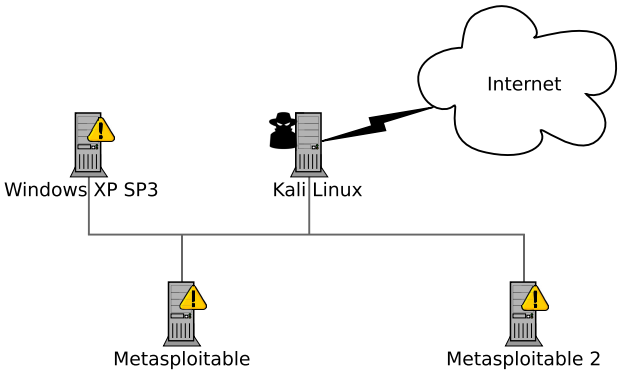


Figure 4. A setup of virtual network used in ASNM-NPBO dataset.

through the HTTP protocol, both SNORT and SURICATA achieved the same low detection rate for all classes of attacks.

The situation is slightly different in the case of tunneling through the HTTPS protocol. The SNORT achieved an average detection rate (ADR) per class equal to 68.75% and SURICATA only 23.25%. We found out the same fact about high priority rules fired by SURICATA on the exploitation of Apache service as in the case of direct attacks detection – neither buffer overflow, nor shellcode, nor remote command execution rules were matched, and thus we decided to accept the previously mentioned potential corporate privacy violation alert as correct detection again. If we were to not accept it, then SURICATA would not detect any tunneled attack on Apache service. Also note that SURICATA fired one non-high-priority alert classified as potentially bad traffic in several instances of attacks tunneled through HTTPS, which exploited BadBlue, DCOM and Samba services:

- `ET POLICY FREAK Weak Export Suite From Client (CVE-2015-0204).`

But we have not considered it as correct detection due to the low priority of the alert as well as the scope of corresponding CVE-2015-0204 is only related to the client code of OpenSSL. The plus notation in Table 8, similar to Table 7, denotes undetected attacks that occurred almost at the same time as some other correctly detected attacks, and thus are considered as their parts. Concluding the results of NIDSs detection, we can state that the proposed tunneling obfuscation technique was successful in evading the NIDSs used since a high number of obfuscated attacks were not detected in comparison to the case where obfuscations were not employed. On the other hand, we emphasize that SNORT has detected the most of direct attacks on Apache service even though it was encrypted.[10] This indicates that VirusTotal may utilize a very paranoid rule set, which causes false positives. Hence, the results of the analysis through VirusTotal API are arguable.

[10]Note that SNORT does not decrypt the traffic but just utilize some patterns presented in the network and transport layers to fire an alert.

Table 8. Detection of obfuscated attacks in ASNM-TUN dataset by SNORT and SURICATA.

Service	Obfuscated Attacks								
	HTTP			HTTPS			All		
	Detected	Total	%	Detected	Total	%	Detected	Total	%
Apache	0	4	0.00	51 +6	57	100.00	57	61	93.40
BadBlue	3	6	50.00	2	4	50.00	5	10	50.00
DCOM	0	4	0.00	3	4	75.00	3	8	37.50
Samba	0	4	0.00	2	4	50.00	2	8	25.00
Summary	3	18	16.67	64	69	92.75	67	87	77.01
ADR*			12.50			68.75			51.49

*Average detection rate per class.

(a) SNORT

Service	Obfuscated Attacks								
	HTTP			HTTPS			All		
	Detected	Total	%	Detected	Total	%	Detected	Total	%
Apache	0	4	0.00	50 +3	57	92.98	53	61	86.89
BadBlue	3	6	50.00	0	4	0.00	3	10	30.00
DCOM	0	4	0.00	0	4	0.00	0	8	0.00
Samba	0	4	0.00	0	4	0.00	0	8	0.00
Summary	3	18	16.67	53	69	76.81	56	87	64.37
ADR*			12.50			23.25			29.22

*Average detection rate per class.

(b) SURICATA

C. ASNM-NPBO DATASET

ASNM-NPBO[11] dataset was built in laboratory conditions using a virtual network architecture (see Figure 4) consisting of three vulnerable machines and the attacker's machine.

All virtual machines were configured with private static IP addresses in order to enable easy automation of the whole exploitation process. Our testing network infrastructure consisted of the attacker's machine equipped with Kali Linux and vulnerable machines that were running Metasploitable 1, 2 [38], and Windows XP with SP 3. We aimed at the selection of vulnerable services with the high severity of their successful exploitation leading to remote shellcode execution through an established backdoor communication. All selected vulnerable services are depicted in Table 9, which also contains CVE IDs and CVSS severity score values. The details about each vulnerability and its exploitation are briefly described in the following:

- **Apache Tomcat 5.5:** First, a dictionary attack was executed in order to obtain access credentials into the application manager's instance [39]. Further, the server's application manager was exploited for the transmission and execution of malicious code [40].

Table 9. A list of vulnerable services in ASNM-NPBO dataset.

Service	CVE	CVSS
Apache Tomcat	2009-3843	10.0
DistCC service	2004-2687	9.3
MSSQL	2000-1209	10.0
PostgreSQL	2007-3280	9.0
Samba service	2007-2447	6.0
Server service of Windows	2008-4250	10.0

[11]The name is derived from **N**on-**P**ayload-**B**ased **O**bfuscations.

- **Microsoft SQL Server 2005:** A dictionary attack was employed to obtain access credentials of MSSQL user [41] and then the procedure `xp_cmdshell` enabling the execution of an arbitrary code was exploited [42].
- **Samba 3.0.20-Debian:** A vulnerability in Samba service enabled the attacker of arbitrary command execution, which exploited MS-RPC functionality when `username_map_script` [43] was allowed in the configuration. There was no need for authentication in this attack.
- **Server Service of Windows XP:** The server service enabled the attacker of arbitrary code execution through crafted RPC request resulting in stack overflow during path canonicalization [44].
- **PostgreSQL 8.3.8:** A dictionary attack was executed in order to obtain access credentials into the PostgreSQL instance [45]. Standard PostgreSQL Linux installation had write access to `/tmp` directory, and it could call user-defined functions (UDF). UDFs utilized shared libraries located on an arbitrary path (e.g., `/tmp`). An attacker exploited this fact and copied its own UDF code to `/tmp` directory and then executed it [46].
- **DistCC 2.18.3:** A vulnerability enabled the attacker remote execution of an arbitrary command through compilation jobs that were executed on the server without any permission check [47].

1) ADVERSARIAL MODIFICATIONS

We proposed several non-payload-based obfuscation techniques [48] when exploiting vulnerable network services as well as during the execution of legitimate communications on the services. The proposed non-payload-based obfuscation

Table 10. Non-payload-based obfuscation techniques with parameters and IDs.

Technique	Parametrized Instance	ID
Spread out packets in time	• constant delay: 1s • constant delay: 8s • normal distribution of delay with 5s mean 2.5s standard deviation (25% correlation)	(a) (b) (c)
Packets' loss	• 25% of packets	(d)
Unreliable network channel simulation	• 25% of packets damaged • 35% of packets damaged • 35% of packets damaged with 25% correlation	(e) (f) (g)
Packets' duplication	• 5% of packets	(h)
Packets' order modifications	• reordering of 25% packets; reordered packets are sent with 10ms delay and 50% correlation • reordering of 50% packets; reordered packets are sent with 10ms delay and 50% correlation	(i) (j)
Fragmentation	• MTU 1000 • MTU 750 • MTU 500 • MTU 250	(k) (l) (m) (n)
Combinations	• normal distribution delay ($\mu = 10ms$, $\sigma = 20ms$) and 25% correlation; loss: 23% of packets; corrupt: 23% of packets; reorder: 23% of packets • normal distribution delay ($\mu = 7750ms$, $\sigma = 150ms$) and 25% correlation; loss: 0.1% of packets; corrupt: 0.1% of packets; duplication: 0.1% of packets; reorder: 0.1% of packets • normal distribution delay ($\mu = 6800ms$, $\sigma = 150ms$) and 25% correlation; loss: 1% of packets; corrupt: 1% of packets; duplication: 1% of packets; reorder 1% of packets	(o) (p) (q)

techniques are described in Table 10. Assuming the background from Section II-D, the proposed non-payload-based obfuscation techniques can modify P_c and P_s packet sets of the original connection c_m by insertion, removal, and transformation of the packets. Symbols of the packet tuple (see Table 25) whose values are sensitive to the obfuscations include: t, $size$, ip_{off}, ip_{sum}, tcp_{sum}, tcp_{seq}, tcp_{ack}, tcp_{off}, tcp_{flags}, tcp_{win}, tcp_{urp} and $data$.[12] The modifications of P_c and P_s of the connection c_m can cause alteration of values of the original network connection features F^m to new ones (see Section II-D).

Then we built an obfuscation tool [23] that morphs network characteristics of a TCP connection at network and transport layers of the TCP/IP stack by applying one or a combination of several non-payload-based obfuscation techniques. Execution of direct communications (non-obfuscated ones) is also supported by the tool as well as capturing network traffic related to communication. The tool is capable of automatic/semi-automatic run and restoring of all modified system settings and consequences of attacks/legitimate

[12]Note the *data* field is sensitive to the obfuscations only in the manner of damaging or splitting the original packet's data.

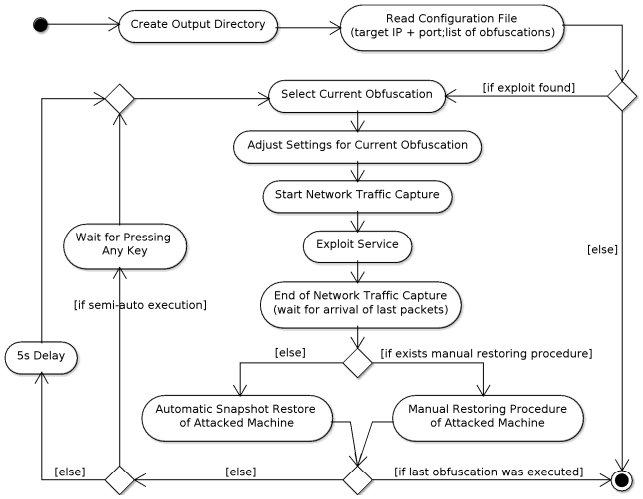


Figure 5. Behavioral state diagram of the obfuscation tool [23].

communications on a target machine. After the successful execution of each desired obfuscation on the selected service, the output of the tool contains several network packet traces associated with pertaining obfuscations. The behavioral state diagram of the obfuscation tool is depicted in Figure 5.

We applied our obfuscation tool for automatic exploitation of all enumerated vulnerable services while using the proposed obfuscations. When exploitation leading to a remote shell was successful, simulated attackers performed simple activities involving various shell commands (such as listing directories, opening, and reading files). An average number of issued commands was around 10, and text files of up to 50kB were opened/read. Note that we labeled each TCP connection representing dictionary attacks as legitimate ones due to two reasons: 1) from the behavioral point of view, they independently appeared just as unsuccessful authentication attempts, which may occur in legitimate traffic as well, 2) more importantly, we employed ASNM features whose subset involves the context of an analyzed TCP connection for their computation – i.e., ASNM features capture relations to other TCP connections initiated from/to a corresponding service.

Table 11. ANSM-NPBO dataset distribution.

Network Service	Count of TCP Connections			
	Legitimate	Direct Attacks	Obfuscated Attacks	Summary
Apache Tomcat	809	61	163	1033
DistCC	100	12	23	135
MSSQL	532	31	103	666
PostgreSQL	737	13	45	795
Samba	4641	19	44	4704
Server	3339	26	100	3465
Other Traffic	647	–	–	647
Summary	10805	162	478	11445

Table 12. Detection of direct attacks in the ASNM-NPBO dataset by SNORT and SURICATA.

	Direct Attacks		
	Detected	Total	%
Apache Tomcat	33 +28	61	100.00
DistCC	12	12	100.00
MSSQL	31	31	100.00
PostgreSQL	13	13	100.00
Samba	19	19	100.00
Server	26	26	100.00
Overall Detection	162	162	100.00
ADR* per Service			100.00

*Average detection rate.

(a) SNORT

	Direct Attacks		
	Detected	Total	%
Apache Tomcat	56 +5	61	100.00
DistCC	0	12	0.00
MSSQL	31	31	100.00
PostgreSQL	0	13	0.00
Samba	0	19	0.00
Server	26	26	100.00
Overall Detection	118	162	72.84
ADR* per Service			50.00

*Average detection rate.

(b) SURICATA

Table 13. Detection of obfuscated attacks in ASNM-NPBO dataset by SNORT and SURICATA.

	Obfuscated Attacks			
	Detected	Total	Evaded	%
Apache Tomcat	128 +36	164	0	100.00
DistCC	23	23	0	100.00
MSSQL	103	103	0	100.00
PostgreSQL	45	45	0	100.00
Samba	44	44	0	100.00
Server	98	100	2	98.00
Overall Detection	478	480		99.58
ADR* per Service				99.67

*Average detection rate.

(a) SNORT

	Obfuscated Attacks			
	Detected	Total	Evaded	%
Apache Tomcat	162 +1	163	0	100.00
DistCC	0	23	0	0.00
MSSQL	103	103	0	100.00
PostgreSQL	0	45	0	0.00
Samba	0	44	0	0.00
Server	98	100	2	98.00
Overall Detection	364	478		76.15
ADR* per Service				49.67

*Average detection rate.

(b) SURICATA

2) LEGITIMATE NETWORK TRAFFIC

The legitimate samples of this dataset were collected from two sources:

- A common usage of all previously mentioned services was obtained in an anonymized form, excluding the payload, from a real campus network in accordance with policies in force. Analyzing packet headers, we observed that a lot of expected legitimate traffic contained malicious activity, as many students did not care about up-to-date software. Therefore, we filtered out network connections yielding high and medium severity alerts by signature-based NIDS – Suricata and SNORT – through Virus Total API [37].

- The second source represented legitimate traffic simulation in our virtual network architecture and also employed all of our non-payload-based obfuscations for the purpose of partially addressing overstimulation in adversarial attacks against IDS [49], and thus making the classification task more challenging. However, only 109 TCP connections were obtained from this stage – this was caused by the fact that services such as Server and DistCC were difficult to emulate.[13] Simulation of legitimate traffic was aimed at various *SELECT* and *INSERT* statements when interacting with the database services (i.e., PostgreSQL, MSSQL); several *GET* and *POST* queries to our custom pages as well as downloading of high volume data when interacting with our HTTP server (i.e., Apache Tomcat); and several queries for downloading and uploading small files into Samba share.

The class distribution of the final dataset after extraction of ASNM features is summarized in Table 11

3) LABELING

ASNM-NPBO dataset contains four types of labels that are enumerated by increasing order of their granularity in the following:

- **label_2:** is a two-class label, which indicates whether an actual sample represents a network buffer overflow attack or a legitimate communication.
- **label_3:** is a three-class label, which distinguishes among legitimate traffic (symbol 3), direct attacks (symbols 1), and obfuscated network attacks (symbol 2).
- **label_poly:** is a label that is composed of 2 parts: (a) a three-class label, and (b) acronym of a network service. This label represents a type of communication on a particular network service.
- **label_poly_o** is the last label, which is composed of 3 parts: (a) three-class label, (b) employed obfuscation technique, and (c) acronym of network service. The label has almost the same interpretation as label_poly but moreover introduces obfuscation technique employed (identified by ID from Table 10) into all obfuscated attack samples.

[13]Note that additionally to those 109 TCP connections that were explicitly simulated, other 2252 TCP connections from obfuscated dictionary attacks were also considered as legitimate, and thus also helped in achieving a resistance against the overstimulation attacks.

4) TESTING WITH SIGNATURE-BASED NIDS

To investigate the effect of the proposed non-payload-based obfuscations on signature-based NIDSs, we performed detection by SNORT and SURICATA in a similar manner as we did in the case of the tunneling obfuscations (see Section IV-B), while the same ruleset was employed.

First, we let NIDSs inspect direct attacks that exploit the current network vulnerabilities. The results of the inspection summarize the detection properties of SNORT and SURICATA, and are depicted in Table 12. We can see in the tables that SNORT overcame SURICATA and correctly detected 100.00% of direct attacks. However, only 33 direct attacks on Apache service were detected by high priority rules of SNORT, and 24 attacks were undetected. Despite it, we considered these attacks as correctly detected, as they occurred almost at the same time as other correctly predicted direct attacks, and thus might be a part of their execution. In the case of SURICATA, the only one such undetected direct attack occurred. Nevertheless, unlike SNORT, SURICATA did not fire any alert representing buffer overflow, shellcode, or remote command execution, but instead fired combination of high priority alerts related to potential corporate privacy violation:

- `ET POLICY Incoming Basic Auth Base64 HTTP Password detected unencrypted`
- `ET POLICY Outgoing Basic Auth Base64 HTTP Password detected unencrypted`
- `ET POLICY HTTP Request on Unusual Port Possibly Hostile`
- `ET POLICY Internet Explorer~6~in use Significant Security Risk,`

which we decided to consider as correctly detected. If we were not to consider them as correctly detected, then SURICATA would not detect any attack on the Apache service.

Next, we analyzed the detection capabilities of both NIDSs on obfuscated attacks and the results are depicted in Table 13. Comparing the detection rate of SNORT and SURICATA on obfuscated attacks, we can conclude that SNORT overcame SURICATA again and the ratio of their correct detection was almost the same as in the case of direct attacks (see Table 12). The only difference occurred during the exploitation of a vulnerability in Server service, where two instances of obfuscated attacks were not detected by any NIDS. These two instances utilized obfuscations with IDs (f) and (g), both from a category of unreliable network traffic channel simulation techniques (see Table 10). There were also several undetected obfuscated attacks on Apache service in both NIDSs, but we were able to track their occurrences due to the label in the dataset, associating them as a part of other correctly detected attacks; hence, the detection rate for Apache service achieved 100.00% for both NIDSs. Regarding Apache service, SURICATA once again did not fire any alert detecting malicious content, however, it fired the previously mentioned combination of high priority alerts stating corporate privacy violation, which we, once again, considered as a correct detection. Also, note that SURICATA fired one non-high-priority alert classified as potentially bad traffic in all instances of direct and obfuscated attacks exploiting PostgreSQL service:

- `ET POLICY Suspicious inbound to PostgreSQL port 5432.`

However, we did not consider it as a correct detection due to the low priority of the alert. As discussed in Section IV-B, VirusTotal likely uses a paranoid rule set, and thus it fired alerts that might contain false positives. Comparing fired alerts before and after obfuscation, we can see that utilized NIDSs detected most of the attacks obfuscated by non-payload-based obfuscations, although the evasion of these NIDSs was not as successful as in the case of tunneling obfuscations. On the other hand, there were a few cases of the successful evasion of these NIDSs: 2 instances of the Server service for SNORT (see Table 13a) and 2 instances of the same service for SURICATA (see Table Table 13b).

V. STATISTICAL ANALYSIS

In this section, we discuss the results of the statistical analysis that we performed on all introduced datasets. In detail, our datasets are analyzed using major statistical methods: (1) skewness for measuring data asymmetry [50], (2) kurtosis for measuring the height and sharpness of central peak in the probability distribution of a feature [50], and (3) Kolmogorov-Smirnov (K-S) test [51] is used to compare the feature distribution to the normal distribution.

Before applying these statistical methods, all values are standardized using Z-score transformation to ensure that all features are in a standard format and meet a confidence interval. In the literature, it is common to see the split of the dataset into testing and training subsets when doing statistical analysis (e.g., [17], [52]–[54]). However, since we do not explicitly specify the training and testing sets in the ASNM datasets, the split of the datasets in our statistical analysis is based on the attribute *label_poly*, which distinguishes between legitimate and malicious traffic, while in the case of datasets with obfuscated traffic it furthermore takes obfuscation into account as a splitting criterion. Hence, we split ASNM-CDX-2009 dataset for the purpose of statistical analysis into two subsets: legitimate traffic, malicious traffic, while we split ASNM-NBPOv2 and ASNM-TUN datasets into three subsets: legitimate traffic, malicious traffic, malicious obfuscated traffic. We performed the statistical analysis by the SPSS tool [55], where the tool provided output only for features with at least one non-zero value.

A. ASNM-CDX-2009 DATASET

Figure 6 shows statistics for ASNM-CDX-2009 dataset. The results demonstrate that skewness and kurtosis of the features extracted from legitimate traffic is more significant than in the case of malicious traffic. This phenomenon especially

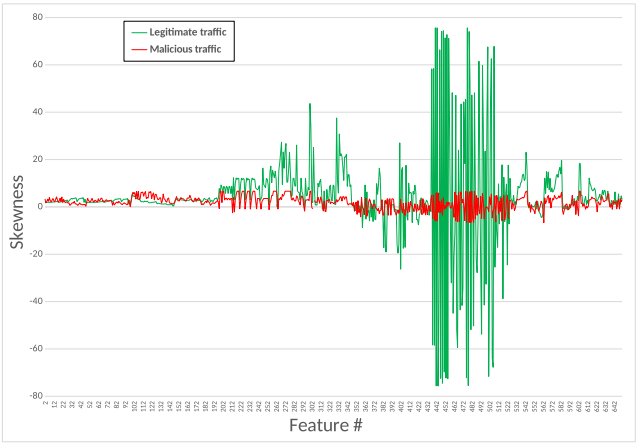
occurs in features based on the polynomial approximations in time domain (features 430-520) since normal traffic has wider spread of session in time while malicious traffic represents just short-lived sessions leading to particular exploitation of the service. Note that the positive and negative values in this range of features are caused by the alteration of the incoming/outgoing packets in the computation of these features. Kurtosis also indicates that the outliers are more common in the normal traffic than malicious one, which occurs due to the same reason as mentioned above. We can observe from the K-S test that similarity with the normal distribution ranges from 0.1 to 0.5, which means that both classes have low similarity with the normal distribution; additionally, we observe that the legitimate traffic is slightly more similar to the normal distribution than malicious traffic.

B. ASNM-TUN DATASET

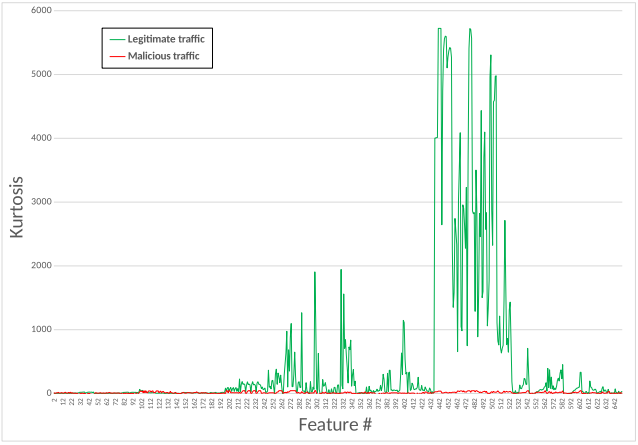
Figure 7 shows statistics for ASNM-TUN dataset, where in addition to the malicious and legitimate traffic, we depict obfuscated malicious traffic. In this case, the skewness and kurtosis of malicious and legitimate traffic is mostly in the similar range. This can be explained by the dataset composition: the dataset contains a relatively small amount of legitimate traffic in contrast to malicious traffic and in contrast to remaining two datasets. Hence, from the statistical point-of-view, the diversity of normal traffic is lower than in the remaining two datasets. Further, we observe that the obfuscated malicious traffic generally has lower skewness and kurtosis than the non-obfuscated malicious traffic. We attribute this observation to the fact that tunneling obfuscation introduces a constant size data that wrap the malicious traffic (into HTTP(S) tunnels), and thus cause a smaller variability of some features. This is manifested in the smaller number of outliers (i.e., kurtosis) and the shifted mean toward the tail of the feature distributions (i.e., skewness). From the K-S test, we see that all three classes of traffic have a low similarity with the normal distribution, where the most similar is legitimate traffic. Between two malicious traffic classes, the higher similarity with the normal distribution occurs in the obfuscated case due to a constant wrapping overhead that is a part of this obfuscation. Note that the goal of the obfuscation is to be more similar to the legitimate traffic, which is naturally the most similar to the normal distribution. Hence, the goal was at some extent met.

C. ASNM-NPBO DATASET

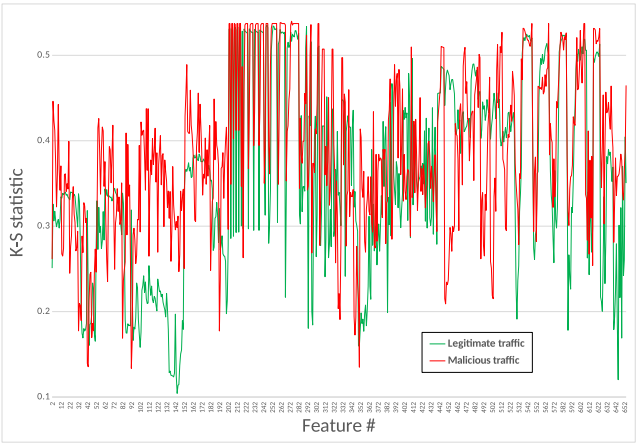
Figure 8 shows statistics for ASNM-NPBO dataset, depicting three traffic classes. Regarding skewness and kurtosis, the observations are similar to the ASNM-CDX-2009 dataset. When comparing malicious and obfuscated malicious traffic together, we see a different phenomenon in contrast to the ASNM-TUN dataset. In the current dataset, obfuscated malicious traffic often evinces higher skewness and kurtosis. This phenomenon is likely caused by the high diversity of obfuscations used in this dataset. In contrast, the ASNM-TUN dataset always utilizes a single type of obfuscation, causing



(a) Skewness



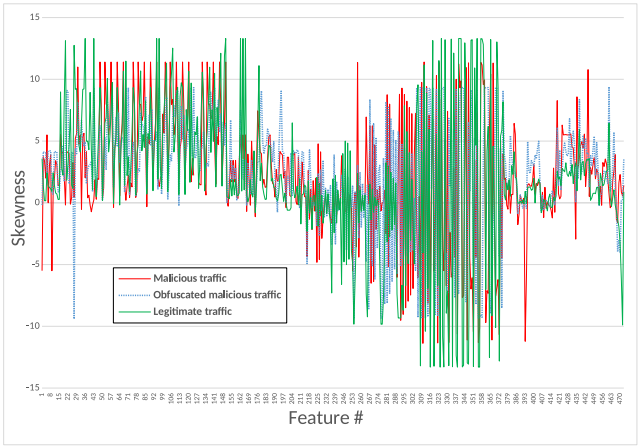
(b) Kurtosis



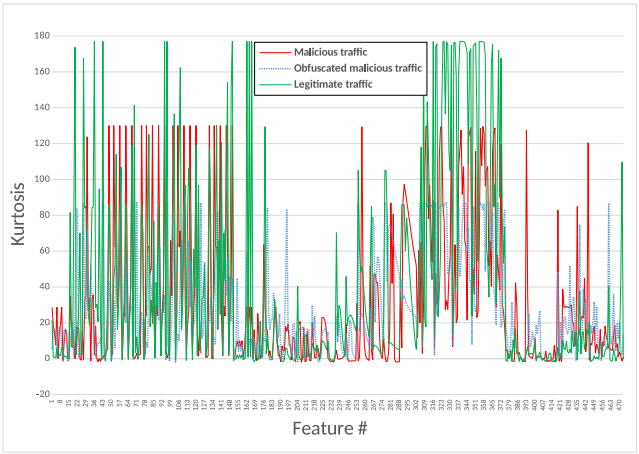
(c) K-S test with the normal distribution

Figure 6. Statistics of the ASNM-CDX-2009 dataset.

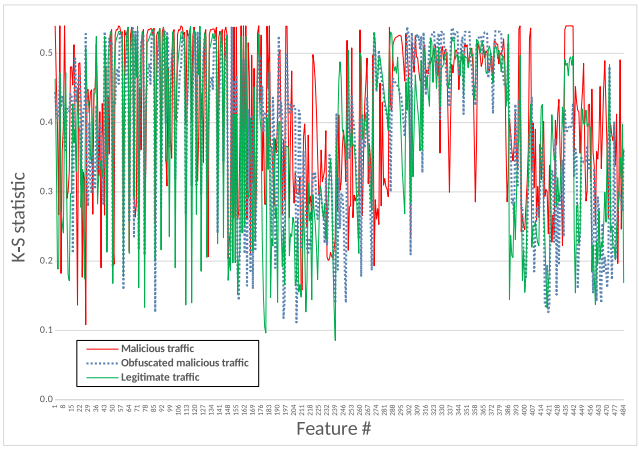
a lower diversity of obfuscated malicious traffic. From the K-S test, we see that all three classes of traffic have a low similarity with the normal distribution, where the most similar appears to be obfuscated malicious traffic. Note that the goal of obfuscated malicious traffic is to be more similar to the legitimate traffic and less similar to the malicious traffic, which was to some extent met in the case of this dataset.



(a) Skewness

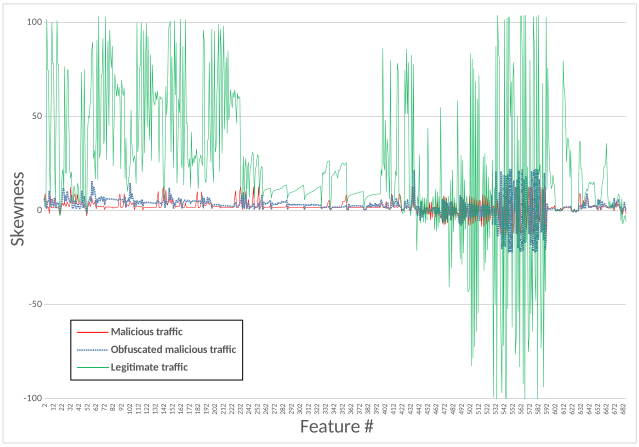


(b) Kurtosis

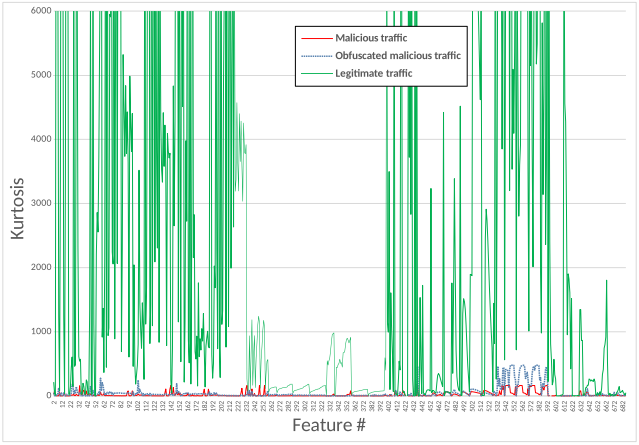


(c) K-S test with the normal distribution

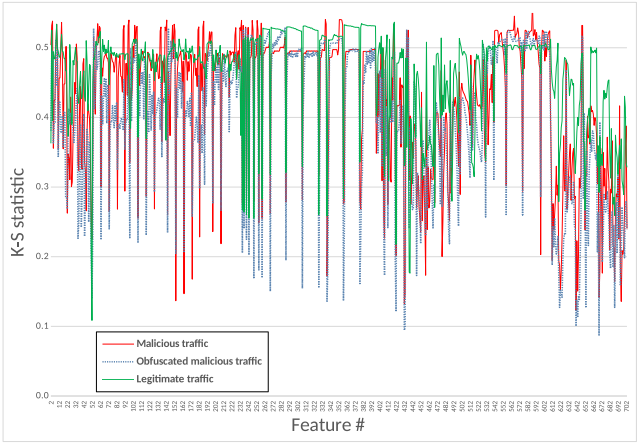
Figure 7. Statistics of the ASNM-TUN dataset.



(a) Skewness



(b) Kurtosis



(c) K-S test with the normal distribution

Figure 8. Statistics of the ASNM-NPBO dataset.

VI. BENCHMARKING THE DATASETS

We conducted several machine learning experiments with ASNM datasets using RapidMiner [56], and we summarize them in this section. We emphasize that the following machine learning experiments are not exhaustive since it is not the intention of this work. On the contrary, we provide the results only as a baseline, while our main intention is as follows: (1) we demonstrate that ASNM features have enough discrimination rate required for the classification of the malicious and legitimate traffic, (2) we show that machine learning models are prone to evasion of their detection by proposed obfuscation techniques, and finally (3) we show

that evaded classifiers can be improved by training data augmentation.

A. FEATURE SELECTION

The goal of feature selection is to reduce the dimensionality of data by removing irrelevant features [57], and thus to speed up or improve the classification. In general, the feature selection has two variants [58]: (1) the *filter model*, and (2) the *wrapper model*. The filter model removes features based on their intrinsic characteristics and does not require a classifier. The wrapper model requires a classifier, and thus it is more computationally expensive than the filter model; moreover, the selected features are biased toward the classifier used.

We consider the filter model as part of the preprocessing, where we remove all zero-value features and reduce features that correlate by more than 95%. The wrapper model might employ optimal or suboptimal approaches. An optimal approach is to do an exhaustive search, which guarantees to find the optimal solution, but its disadvantage is its computational cost, thus it is rarely used in the practice. There are a few methods that can obtain the optimal feature set without exhaustive search (e.g., by genetic algorithms); however, they are still computationally expensive. Therefore, suboptimal search methods are more common in practice. The well known suboptimal sequential methods for feature selection are: (1) *forward feature selection* (FFS), which starts with a single feature and in each generation adds the new feature that improves the target metric in the highest extent; (2) *backward elimination*, which starts with all features included and step-by-step eliminates a single feature that causes the lowest deterioration of the target metric. Both of the approaches have their pros/cons, and their suitable use cases. Since our intention is to provide only initial results demonstrating the discrimination capability of ASNM features as well as the effect of obfuscations, we opt for FFS, which is the fastest and cheapest approach.

The next parameter of the feature selection is the target metric, such as accuracy, F_1*-measure*, average recall, average precision, chi-square, information gain, correlation coefficient, etc. In our case, we opt for F_1*-measure* since it copes with the imbalances of class distribution in our datasets.

B. ASNM-CDX-2009 DATASET

1) FORWARD FEATURE SELECTION

First, we used 5-fold cross-validation and forward feature selection (FFS) on top of the Naive Bayes classifier with kernel functions for the estimation of density distribution, which represents a non-parametric estimation method.[14] In FFS, we accepted one iteration without improvement as we wanted to avoid the selection process from getting stuck in local extremes. The maximal number of selected features was limited to 20, and we used the binary label of the dataset (i.e., *label*_2) for this experiment. The resulting set of selected features by FFS is presented in Table 27 of Appendix.

[14]Note that in the Naive Bayess classifier from RapidMiner [56], we used greedy kernel density estimation method with minimum bandwidth 0.1 and maximum number of kernels equal to 10.

Table 14. A comparison of several classifiers on ASNM-CDX-2009 dataset. The results were obtained with 100 times repeated 5-fold cross-validation.

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Naive Bayes	83.00	1.60	0.03	0.01	88.48	1.39	91.49	0.80
SVM	76.94	3.60	0.06	0.01	82.80	2.73	88.44	1.79
Decision Tree	75.40	2.32	0.11	0.03	79.33	2.48	87.65	1.16

2) COMPARISON OF SEVERAL CLASSIFIERS

Next, we compared three non-parametric classifiers from RapidMiner by 5-fold cross-validation while using a subset of ASNM features obtained by FFS in the above experiment. We repeated the cross-validation 100 times to obtain statistically significant results, expressed by the mean and standard deviation. The results for individual classifiers are presented in Table 14. First, we utilized the Naive Bayes classifier with kernel density estimation, minimum bandwidth equal to 0.1, and maximum number of kernels equal to 10. Next, we evaluated the performance of the SVM classifier (based on libsvm), which utilized a radial basis function as the non-linear kernel, the cost parameter equal to 10.0, and γ parameter equal to 1.0. In the case of the decision tree classifier, we utilized the Gini index as a selection criterion for the splitting of attributes. The decision tree had pruning enabled and we constrained the maximal depth of the tree to 8 levels since deep trees do not generalize data well, hence suffer from over-fitting.

C. ASNM-TUN DATASET

1) FORWARD FEATURE SELECTION

Similar to the case of the previous dataset, we started again with the FFS method using the same Naive Bayes classifier and 5-fold cross-validation, while we allowed acceptance of one FFS iteration without improvement to avoid the selection process becoming stuck in local extremes. All cross-validation experiments have been adjusted to employ stratified sampling during assembling of folds, which ensured equally balanced class distribution of each fold. We performed two-class prediction, and thus we used the label denoted as *label*_2.

The experiment consisted of two executions of FFS. In the first execution, we took legitimate traffic and direct attack entries as an input, which represented the case where a classifier was trained without knowledge about obfuscated attacks. In the second execution, we took the whole dataset as the input – consisting of legitimate traffic, direct attacks, obfuscated attacks – therefore, representing the case in which the classifier was aware of obfuscated attacks. The selected features of both executions are depicted in Table 26 of Appendix. The penultimate column of the table (i.e., FFS DOL) denotes

Table 15. A comparison of several classifiers on the subset of ASNM-TUN dataset containing only direct attacks and legitimate traffic. The results were obtained with 100 times repeated 5-fold cross-validation.

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Naive Bayes	92.26	1.21	1.19	0.48	95.13	0.74	95.53	0.64
Decision Tree	95.45	0.97	6.18	1.31	93.68	1.01	94.63	0.87
SVM	93.62	1.54	4.98	0.79	93.41	1.04	94.32	0.89

the selected features where the whole dataset was utilized for the FFS, and the last column (i.e., FFS DL) denotes the case where only direct attacks and legitimate traffic were taken into account.

Since some features were inconvenient for comparison of synthetic attacks with legitimate traffic captured in a real network, these features were removed from the dataset in the pre-processing phase of our experiment. The examples include TTL-based features, IP addresses, ports, MAC addresses, the occurrence of source/destination host in the monitored local network, some context-based features, etc.

We note that several mutual features were selected in both cases, which means they provided a value regardless of whether obfuscation was performed or not. Almost all of the following experiments will use the feature set obtained from the second execution (i.e., FFS DOL features), as we consider them as more appropriate for general behavior representation of both kinds of attacks.

2) NORMAL OPERATION

To evaluate the performance of classifiers under normal circumstances (not considering obfuscated attacks), we repeated 5-fold cross-validation 100 times with FFS DL features using all direct attack samples and legitimate traffic samples (see Table 15). All three classifiers achieved a high performance under the known conditions. For all classifiers, we used the same settings as for other datasets (see Section VI-B and Section VI-D).

3) EVASIONS

To evaluate the performance of classifiers under challenging conditions that assume obfuscations in place, we executed an experiment that performed detection of malicious obfuscated attacks by the classifier trained on all direct attacks and legitimate traffic samples (see Table 16). It represented the situation when the classifier had no previous knowledge about obfuscated attacks, and therefore we used FFS DL feature set. In the result, most of the obfuscated attacks were misclassified as legitimate traffic, and thus TPR was exacerbated by more than 30% in the case of all classifiers used.

4) TRAINING DATA AUGMENTATION

Our second binary classification experiment considered explicit information about obfuscated attacks in the training phase of the classifier. Therefore, we used direct and obfuscated attacks labeled as one class while repeating 5-fold cross-validation 100 times to obtain statistically significant results. Since the classifier ''was aware'' about obfuscations, we used FFS DOL feature set for the purpose of this experiment (see Table 17), but for completeness, we executed this experiment with FFS DL feature set as well (see Table 19). In the case of FFS DOL features, the outcome of this experiment indicates an improved TPR and a high F_1*-measure* in all classifiers. In the case of FFS DL features, we observe that performance improvement was not as significant as in the previous case, in particular, all classifiers still suffered from high FPR. This indicates the importance of not only training data augmentation by obfuscated attacks but also updating the feature set to reflect obfuscations.

Table 16. Prediction of obfuscated attacks and all attacks in the ASNM-TUN dataset by classifiers trained without knowledge about obfuscated attacks (i.e., trained on all direct attacks and legitimate traffic samples).

Classifier	TPR (↑)	Δ TPR
Decision Tree	11.49%	-83.96%
SVM	9.20%	-84.42%
Naive Bayes	3.45%	-88.81%

(a) Obfuscated attacks

Classifier	TPR (↑)	Δ TPR
Decision Tree	64.52%	-30.93%
SVM	62.21%	-31.41%
Naive Bayes	60.83%	-31.43%

(b) All attacks

5) COMPARISON OF SEVERAL CLASSIFIERS

We observe that all classifiers suffer from high sensitivity to evasions caused by tunneling obfuscations (see Table 16), while Naive Bayes classifier was the most sensitive one.

After augmentation of a training data without updating the feature set (see Table 19), we observe that the decision tree classifier is the most robust, while SVM achieves the worst performance. These observations held even when we made a training data augmentation with updating the feature set (see Table 17). When updating the feature set, we can also see that F_1 for all classifiers achieved a higher rate than in the case of the baseline experiment with direct attacks and legitimate traffic (see Table 15). Moreover, the only classifier that did suffer from deteriorated FPR after training data augmentation was Naive Bayes.

For more experiments with this dataset, including tri-nominal and multi-nominal labels and individual feature analysis, we refer the reader to [13], [59], and [14].

D. ASNM-NPBO DATASET

1) FORWARD FEATURE SELECTION

Similar to the case of the previous datasets, we again started with the FFS method using the Naive Bayes classifier and 5-fold cross-validation, while we allowed acceptance of one FFS iteration without improvement. We performed two-class

Table 17. Performance measures from 100 times repeated 5-fold cross-validation of the whole ASNM-TUN dataset using FFS DOL feature set, representing the situation where classifiers were aware of some obfuscated attacks, and therefore caused performance improvement in contrast to the classifiers that were aware only about direct attacks (see Table 16).

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Decision Tree	96.51	0.91	4.46	1.20	96.43	0.66	96.02	0.75
Naive Bayes	96.29	0.83	6.48	0.82	95.57	0.57	94.91	0.62
SVM	93.07	1.02	5.74	0.74	94.13	0.65	93.69	0.66

(a) Performace measures (FFS DOL features)

Classifier	Δ TPR (↑)	Δ FPR
Naive Bayes	+35.46%	+5.29%
Decision Tree	+31.99%	-1.72%
SVM	+30.86%	+0.76%

(b) Improvement (FFS DOL features)

prediction (i.e., using *label_2*) in two executions of FFS using the Naive Bayes classifier with the same parameters as in our previous experiments – the first execution did not contain obfuscated attack samples (i.e., resulting into FFS DL features) and the second execution included these samples (i.e., resulting into FFS DOL features). The selected features of both executions are depicted in Table 28 of Appendix. Similar to the case of the ASNM-TUN dataset, we removed the features that are inconvenient for the comparison of synthetic attacks with legitimate traffic captured from the real world.

2) NORMAL OPERATION

To evaluate the performance of classifiers under normal circumstances (not considering obfuscated attacks), we repeated 5-fold cross-validation 100 times with FFS DL features using all direct attack samples and legitimate traffic samples. The performance measures of three classifiers validated by the cross-validation are shown in Table 18. All three classifiers achieved a high performance under the known conditions. For all classifiers, we used the same settings as in the previous experiments (see Section VI-B and Section VI-C).

3) EVASIONS

Next, we took the classifiers trained on all data samples of direct attacks and legitimate traffic, and we applied them for the prediction of the obfuscated attacks and all attacks,

Table 18. Results from 100 times repeated 5-fold cross-validation on a subset of ASNM-NPBO dataset that contains only direct attacks and legitimate traffic samples.

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Decision Tree	93.85	1.70	0.10	0.03	93.59	1.33	96.87	0.85
SVM	88.12	1.73	0.01	0.01	93.21	1.05	94.05	0.87
Naive Bayes	95.71	0.56	0.61	0.03	92.94	0.34	97.55	0.28

Table 19. Performance measures from 100 times repeated 5-fold cross-validation of the whole ASNM-TUN dataset using FFS DL feature set, representing the situation where classifiers were aware of some obfuscated attacks, and therefore caused performance improvement in contrast to the classifiers that were aware only about direct attacks (see Table 16). However, since the feature set was not updated to reflect obfuscated attacks, the performance upgrade is not significant.

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Decision Tree	87.03	1.92	18.12	2.49	86.24	1.30	84.45	1.46
Naive Bayes	90.97	2.04	39.86	6.10	81.93	1.22	75.54	2.38
SVM	78.25	1.59	20.84	1.44	80.09	1.12	78.71	1.08

(a) Performace measures (FFS DL features)

Classifier	Δ TPR (↑)	Δ FPR
Naive Bayes	+30.14%	+38.67%
Decision Tree	+22.51%	+11.94%
SVM	+16.04%	+15.86%

(b) Improvement (FFS DL features)

respectively (see Table 20).[15] We observe that TPRs were deteriorated for all classifiers, which means that some obfuscated attacks were successful – they were predicted as legitimate traffic, and thus caused evasion of the classifiers.

4) TRAINING DATA AUGMENTATION

To improve the resistance of the classifiers against evasions, we widened their knowledge about different mixtures of obfuscated attack instances, which was accomplished by running 5-fold cross-validation of the whole dataset. The cross-validation was repeated 100 times to obtain the statistically significant results. In this experiment, we used FFS DOL features, which consider knowledge about obfuscated attacks, and thus updating not only the model of the classifier but also the underlying feature set (in contrast to the previous experiment). Additionally, we show the results with FFS DL

Table 20. Prediction of obfuscated attacks and all attacks in the ASNM-NPBO dataset by classifiers trained without knowledge about obfuscated attacks (i.e., trained on all direct attacks and legitimate traffic samples).

Classifier	TPR (↑)	Δ TPR
Naive Bayes	52.30%	-45.85%
Decision Tree	36.61%	-59.07%
SVM	15.90%	-66.82%

(a) Obfuscated attacks

Classifier	TPR (↑)	Δ TPR
Naive Bayes	64.38%	-33.77%
Decision Tree	52.03%	-43.65%
SVM	26.25%	-56.47%

(b) All attacks

[15] Note that we do not depict FPRs in the tables since no changes to the composition of legitimate traffic samples were made, hence FPRs remain the same as in Table 18.

Table 21. Performance measures from 100 times repeated 5-fold cross-validation of the whole ASNM-NPBO dataset using FFS DOL feature set, representing the situation where the classifiers were aware of some obfuscated attacks, and therefore caused performance improvement in contrast to the classifiers that were aware only about direct attacks (see Table 20).

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Naive Bayes	98.01	0.37	0.19	0.01	97.39	0.23	98.91	0.19
Decision Tree	96.97	0.57	0.20	0.04	96.83	0.46	98.38	0.29
SVM	88.66	0.79	0.03	0.01	93.73	0.46	94.32	0.39

(a) Performace measures (FFS DOL features)

Classifier	Δ TPR (↑)	Δ FPR
SVM	+62.41%	+0.02%
Decision Tree	+44.94%	+0.10%
Naive Bayes	+33.63%	-0.42%

(b) Improvement (FFS DOL features)

Table 22. Performance measures from 100 times repeated 5-fold cross-validation of the whole ASNM-NPBO dataset using FFS DL feature set, representing the situation where classifiers were aware of some obfuscated attacks, and therefore caused performance improvement in contrast to the classifiers that were aware only about direct attacks (see Table 20).

Classifier	TPR [%]		FPR [%]		F_1 [%] (↑)		Average Recall [%]	
	mean	std	mean	std	mean	std	mean	std
Decision Tree	95.49	0.61	0.26	0.05	95.53	0.53	97.61	0.31
SVM	89.33	0.58	0.10	0.02	93.54	0.36	94.62	0.29
Naive Bayes	95.71	0.56	0.61	0.03	92.94	0.34	97.55	0.28

(a) Performace measures (FFS DL features)

Classifier	Δ TPR (↑)	Δ FPR
SVM	+63.08%	+0.09%
Decision Tree	+43.46%	+0.16%
Naive Bayes	+31.33%	-0.00%

(b) Improvement (FFS DL features)

features that consider updating the model only. Particular results of this experiment are shown in Table 21 for FFS DOL features and Table 22 for FFS DL features. Comparing against the results from the previous experiment (see FPRs from Table 18 and TPRs from from Table 20b), most of the classifiers were significantly improved in TPR, while FPR deteriorated slightly. Hence, the classifiers trained with knowledge about some obfuscated attacks were able to detect the same and similar obfuscated attacks later.

5) COMPARISON OF SEVERAL CLASSIFIERS

From the previous experiments, we can say that the Naive Bayes classifier was the least sensitive to evasions by non-payload-based obfuscations (see Table 20), while SVM was the most sensitive classifier.[16] This might be caused by over-fitting of the training data. Moreover, all classifiers used the feature sets selected by FFS with the Naive Bayes classifier, which might be seen as a disadvantage for classifiers other than Naive Bayes. Nevertheless, we reran FFS with individual classifiers as well but obtained results much worse than in the case of using the features selected by the Naive Bayes classifier.

After augmentation of a training data without updating the feature set (see Table 22), we observe that the decision tree classifier is the most robust, while Naive Bayes achieves the worst performance. However, when making a training data augmentation with updating the feature set (see Table 21), Naive Bayes performed better than other classifiers, which might be caused again by over-fitting of training data.

For more experiments with this dataset, including tri-nominal and multi-nominal labels, detection of unknown obfuscations by a custom leave-one-out validation, and individual feature analysis, we refer the reader to [23] and [14].

[16]The bigger the decrease in TPR, the more sensitive a classifier is to obfuscations.

VII. DISCUSSION

A. AGE OF VULNERABILITIES

Although there is a plethora of publicly available exploit-codes for contemporary vulnerabilities, the situation with corresponding available vulnerable software is more difficult due to understandable prevention reasons imposed by software vendors. Therefore, we were able to contain only older available high-severity vulnerable services that are outdated. However, we conjecture that from the point of view of non-payload-based network intrusion detection (not inspecting the payload of packets), the behavioral characteristics of simulated high-severity attacks are similar regardless of the age of vulnerabilities. In particular, we refer to the buffer overflow attacks, which are executed in a few stages involving a repeated transfer of one or more packets with the maximum payload. In our future work (and our web page), we will release the ASNM-NPBO-II dataset, where we will focus on the newer vulnerabilities.

B. COMPARISON WITH THE REAL-WORLD TRAFFIC

ASNM datasets contain a high ratio of malicious to legitimate connections,[17] while in practice this ratio is usually several orders of magnitude lesser. Although an arbitrary value of this ratio does not distort the performance of the classifier when an appropriate performance measure is chosen (e.g., $F_1 - measure$), it might impact the accuracy of modeling the legitimate class – a high volume of legitimate traffic occurred in practice can result in a high divergence of data, which might not be captured by models built from our datasets in a sufficient manner. Therefore in practice, classifiers would require much more legitimate data than it is contained in our datasets.

Next, we note that all attacks that we executed using Metasploit framework in the laboratory setting might be in practice influenced by actual network conditions, which

[17]This ratio is equal to 0.76%, 55.07%, and 5.59% in the case of ASNM-CDX, ASNM-TUN, and ASNM-NPBO, respectively.

might cause higher delays or retransmissions than in our direct attack class. However, these network conditions are explicitly emulated by network scenarios in the ASNM-TUN dataset, while in the ASNM-NPBO dataset they are covered by non-payload-based obfuscations that go even beyond what is commonly observed in the real-world network traffic.

C. CROSS-DATASET EVALUATION

In this paper, we provided only a basic benchmarking of several supervised classifiers on ASNM datasets. Nevertheless, it is worth noting that different benchmarking techniques can be used as well. One example is cross-dataset evaluation, where the target classifier is trained on the input data of one dataset, and then it is evaluated on the data taken from another dataset. We leave these tasks as an open challenge for the community.

D. CHALLENGE FOR THE COMMUNITY

We highlight that benchmarking of the ASNM datasets (see Section VI) serves only for the demonstration purposes and its role is to show discriminative aspects of ASNM features in the domain of network intrusion detection as well as to show the impact of the proposed obfuscations. In this work, we used a very basic setup for the machine learning pipeline, and we believe that the results obtained can be improved in future work and by the community. For example, we utilized a basic suboptimal feature selection method (i.e., FFS) due to its speed. However, the literature contains plenty of other relatively cheap feature selection methods that enable to over-come a drawback of FFS, e.g., [60]–[64], which might lead to better results than we obtained. FFS might also be improved by a different target metric; for example, Thaseen and Kumar [65], Thaseen *et al.* [66] achieved the best results with chi-square metric in the multi-class classification of NSL-KDD dataset.

Finally, the classifiers used in this work were selected due to their speed,[18] and we believe that many other classifiers, especially ensembles, might improve on the results that we presented here. For the objective comparison with our baseline, we recommend the community to use 5-fold cross validation and report on statistically significant results obtained by repeated execution of cross-validation (i.e., in our case repeated 100 times).

VIII. RELATED WORK

We split datasets that can be used for evaluation of NIDSs into two categories: (1) datasets containing *raw network traffic traces*, and (2) datasets containing *high-level features*. We summarize the properties of these datasets in Table 23, and we compare them with ASNM datasets.

[18]Time complexities of training the decision tree and the Naive Bayes are $O(nd^2)$ and $O(nd)$, respectively; where n represents the number of samples and d represents the number of features. In the case of SVM, the time complexity of training depends on both the number support vectors s and the number of training data samples n, and it can be constrained by $O(s^3 + n^2)$ [67].

A. DATASETS OF NETWORK TRAFFIC TRACES

Datasets from this category have one property in common: they contain network traffic traces with optional data serving for labeling purposes. The first four representatives of this category are large collections of datasets and are referred to as projects – MWS [68], PREDICT [69], CAIDA [70], NETRESEC [71]. The next four examples represent just one specific collection and are referred to as datasets – DARPA [72], CCRC [73], CDX [12], CONTIAGO [74].

1) PROJECT MWS

The project MWS represents a collection of various types of datasets that are primarily intended for use in anti-malware research [68], but some of them are also applicable in network intrusion detection. A summary of the MWS datasets is available in Japanese [75]–[79], and it covers three categories of datasets, which is based on phases of attacks: (1) probing, (2) infection, and (3) malware activities after infection. From the perspective of network intrusion detection, we consider PRACTICE, D3M, CCC, and NICTER as related datasets of MWS. However, for network intrusion detection it is also important to have a ground truth, which can be inferred from the MWS datasets called FFRI, IIJ MITF, D3M, and CCC. These datasets were collected using various honeypots that were exposed to the Internet and some of to darknet.

2) PROJECT PREDICT

The project PREDICT [69] provides 430 datasets in 14 categories contributed by several data providers. From all 14 categories, just three of them are relevant to the network intrusion detection and could be used for evaluation purposes: (1) blackhole address space, (2) IP packet headers containing DDoS, port scans, worms, and (3) synthetically generated data. Also, note that the IDS and firewall category contains a large collection of logs that do not contain any PCAP files, therefore it could not be used for IDS evaluation. categories.

3) PROJECT CAIDA

Center for Applied Internet Data Analysis (CAIDA) [70] collects several different network data types at geographically diverse locations. From the network intrusion detection perspective, CAIDA includes datasets containing e.g. DDoS attacks [80], [81], botnet traffic [82], dumps of various worms (Conficker [83], Code-Red [84], Witty [85]). However, these datasets lack labeling, and therefore their utilization for NIDS would require additional labeling effort.

4) PROJECT NETRESEC

NETRESEC [71] maintains a comprehensive list of PCAP files that can be used for the evaluation of network intrusion detection approaches. The datasets are divided into several categories involving cyber defense exercises/competitions, forensics challenges, data captures from honeypots, sandboxes, and attacks at industrial control systems (ICS).

Table 23. A comparison of related work and ASNM datasets.

Dataset	Type of Attacks / Focus	Type of Data		Adversarial Techniques Used	Type of Adversarial Techniques
		Network Traces	Features		
Project MWS []	• Probing, intrusions caused by malware, intrusions collected by honeypots and sandboxes, darknet traffic	Yes	-	-	-
Project PREDICT	• Blackhole address space data, botnet traffic, DoS, port scans, worm outbreaks	Yes	-	-	-
Project CAIDA	• DDoS attacks, botnet traffic, traces from various worms, port scans	Yes	-	-	-
Project NETRESEC	• Attacks from cyber defense exercises, forensic challenges, data captures from honeypots, sandboxes, and attacks at ICS	Yes	-	-	-
DARPA 98/99 Datasets	• DoS, intrusions, port scanning, local privilege escalation	Yes	-	-	-
CCRC 2006 Dataset	• Various intrusions for signature-based NIDSs	Yes	-	Yes	• L2 and L3 protocol-based obfuscations
CDX 2009 Dataset	• Intrusions from the cyber defense exercise, involving buffer overflow attacks	Yes	-	-	-
Twente 2009 Dataset	• Intrusions collected by honeypots	Yes	-	-	-
ISCX 2012 Dataset	• Multi-stage intrusions	Yes	-	-	-
Contagio 2015 Dataset	• Intrusions caused by malware	Yes	-	-	-
KDD Cup '99 Dataset	• DoS, intrusions, port scanning, local privilege escalation	-	Yes	-	-
NSL KDD '99 Dataset	• DoS, intrusions, port scanning, local privilege escalation	-	Yes	-	-
Moore's 2005 Datasets	• Focus is on the flow classification, while intrusions are not explicitly addressed	-	Yes	-	-
Kyoto 2006+ Dataset	• Various intrusions collected by honeypots	-	Yes	-	-
OptiFilter 2014 Dataset	• Various intrusions		Yes	-	-
CICIDS 2017 Dataset	• DoS, DDoS, XSS, SQL injection, Heartbleed, infiltration through the scam, and port scanning	-	Yes	-	-
ASNM-CDX-2009 Dataset	• Intrusions (buffer overflow) from the cyber defense exercise	-	Yes	-	-
ASNM-TUN Dataset	• High severity intrusions leading to remote shell via the buffer overflow	-	Yes	Yes	• Tunneling in HTTP and HTTPS protocols
ASNM-NPBO Dataset	• High severity intrusions leading to shell code (most attacks are from Metasploitable machines)	-	Yes	Yes	• L2, L3, and L4 protocol-based obfuscations

5) DARPA 1998 AND 1999 DATASETS

The Cyber Systems and Technology Group [72] of MIT Lincoln Laboratory has collected the first standard corpora for evaluation of network intrusion detection systems in 1998 and 1999. There were two datasets DARPA 1998 and 1999 collected, and later there was the third dataset released, marked as DARPA 2000, which addresses specific network scenarios. DARPA dataset contains network traffic generated by software that is not publicly available [86].

6) CCRC 2006 DATASET

Massicotte *et al.* [73] developed a framework for automatic evaluation of intrusion detection systems, and they collected a dataset consisting of several network attack simulations. Moreover, the framework contains a mutation layer that performs various L2 and L3 protocol based obfuscations using tools Fragroute [87] and Whisker [88], which is similar to obfuscations in the ASNM-NPBO dataset. The CRCC (Canada Communication Research Center) dataset is specific to signature-based network intrusion detection systems and contains only well-known attacks, without background traffic. The authors also report an initial evaluation of the framework on two NIDSs, namely SORT [89] and Bro [90].

7) CDX 2009 DATASET

The CDX 2009 dataset was introduced by Sangster *et al.* [12] and it contains data in tcpdump format as well as SNORT [27] intrusion prevention logs. We used this dataset in our research, and it is described in Section IV-A.

8) TWENTE 2009 DATASET

The Twente 2009 dataset [91] consists of 14.2M network flows (i.e., 155M packets) collected during a period of 9 days in 2008, where 7.6M of intrusion alerts were generated. The flows in this dataset were assembled by a modified version of *softflowd* utility, and 98% of them have been labeled by the authors. The authors collected the dataset by a honeypot installed on a virtual host with running Citrix XenServer 5. The deployed honeypot ran with three opened services: OpenSSH, Apache web server, and FTP server *proftp*.

9) ISCX 2012 DATASET

Shiravi *et al.* [92] presented guidelines for the generation of benchmark dataset consisting of creating a malicious and benign profile that were later executed during dataset generation. The authors generated their own dataset of network traffic (including the payload) for various network services such as HTTP, SMTP, SSH, IMAP, POP3, FTP. Several multi-stage attacks were also simulated during the collection and they cover all three categories from the MWS project.

10) CONTAGIO 2015 DATASET

The Contagio dataset [74] contains a collection of almost 1000 PCAP files from malware analysis, and thus it contains implicit expert knowledge about the occurrence of attacks/malware.

B. DATASETS CONSISTING OF HIGH-LEVEL FEATURES

This section contains representatives that were built from network traffic traces, hence it can be interpreted as a post-processed version of the former category. Examples are: KDD Cup '99 [17], NSL KDD '99 [93], Moore's 2005 [15], Kyoto 2006+ [16], and OptiFilter 2014 [94].

1) KDD CUP '99

The KDD Cup '99 [17] dataset was created from network traces of the DARPA 1998 dataset. The authors explicitly split the dataset into training and testing subsets, where The training subset contains 24 attack types and the testing subset contains an additional 14 attack types. Each connection sample in the dataset contains 41 features, and simulated attacks fall into four main categories [17], [93]: (1) DoS attacks, (2) external attacks exploiting network vulnerabilities (a.k.a., remote to local attacks / R2L), (3) insider threat attacks [95] in which a user with an ordinary account escalates her privileges to get superuser access (a.k.a., the user to root attack / U2R), and (4) reconnaissance attacks such as port scans.

The features of the KDD '99 dataset are divided into three categories [17]: (1) *basic features*, which are extracted from TCP and UDP communications, (2) *content features*, which inspect application-level data, and (3) *traffic features*, which calculate statistics related to protocol behavior, service, etc., and they are computed using a short time window (i.e., 2s).

Stolfo *et al.* [96] criticize time-based features since there exist several slow probing attacks that scan a host using a longer time window than 2s. Since content features parse the payload of packets regardless of whether it is encrypted or not, they can be extracted only at a host, as opposed to basic and traffic features that can be extracted from network traffic.

2) NSL KDD '99

Deficiencies of the KDD Cup '99 dataset were discussed in [93]. The main deficiency of the original dataset relates to redundancy – 78% entries of the training set and 75% entries in the testing set are replicated. The original dataset was modified, reduced, and released as the NSL KDD '99 dataset [52]. The training dataset contains about 130k entries and the testing one about 23k. All samples are sorted into the original 24 classes as well as into two classes representing a benign or a malicious entry.

3) MOORE'S 2005

The Moore's 2005 datasets [15] are primarily intended to aid in the assessment of network traffic classification. A number of datasets are described; each dataset consists of a number of entries representing TCP connection flows that are described by a feature set referred to as discriminators [15]. Input data for feature extraction were obtained by the Network Monitor tool [97]. In contrast to previously described KDD datasets, Moore's datasets were built from network traffic traces, and there were no data utilized from host machines during the extraction of the features.

4) KYOTO 2006+

Song *et al.* [16] presented the Kyoto 2006+ dataset, which was built from the 3 years of real-world network traffic collected by various types of honeypots. To label the dataset, the authors used a few security tools: SNS7160 IDS system, ClamAV software, Ashula, and SNORT. The dataset contains over 50M of normal sessions and over 43M of attack sessions. The Kyoto 2006+ dataset [98] consists of 14 statistical features taken from the KDD Cup '99 dataset and 10 additional features containing an indication of alerts from the security tools used, IP addresses, ports, and duration of sessions. The authors have not used any content-based features and focused only on network traffic data.

5) OptiFilter 2014/SecMonet

Salem *et al.* proposed OptiFilter [94], a framework that constructs connection vectors from network flows. OptiFilter handles ARP, ICMP, IP/TCP, and IP/UDP protocols. Moreover, it utilizes a finite state machine on TCP and UDP connections for monitoring of their state.

The extracted features of OptiFilter are influenced by KDD Cup '99 and Kyoto 2006+ datasets and cover statistical aspects of analyzed flows as well as host-based data extracted by SNMP. In the result, the authors generated a dataset called SecMonet, in which 17 common services

Table 24. Symbols of the TCP connection tuple.

Symbol	Description
$t_s \in \mathbb{R}^+$	Timestamp of the connection's start.
$t_e \in \mathbb{R}^+$	Timestamp of the connection's end.
$p_c \in \{0, \ldots, 2^{16}-1\}$	Port of the client within the TCP connection.
$p_s \in \{0, \ldots, 2^{16}-1\}$	Port of the server within the TCP connection.
$ip_c \in \{0, \ldots, 2^{32}-1\}$	IPv4 address of the client.
$ip_s \in \{0, \ldots, 2^{32}-1\}$	IPv4 address of the server.
P_c	Set of packets sent by client to server.
P_s	Set of packets sent by server to client.

were captured (e.g., FTP, SSH, telnet, SMTP, SMB, NFS, etc.). However, it is not clear whether the dataset contains a self-collected malicious traffic or it is only substituted from KDD Cup '99.

6) CICIDS 2017 DATASET

The CICIDS 2017 dataset [99] consists of network attacks such as DoS, DDoS, XSS, SQL injection, Heartbleed, infiltration through the scam, and port scanning. The authors generated benign data based on the extracted profile from an analysis of 25 users, which is in line with the approach proposed in [92]. The infrastructure used for the data collection consisted of 15 vulnerable Linux-based & Windows-based machines and 4 attacker machines. Further, the authors extracted 80 features using CICFlowMeter tool [19] and provided them along with the network traffic traces.

Table 25. Symbols of the packet tuple.

Symbol	Description
$t \in \Re_0^+$	Relative time of the packet capture (L1, $*$).
$size \in \aleph$	Size of the whole Ethernet frame including Ethernet header (L1, $*$).
$eth_{src} \in \{0, \ldots, 2^{48}-1\}$	Source MAC address of the frame (L2, Ethernet).
$eth_{dst} \in \{0, \ldots, 2^{48}-1\}$	Destination MAC address of the frame (L2, Ethernet).
$ip_{off} \in \{0, \ldots, 2^{13}-1\}$	Offset field (L3, IPv4).
$ip_{ttl} \in \{0, \ldots, 2^{8}-1\}$	Time to live field (L3, IPv4).
$ip_{p} \in \{0, \ldots, 2^{8}-1\}$	Protocol field (L3, IPv4).
$ip_{sum} \in \{0, \ldots, 2^{16}-1\}$	Checksum of the header (L3, IPv4).
$ip_{src} \in \{0, \ldots, 2^{32}-1\}$	Source IP address of the packet (L3, IPv4).
$ip_{dst} \in \{0, \ldots, 2^{32}-1\}$	Destination IP address of the packet (L3, IPv4).
$ip_{dscp} \in \{0, \ldots, 2^{8}-1\}$	Differentiated services code point field (L3, IPv4).
$tcp_{sport} \in \{0, \ldots, 2^{16}-1\}$	Source port of the packet (L4, TCP).
$tcp_{dport} \in \{0, \ldots, 2^{16}-1\}$	Destination port of the packet (L4, TCP).
$tcp_{sum} \in \{0, \ldots, 2^{16}-1\}$	Checksum of the header (L4, TCP).
$tcp_{seq} \in \{0, \ldots, 2^{32}-1\}$	Sequence number of the packet (L4, TCP).
$tcp_{ack} \in \{0, \ldots, 2^{32}-1\}$	Acknowledgment number of the packet (L4, TCP).
$tcp_{off} \in \{0, \ldots, 2^{8}-1\}$	Offset and reserved fields together (L4, TCP).
$tcp_{flags} \in \{0, \ldots, 2^{8}-1\}$	Control bits (L4, TCP).
$tcp_{win} \in \{0, \ldots, 2^{16}-1\}$	Window field (L4, TCP).
$tcp_{urp} \in \{0, \ldots, 2^{16}-1\}$	Urgent pointer field (L4, TCP).
$data \in \{0, \ldots, 2^{8}-1\}^*$	Payload of the packet (L7, $*$).

IX. CONCLUSION

In this paper, we presented three datasets consisting of extracted high-level network features (ASNM features). These datasets are intended for non-payload-based network intrusion detection and adversarial classification, enabling to test evasion resistance of machine learning-based classifiers. In detail, ASNM-CDX-2009 dataset might serve for basic benchmarking of machine learning-based classifiers, while ASNM-TUN and ASNM-NPBO datasets might serve for more advanced benchmarking of these classifiers, such as testing the classifiers on evasion resistance. In future work, we plan to extend ASNM datasets with data collected from the most recent network vulnerabilities.

APPENDIX A
TUPLES OF TCP CONNECTION AND PACKET

We detail the TCP connection tuple in Table 24. In the table, the superscript at $\mathbb{R}^*$ represents a set of positive real numbers,

Table 26. ASNM features selected by FFS with the Naive Bayes classifier using ASNM-TUN dataset.

Feature ID	Description	FFS DOL	FFS DL
SigPktLenIn	• Std. deviation of inbound (server to client) packet sizes.	X	
ConTcpFinCntIn	• The number of TCP FIN flags occurred in inbound traffic.	X	X
ConTcpSynCntIn	• The number of TCP SYN flags occurred in inbound traffic.	X	X
InPktLen32s10i[0]	• Lengths of inbound packets occurred in the first 32 seconds of a connection which are distributed into 10 intervals. The feature represents totaled inbound packet lengths of the 1st interval.	X	
InPktLen1s10i[2]	• The same as the previous one, but computed above the first second of a connection. The feature represents totaled inbound packet lengths of the 3rd interval.	X	
InPktLen8s10i[7]	• The same as the previous one, but computed above the first 8 seconds of a connection. The feature represents totaled inbound packet lengths of the 8th interval.	X	
OutPktLen1s10i[0]	• Lengths of outbound (client to server) packets occurred in the first second of a connection which are distributed into 10 intervals. The feature represents totaled outbound packet lengths of the 1st interval.	X	
FourGonAngleN[9]*	• Fast Fourier Transformation (FFT) of all packet sizes. The feature represents the angle of the 10th coefficient of the FFT in goniometric representation.	X	X
InPktLen8s10i[1]	• Lengths of inbound packets occurred in the first 8 seconds of a connection which are distributed into 10 intervals. The feature represents totaled inbound packet lengths of the 2nd interval.		X
PolyInd8ordOut[5]	• Approximation of outbound packet lengths in index domain by polynomial of 8th order. The feature represents 6th coefficient of the polynomial.		X
PolyInd8ordIn[5]	• Approximation of inbound packet lengths in index domain by polynomial of 8th order. The feature represents 6th coefficient of the polynomial.		X

*Sizes of inbound and outbound packets are represented by negative and positive values, respectively.

Table 27. ASNM features selected by FFS with the Naive Bayes classifier using ASNM-CDX-2009 dataset.

Feature ID	Description
ConTcpFinCntOut	• The number of TCP FIN flags occurred in outbound traffic.
FourGonModulIn[2]	• Fast Fourier Transformation (FFT) of inbound packet sizes. The feature represents the module of the 3rd coefficient of the FFT in goniometric representation.
FourGonModulIn[18]	• Fast Fourier Transformation (FFT) of outbound packet sizes. The feature represents the module of the 19th coefficient of the FFT in goniometric representation.
GaussProds8All[7]	• Normalized products of all packet sizes with 8 Gaussian curves. The feature represents a product of the 8th slice of packets with a Gaussian function that fits the interval of the packets' slice.
InPkt1s20iTr4KB[7]	• The accumulated number of inbound packets occurred in the 1st second of a connection distributed into 20 intervals, each after accumulation of 4KB of data. The feature represents the 8th component of the vector.
InPkt1s20iTr4KB[16]	• The same as the previous one, but it represents the 16th component of the vector.
InPkt1s20iTr4KB[19]	• The same as the previous one, but it represents the 19th component of the vector.
InPkt1s20iTr4KB[14]	• The same as the previous one, but it represents the 14th component of the vector.
InPkt1s20iTr4KB[14]	• The same as the previous one, but the interval used is 64 seconds and threshold used is 4KB.
OutPkt64s20iTr2KB[9]	• The accumulated number of outbound packets occurred in the first 64 seconds of a connection distributed into 20 intervals, each after accumulation of 2KB of data. The feature represents the 10th component of the vector.
InPkt64s10i[6]	• the number of inbound packets occurred in the first 64 seconds of a connection, which are distributed into 10 intervals. The feature represents the 6th component of the vector.
ModPktLenIn	• Mode of packet sizes in inbound traffic of a connection.
ModTosIp	• Mode of TOS (type of service) field in IP headers of all traffic.
PolyInd3ordOut[3]	• Approximation of outbound packet lengths in index domain by polynomial of 3rd order. The feature represents 4th coefficient of the polynomial.
PolyInd10ordIn[1]	• The same as the previous one, but the order of the polynomial is 10 and the coefficient number is 2.
polyTime8ordOut[2]	• The same as the previous one, but the order of the polynomial is 8 and the coefficient number is 3.
polyTime8ordOut[5]	• The same as the previous one, but representing the 6th coefficient.
polyTime10ordIn[2]	• The same as the previous one, but the order of the polynomial is 10 and the coefficient number is 3.
polyTime10ordOut[7]	• The same as the previous one, but representing the 8th coefficient.
polyTime13ordIn[8]	• The same as the previous one, but the order of the polynomial is 13 and the coefficient number is 9.

Table 28. ASNM features selected by FFS using the Naive Bayes classifier.

Feature ID	Description	FFS DOL	FFS DL
SigPktLenOut	• Std. deviation of outbound (client to server) packet sizes.	X	X
MeanPktLenIn	• Mean of packet sizes in inbound traffic of a connection.	X	X
CntOfOldFlows	• The number of mutual connections between client and server, which started up to 5 minutes before start of an analyzed connection.	X	X
CntOfNewFlows	• The number of mutual connections between client and server, which started up to 5 minutes after the end of an analyzed connection.	X	X
ModTCPHdrLen	• Modus of TCP header lengths in all traffic.	X	
UrgCntIn	• The number of TCP URG flags occurred in inbound traffic.	X	
FinCntIn	• The number of TCP FIN flags occurred in inbound traffic.		X
PshCntIn	• The number of TCP PUSH flags occurred in inbound traffic.		X
FourGonModulIn[1]	• Fast Fourier Transformation (FFT) of inbound packet sizes. The feature represents the module of the 2nd coefficient of the FFT in goniometric representation.	X	X
FourGonModulOut[1]	• The same as the previous one, but it represents the module of the 2nd coefficient of the FFT for outbound traffic.		X
FourGonAngleOut[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X	
FourGonAngleN[9]	• Fast Fourier Transformation (FFT) of all packet sizes, where inbound and outbound packets are represented by negative and positive values, respectively. The feature represents the angle of the 10th coefficient of the FFT in goniometric representation.	X	X
FourGonAngleN[1]	• The same as the previous one, but it represents the angle of the 2nd coefficient of the FFT.	X	
FourGonModulN[0]	• The same as the previous one, but it represents the module of the 1st coefficient of the FFT.		X
PolyInd13ordOut[13]	• Approximation of outbound communication by polynomial of 13th order in the index domain of packet occurrences. The feature represents the 14th coefficient of the approximation.	X	
PolyInd3ordOut[3]	• The same as the previous one, but it represents the 4th coefficient of the approximation.		X
GaussProds8All[1]	• Normalized products of all packet sizes with 8 Gaussian curves. The feature represents a product of the 2nd slice of packets with a Gaussian function that fits the interval of the packets' slice.	X	
GaussProds8Out[7]	• The same as the previous one, but computed above outbound packets and represents a product of the 8th slice of packets with a Gaussian function that fits the interval of the packets' slice.		X
InPktLen1s10i[5]	• Lengths of inbound packets occurred in the first second of a connection, which are distributed into 10 intervals. The feature represents totaled outbound packet lengths of the 6th interval.	X	
OutPktLen32s10i[3]	• The same as the previous one, but computed above the first 32 seconds of a connection. The feature represents totaled outbound packet lengths of the 4th interval.		X
OutPktLen4s10i[2]	• The same as the previous one, but computed above the first 4 seconds of a connection. The feature represents totaled outbound packet lengths of the 3rd interval.		X

including zero. Next, we present a detailed description of items from the packet tuple in Table 25. The description contains an assignment to a particular layer of the ISO/OSI stacked model together with supported instances of the protocols – placeholder $*$ represents an arbitrary protocol instance.

Also note that in the case of the *data* field, superscript in X^* represents an iteration of the set X.

APPENDIX B
FFS SELECTED FEATURES

As part of benchmarking ASNM datasets, in this section, we enumerate subsets of the ASNM features selected by the FFS method with Naive Bayes classifier running over 5-fold cross-validation. We present ASNM features selected using: (1) the ASNM-CDX-2009 dataset in Table 27, (2) the ASNM-TUN dataset in Table 26, and (3) the ASNM-NPBO dataset in Table 28. Note that the FFS DL set denotes features selected when only legitimate samples and direct attacks were included in the FFS experiment. The FFS DOL set denotes selected features when, in addition to the previous case, obfuscated attacks were included in the FFS experiment.

REFERENCES

[1] *Top Intrusion Attacks*. Accessed: Jun. 30, 2019. [Online]. Available: https://www.mcafee.com/threat-intelligence/ips/top-attacks.aspx

[2] *Top Targeted Vulnerabilities*. Accessed: Jun. 30, 2019. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA15-119A

[3] H. Debar, M. Dacier, and A. Wespi, "A revised taxonomy for intrusion-detection systems," *Ann. Telecommun.*, vol. 55, no. 7, pp. 361–378, 2000.

[4] S. A. Ludwig, "Applying a neural network ensemble to intrusion detection," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 3, pp. 177–188, Jul. 2019.

[5] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.

[6] P. Tao, Z. Sun, and Z. Sun, "An improved intrusion detection algorithm based on ga and SVM," *IEEE Access*, vol. 6, pp. 13624–13631, 2018.

[7] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[8] R. A. Niazi and Y. Faheem, "A Bayesian game-theoretic intrusion detection system for hypervisor-based software defined networks in smart grids," *IEEE Access*, vol. 7, pp. 88656–88672, 2019.

[9] I. Homoliak, D. Breitenbacher, and P. Hanacek, "Convergence optimization of backpropagation artificial neural network used for dichotomous classification of intrusion detection dataset," *J. Comput.*, vol. 12, no. 2, pp. 143–155, 2017.

[10] I. Homoliak, M. Barabas, P. Chmelar, M. Drozd, and P. Hanacek, "ASNM: Advanced security network metrics for attack vector description," in *Proc. Conf. Secur. Manage.*, 2013, pp. 350–358.

[11] *Tcpdump.org*. Accessed: May 3, 2020. [Online]. Available: http://www.tcpdump.org/

[12] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, W. J. Adams, C. Morrell, and G. Conti, "Toward instrumenting network warfare competitions to generate labeled datasets," in *Proc. 2nd Workshop Cyber Secur. Experimentation Test (CSET)*, 2009.

[13] I. Homoliak, D. Ovšonka, M. Grégr, and P. Hanáček, "NBA of obfuscated network vulnerabilities' exploitation hidden into HTTPS traffic," in *Proc. 9th Int. Conf. Internet Technol. Secured Trans. (ICITST)*, Dec. 2014, pp. 311–318.

[14] I. Homoliak, "Intrusion detection in network traffic," Ph.D. dissertation, Dept. Inf. Technol., Univ. Technol., Brno, Czechia, 2016.

[15] A. W. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Intel Research, Cambridge, MA, USA, Tech. Rep. RR-05-13, 2005.

[16] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, "Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation," in *Proc. 1st Workshop Building Anal. Datasets Gathering Exp. Returns Secur.*, 2011, pp. 29–36.

[17] (1999). *KDD Cup 99*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[18] *NetFlow*, Cisco Syst., San Jose, CA, USA, 2019.

[19] A. Habibi Lashkari, G. Draper Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. ICISSP*, 2017, pp. 253–262.

[20] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Sep. 2015, pp. 134–142.

[21] I. Homoliak, L. Sulak, and P. Hanacek, "Features for behavioral anomaly detection of connectionless network buffer overflow attacks," in *Proc. Int. Workshop Inf. Secur. Appl.* Jeju, South Korea: Springer, 2016, pp. 66–78.

[22] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 2, 1995, pp. 1137–1145.

[23] I. Homoliak, M. Teknös, M. Ochoa, D. Breitenbacher, S. Hosseini, and P. Hanacek, "Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach," *ICST Trans. Secur. Saf.*, vol. 5, no. 17, Jan. 2019, Art. no. 156245.

[24] M. Barabas, I. Homoliak, M. Kacic, and P. Hanacek, "Detection of network buffer overflow attacks: A case study," in *Proc. 47th Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2013, pp. 1–4.

[25] I. Homoliak, "Metrics for intrusion detection in network traffic," M.S. thesis, Dept. Intell. Syst., Inf. Technol., Univ. Technol., Brno, Czechia, (in Slovak), 2011.

[26] (2011). *Cyber Research Center: Data Sets*. [Online]. Available: [Online]. Available: https://westpoint.edu/centers-and-research/cyber-research-center/data-s%ets

[27] Cisco. (2019). *SNORT*. [Online]. Available: https://www.snort.org/

[28] NIST. *CVE-2002-0082: Buffer Overflow Vulnerability of Mod_ssl and Apache-SSL*. Accessed: May 3, 2020. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2002-0082

[29] (2019). *Rapid7: Remotely exploitable buffer overflow in mod_ssl*. [Online]. Available: https://www.rapid7.com/db/vulnerabilities/HTTP-MODS-0003

[30] (2019). *Rapid7: Badblue 2.72b Passthru Buffer Overflow*. [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/http/badblue_passthru

[31] NIST. *CVE-2007-6377: Stack-Based Buffer Overflow Vulnerability in BadBlue*. Accessed: May 3, 2020. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2007-6377

[32] NIST. *CVE-2003-0352: Buffer Overflow in DCOM Interface for RPC in MS Windows NT 4.0, 2000, XP and Server 2003*. Accessed: May 3, 2020. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0352

[33] (2019). *Rapid7: MS03-026 Microsoft RPC DCOM Interface Overflow*. [Online]. Available: https://www.rapid7.com/db/modules/exploit/windows/dcerpc/ms03_026_dcom

[34] NIST. *CVE-2003-0201: Buffer Overflow in the Samba Service*. Accessed: May 3, 2020. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2003-0201

[35] (2019). *Rapid7: Samba Trans2open overflow (Linux x86)*. [Online]. Available: https://www.rapid7.com/db/modules/exploit/linux/samba/trans2open

[36] Open Information Security Foundation. (2019). *Suricata IDS*. [Online]. Available: http://suricata-ids.org/

[37] *VirusTotal—Virus, Malware and URL Scanner*. [Online]. Available: https://www.virustotal.com/

[38] (2019). *Rapid7: Metasploitable—Virtual Machine to Test Metasploit*. [Online]. Available: https://information.rapid7.com/metasploitable-download.html

[39] (2019). *Rapid7: Tomcat Application Manager Login Utility*. [Online]. Available: http://www.rapid7.com/db/modules/auxiliary/scanner/http/tomcat_mgr_logi%n

[40] (2019). *Rapid7: Apache Tomcat Manager Application Deployer Authenticated Code Execution*. [Online]. Available: http://www.rapid7.com/db/modules/exploit/multi/http/tomcat_mgr_deploy

[41] (2019). *Rapid7: MSSQL Login Utility*. [Online]. Available: https://www.rapid7.com/db/modules/auxiliary/scanner/mssql/mssql_login

[42] (2019). *Rapid7: Microsoft SQL Server Payload Execution*. [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/mssql/mssql_payload

[43] (2019). *Rapid7: Samba Username Map Script Command Execution*. [Online]. Available: http://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script

[44] (2019). *Rapid7: Microsoft Server service Relative Path Stack Corruption*. [Online]. Available: http://www.rapid7.com/db/modules/exploit/windows/smb/ms08_067_netapi
[45] (2019). *Rapid7: PostgreSQL Login Utility*. [Online]. Available: http://www.rapid7.com/db/modules/auxiliary/scanner/postgres/postgres_lo%gin
[46] (2019). *Rapid7: PostgreSQL for Linux Payload Execution*. [Online]. Available: [Online]. Available: http://www.rapid7.com/db/modules/exploit/linux/postgres/postgres_payloa%d
[47] *Rapid7: DistCC Daemon Command Execution*. (2019). [Online]. Available: http://www.rapid7.com/db/modules/exploit/unix/misc/distcc_exec
[48] I. Homoliak, M. Teknos, M. Barabas, and P. Hanacek, "Exploitation of netem utility for non-payload-based obfuscation techniques improving network anomaly detection," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.* Guangzhou, China: Springer, 2016, pp. 770–773.
[49] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues," *Inf. Sci.*, vol. 239, pp. 201–225, Aug. 2013.
[50] K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications," *Biometrika*, vol. 57, no. 3, pp. 519–530, 1970.
[51] A. Justel, D. Peña, and R. Zamar, "A multivariate Kolmogorov-Smirnov test of goodness of fit," *Statist. Probab. Lett.*, vol. 35, no. 3, pp. 251–259, Oct. 1997.
[52] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. (2009). *NSL-KDD Dataset*. [Online]. Available: https://web.archive.org/web/20150205070216/http://nsl.cs.unb.ca/NSL-KDD%/
[53] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J. Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[54] A. Verma and V. Ranga, "Evaluation of network intrusion detection systems for RPL based 6LoWPAN networks in IoT," *Wireless Pers. Commun.*, vol. 108, no. 3, pp. 1571–1594, Oct. 2019.
[55] (2020). *IBM SPSS Statistics*. [Online]. Available: https://www.ibm.com/products/spss-statistics
[56] *RapidMiner: RapidMiner Studio*. [Online]. Available: https://rapidminer.com/products/studio/
[57] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
[58] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning Proceedings 1994*. Amsterdam, The Netherlands: Elsevier, 1994, pp. 121–129.
[59] I. Homoliak, D. Ovšonka, K. Koranda, and P. Hanáček, "Characteristics of buffer overflow attacks tunneled in HTTP traffic," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2014, pp. 188–193.
[60] K. Z. Mao, "Orthogonal forward selection and backward elimination algorithms for feature subset selection," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 34, no. 1, pp. 629–634, Feb. 2004.
[61] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognit. Lett.*, vol. 15, no. 11, pp. 1119–1125, Nov. 1994.
[62] S. Nakariyakul and D. P. Casasent, "An improvement on floating search algorithms for feature subset selection," *Pattern Recognit.*, vol. 42, no. 9, pp. 1932–1940, Sep. 2009.
[63] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern Recognit.*, vol. 43, no. 1, pp. 5–13, Jan. 2010.
[64] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.
[65] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, Oct. 2017.
[66] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of classifiers," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3357–3368, Apr. 2019.
[67] O. Chapelle, V. Sindhwani, and S. S. Keerthi, "Optimization techniques for semi-supervised support vector machines," *J. Mach. Learn. Res.*, vol. 9, no. 6, pp. 203–233, 2008.
[68] M. Hatada, M. Akiyama, T. Matsuki, and T. Kasama, "Empowering anti-malware research in japan by sharing the MWS datasets," *J. Inf. Process.*, vol. 23, no. 5, pp. 579–588, 2015.
[69] (2019). *PREDICT dataset: Protected Repository for the Defense of Infrastructure against Cyber Threats*. [Online]. Available: https://www.dhs.gov/publication/dhsstpia-006-protected-repository-defen%se-infrastructure-against-cyber-threats
[70] (2019). *CAIDA: The Cooperative Association for Internet Data Analysis*. [Online]. Available: http://www.caida.org/
[71] *Netresec—Publicly Available PCAP Files*. (2019). [Online]. Available: http://www.netresec.com/?page=PcapFiles
[72] (Jan. 13, 2014). *DARPA Intrusion Detection Evaluation*. [Online]. Available: http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/
[73] F. Massicotte, F. Gagnon, Y. Labiche, L. Briand, and M. Couture, "Automatic evaluation of intrusion detection systems," in *Proc. 22nd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 2006, pp. 361–370.
[74] (2015). *Contagio Malware Dump: Collection of PCAP Files From Malware Analysis*. [Online]. Available: http://contagiodump.blogspot.cz/2013/04/collection-of-pcap-files-from-m%alware.html
[75] M. Hatada, M. Akiyama, T. Matsuki, and T. Kasama, "Empowering anti-malware research in Japan by sharing the MWS datasets," *J. Inf. Process.*, vol. 23, no. 5, pp. 579–588, 2015.
[76] M. Hatada, I. Nakatsuru, and M. Akiyama, "Datasets for anti-malware research—MWS 2011 datases," (in Japanese), in *Proc. IPSJ Malware Workshop (MWS)*, 2011, pp. 19–21.
[77] M. Hatada, Y. Nakatsuru, M. Akiyama, and S. Miwa, "Datasets for anti-malware research—MWS 2010 datasets," in *Proc. IPSJ Malware Workshop (MWS)*, 2010, pp. 1–5.
[78] M. Hatada, Y. Nakatsuru, M. Terada, and Y. Shinoda, "Dataset for anti-malware research and research achievements shared at the workshop," in *Proc. Comput. Secur. Symp.*, 2009, pp. 1–8.
[79] M. Kamizono, M. Hatada, M. Terada, M. Akiyama, T. Kasama, and J. Murakami, "Datasets for anti-malware research—MWS datasets 2013," in *Proc. IPSJ Comput. Secur. Symp.*, 2013, pp. 1–8.
[80] (2019). *The UCSD CAIDA Backscatter Dataset*. [Online]. Available: http://www.caida.org/data/passive/backscatter_dataset.xml
[81] (2019). *The CAIDA UCSD. DDoS Attack 2007 dataset*. [Online]. Available: http://www.caida.org/data/passive/ddos-20070804_dataset.xml
[82] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescapé, "Analysis of a '/0' stealth scan from a Botnet," in *Proc. ACM Conf. Internet Meas. Conf. (IMC)*, 2012, pp. 1–14.
[83] Three Days Of Conficker. (2019). *The CAIDA UCSD Network Telescope*. [Online]. Available: http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml
[84] (2019). *The UCSD CAIDA Dataset on the Code-Red Worms*. [Online]. Available: http://www.caida.org/data/passive/codered_worms_dataset.xml
[85] *The CAIDA UCSD Dataset on the Witty Worm*. (2019). [Online]. Available: http://www.caida.org/data/passive/witty_worm_dataset.xml
[86] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of DARPA dataset for intrusion detection system evaluation," in *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security* (SPIE Proceedings), vol. 6973, B. V. Dasarathy, Ed. Bellingham, WA, USA: SPIE, Mar. 2008, p. 69730G, doi: 10.1117/12.777341.
[87] D. Song. *Fragroute*. Accessed: May 3, 2020. [Online]. Available: http://www.monkey.org/~dugsong/fragroute/
[88] R. F. Puppy. (1999). *A Look at Whisker's Anti-IDS Tactics*. [Online]. Available: http://www.ussrback.com/docs/papers/IDS/whiskerids.html
[89] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proc. LISA*, 1999, vol. 99, no. 1, pp. 229–238.
[90] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, nos. 23–24, pp. 2435–2463, Dec. 1999.
[91] A. Sperotto, R. Sadre, F. Van Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *Proc. Int. Workshop IP Operations Manage.* Venice, Italy: Springer, 2009, pp. 39–50.
[92] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[93] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 53–58.
[94] M. Salem, S. Reissmann, and U. Buehler, "Persistent dataset generation using real-time operative framework," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2014, pp. 1023–1027.
[95] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–40, May 2019.

[96] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the JAM project," in *Proc. DARPA Inf. Survivability Conf. Expo. (DISCEX00)*, vol. 2, 2000, pp. 130–144.
[97] A. Moore, J. Hall, C. Kreibich, E. Harris, and I. Pratt, "Architecture of a network monitor," in *Proc. Passive Act. Meas. Workshop (PAM)*, 2003.
[98] (2006). *Kyoto 2006+ Dataset*. [Online]. Available: http://www.takakura.com/Kyoto_data/
[99] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

IVAN HOMOLIAK was born in Rimavska Sobota, Slovakia, in 1987. He received the B.S. degree in information technology, the M.S. degree in intrusion detection and supervised machine learning, and the Ph.D. degree in adversarial intrusion detection in network traffic from the Faculty of Information Technology (FIT), Brno University of Technology (BUT), Czech Republic, in 2007, 2012, and 2016, respectively. From 2016 to 2020, he was a Postdoctoral Research Fellow with the Singapore University of Technology and Design (SUTD), and he worked in the project aimed at distributed computing and blockchains. Prior to that, Ivan worked on the project focusing on insider threat detection in SUTD, where he pursued the application of ML in this area and analysis of security issues. Currently, he is the research scientist at FIT BUT, and he focuses on security aspects of blockchains and their applications.

KAMIL MALINKA was born in Valtice, Czech Republic, in 1982. He received the M.S. degree in applied informatics from Masaryk University (MU), in 2005, and the Ph.D. degree in biometrics and anonymity systems from the Faculty of Information Technology (FIT), Brno University of Technology (BUT), Czech Republic, in 2010. He currently works as an Assistant Professor with the FIT, BUT. He also works as an IT Architect and a Researcher with MU, and he is a member of the local security research group Security@FIT, which focuses on research in the fields of computer and network security.

PETR HANACEK was born in Uherske Hradiste, Czech Republic, in 1964. He received the M.S. degree in computer engineering, the Ph.D. degree in computer science, and the Habilitation degree from the Brno University of Technology (BUT), Czech Republic, in 1988, 1997, and 2003, respectively. From 1987 to 2001, he worked at the Department of Computer Science, Faculty of Electrical Engineering and Computer Science, BUT. He is currently the Head of the Department of Intelligent Systems and works as an Associate Professor with the Faculty of Information Technology (FIT), BUT. He leads the local security research group Security@FIT, which focuses on research in the fields of computer and network security. Since 2002, he has been with the FIT, BUT. He is a member of the Czech and Slovak Information Society (CIS) and the Czech and Slovak Simulation Society (CSSS), and a member of the Special Interest Group on Security, Audit, and Control (ACM–SIGSAC).