Traffic Generator Based on Behavioral Pattern

Matej Kacic, Daniel Ovsonka, Petr Hanacek and Maros Barabas
Faculty of Information Technology
Brno University of Technology
Boetechova 2, 612 66 Brno, Czech Republic
Email: ikacic@fit.vutbr.cz

***Abstract*—Network traffic generation was the subject of many research projects in the past, but none of them could generate network data which has the same nature as a traffic from human or machine behavior. In this paper we introduce a related work in this field of area focusing on advantages and disadvantages, such as authenticity, inaccuracy, wrong timings and real-time generation. Further, we propose a generator capable of generating traffic with predefined behavioral pattern of a valid communication. The high diversity and randomness of generated traffic allowed us to use this generator for testing of our reputation system, which evaluates value of reputation based on behavioral pattern by analysing network traffic.**

***Keywords*—Network traffic generator, behavioral pattern, wireless, http traffic, malware injection, security**

I. Introduction

The main idea of the proposed solution is to create a traffic generator, which is capable of generating frames/packets from L2 layer to L7 layer with high authenticity and accuracy.The goal is to create a system capable of generating traffic that is statistically almost identical to a genuine (real, valid) network traffic, but also providing high extensibility and performance. One of the main requirements on the system is a mechanism of defining rules, which can describe a behavioral pattern of each entity within the network. This behavioral pattern is further used in process of generating traffic to simulate the real behavior of an entity, but with different data and/or properties.

The basic prerequisite for the system is the real network data, which is based on stored PCAP dump files previously obtained from a real network traffic. The key part of the system is a mathematical model, whose purpose is to describe the behavior of entities on the network. The genuine network traffic is analysed and based on previously defined set of rules (composed of protocol dependent and independent rules), a model of analysed entity is created. Based on these rules the obtained network communication is processed and altered to increase diversity and randomness of generated traffic, but in a way to keep consistency of the communication.

As it was already mentioned, the major requirement is similarity of generated traffic data to a real traffic created by network users or machines, and it should be difficult or even impossible to recognize generated traffic from the genuine one. For example, this could be achieved by simulating user behavior by using graphic user interface testing tools to automatically generate network traffic.

Based on these findings and requirements we created a network traffic generator built on a large database of genuine network traffic data. Original network traffic data are transformed to a traffic, adjusted according to user settings. The generator engine uses probability-based algorithms, therefore the output is always different even if the settings are the same. The database of genuine network data is used as a source for the engine but all the relevant fields in protocol headers are modified. The prototype of generator currently supports Hyper Text Transport Protocol (HTTP) and HTTP Secure (HTTPS) protocol and is easily extensible by any other text-based protocol. The only requirements for adding protocol extension is the existence of source data (dump of genuine network traffic) for the protocol and user-defined protocol-dependent rewrite rules. The advantage of this approach is the possibility to specify the size, duration or number of appearing entities in the output (i.e. generated) traffic. This allows a user to create various scenarios for later analysis or research purpose.

The paper is divided as follows. Section II covers the summary of our investigation of state of the art in the area of network traffic generators. High level design architecture is described in section III and HTTP traffic generation as the most complex model of our solution is proposed in section V. The paper is concluded with results and test scenario in section VI.

II. Related work

Traffic generation solutions can be classified into three main groups: trace-based, model-based, and testing network devices. In the following sections we describe the main ideas of each approach with their advantages, disadvantages and references to related research.

A. Model-based approaches

The model-based approach uses a stochastic traffic model where parameters of the model are based on traffic characteristics of measured data. The main disadvantage of this approach is the use of very sophisticated and complex models to get high accuracy results. In many cases the model has so many parameters that it is impossible to implement it.

The article [1] introduced new dynamic application-level protocol replay techniques using statistical models of user behavior to generate workloads on the testbed hosts. They emulate remote services by re-assembling real traffic to a very

high degree of fidelity. They also control windows applications like Internet Explorer or Outlook to generates traffic. This solution provides very low variety of applications protocols and generating itself depends on windows platform.

Authors and co. [2] proposed a method, called Event-driven Automata Synchronized Replay (EAR), to address real traffic replay over wireless LAN. EAR transforms the captured packet trace into a sequence of events that follow the IEEE 802.11 protocol. The three-level automata are applied to achieve packet-replay control and synchronize the environment effects in traffic replay with the packets and signals captured in a real environment.

Authors in [3] presented the evaluation of LiTGen, a realistic IP traffic model for generation of IP traffic with accurate time scale properties and performance. They confront LiTGen against real data traces using two methods of evaluation. These methods respectively allow to observe the causes and consequences of the traffic burstiness. The result of their work points the importance of precise modelling of random variables distributions involved in the underlying mode. It should be noticed that the IP model used in LiTGen is quite simple, thus the synthetic traces do not reflect any behavioral pattern.

B. Trace-based approaches

The trace-based approach uses data / measurements from real environment. This kind of data contain real payload and real packets (frames) headers, therefore the authenticity and accuracy is guaranteed. Correct inter-packet timings strictly depend on specific solution [4].

he result of this approach is privacy issues, which are the reason why this solution cannot be applicable to production environment. This approach has many disadvantages, a major handicap is that communication must be generated in real-time, therefore it is not effective for larger test-cases. The extensibility of this approach is also difficult because a user has to rewrite or create new scripts and prepare experimental environment with all required applications, which is extremely time consuming.

A solution Tmix proposed in [5] uses real network traces to replay correct packet inter-arrival timing and ns2 simulator to mimic the TCP behavior. It starts from trace of TCP/IP headers on a production network, and then model is constructed for all the TCP connections observed in the network. Tmix algorithm is reused to replay the connections and reproduce the application-level behaviors observed on the original network.The main disadvantage of this solution is absence of realistic packet payload.

Last research in this area is Multi-Functional Emulator for Traffic Analysis [4], where authors presented user behavior based traffic emulation system, which uses real measurements to achieve a complete payload and realistic inter-packet timing. They emulate network applications on different platforms (Windows, Android)and different technologies (wired, Wifi, 3g) by remotely controlling those applications. The emulator needs real running devices to eavesdrop the traffic traces from the network whenever the change of behavioral model occurs.

C. Testing network devices

Generators belonging to this category represent high performance packet generators. We have investigated many existing solutions which are used for testing network devices, but there is a problem with authenticity, inaccuracy of packets, and wrong or insufficient timings which do not correspond to a real network.

The KUTE is UDP packet generator and receiver which runs entirely in the Linux kernel [6]. It has been implemented with purpose of sending and receiving higher packet rates on high speed network interfaces (e.g. Gigabit Ethernet). KUTE, unlike other generators which are using user-space environment, can handle inter-packet times more accurately. There are number of tools (RUDE [7], MGEN [8]) which work very similar to KUTE.

The article in [9] analyzes four of the most used packet-level traffic generators (RUDE, MGEN, KUTE, D-ITG Fig. 1. High level design of Traffic generator [10]) shows how they fail to follow the requested profiles. The generated traffic profiles are very different from those requested.They also identify problems affecting their accuracy.

Another approach is used in BRUNO [11], which is based on cooperative PC/NP architecture. An advanced software tool runs on a host PC and instructs the processing engines of an Intel IXP2400 Network Processor, which takes care of the actual traffic generation. This approach can achieve traffic load of 800 Mbps. Other solutions, described in previous paragraph, achieve maximum traffic load of 400 Mbps.

All solutions in this category have great performance and accurate timing, but they are unsuitable for generating different types of traffic (TCP/UDP, HTTP, FTP, etc.). In our system we focus on generating traffic based on defined rules with corresponding behavioral patterns and we focus on precise inter-packet timings with realistic data payload. Our generator is not designed for real-time stress testing of network devices, but we can retransmit generated traces directly to the wired or WiFi networks.

III. HIGH LEVEL DESIGN

This section describes in detail the scheme and also the dataflow of the traffic generator. The system can be separated into several modules, each having specific use:

This section describes in detail the scheme and also the dataflow of the traffic generator. The system can be separated into several modules, each having specific use.

A. Calendar

Calendar is a graphic, web based module which offers the possibility of easily adding behavioral patterns to the system. It is called calendar because of format of data entry where every behavioral pattern is represented by a rule added to particular timeslot. Every single rule defines the time interval, duration and protocol specific details of simulated behavior. In case of simulating HTTP traffic, there is an option to specify which domains will be visited in simulated communication and also how much data will be transferred. Users can create or modify

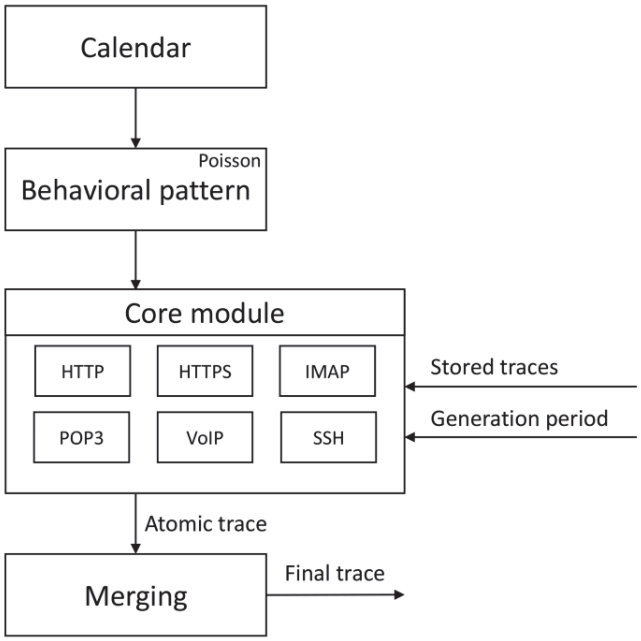

Fig. 1. High level design of Traffic generator

these rules that are subsequently used for the traffic simulation. Implementation of this module is based on SQL database, used as a storage of all created rules, it serves as a metadata for stored PCAP files and it also creates the interconnection with the other modules.

B. The Core module

This module provides the key functionality of the network traffic generator. This module takes multiple inputs; database of captured PCAP files and also information from calendar module. The input data are processed to generate the output in form of PCAP file. The module consists of several submodules where each of them is used to generate one specific protocol communication. The core module is easily extensible by a custom module which can generate any arbitrary protocol. The implementation of this module provides a layer to easily transform low level protocols like DNS, DHCP etc. Our actual work aims to generating HTTP traffic, so the most important part is HTTP submodule which is described in section V.

C. WiFi module

One of the most important criteria for determining the wireless users behavior is user mobility. The mobility pattern can be very different in a time, for the reason of nature of our environment, habits and modern devices.

There are also some device types with very high mobility - mobile phones and tablets with VoIP services. An example of this type of user might be a person using VoIP application while walking the office.

We can simulate the user mobility by different approaches:

- Changing Access Points - user can move between access points in time i.e. we can simulate this movement in time of day.
- Location of wireless device - changing the value of signal strengths in radio header.

The WiFi module is an extension of generation core. It extends the core module with capability of generating wireless frames according to standard 802.11i [**?**] (encrypting, decrypting L2 frames, computing of correct initialization vector and checksums).

First, the live communication is sniffed from live wireless environment for multiple devices. Sniffed communication is decrypted and stored into pcap files. The system preserves a radiotap header and MAC header untouched. In order to keep compatibility with the core module, decrypted payload of the wireless frames is transformed into ethernet form. When the core module returns newly generated data, this module will develop a framework [**?**] for this purpose, which is capable of sniffing, managing encryption keys (PTK/GTK), decrypting and encrypting data via stored encryption keys, etc.

D. Malware module

This module adds the malicious packets into generated network traffic. We have a collection of basic types of malware, which is then analysed in sandbox environment. For each malware the network communication are generated and stored. These traffic traces are imported into our traffic generator, where the malicious communication is merged and combined with generated traffic from core module.

Malware module is also capable of generating Distributed Denial-of-Service (DDoS) attacks data. In order to generate DDoS network traffic we have to specify the number of bots, packets per second and kilobytes per second. This generation is based on a framework for generating realistic traffic for DDoS attacks and Flash Events [15].

IV. Data Acquisition

The important task in the context of this work was to create a large database of real network traffic for each simulated protocol. The number of captured packets for each protocol gives us more variability while generating simulated network traffic, so data gathered from real network traffic is key part of our system. One of the goals of the project is to provide customizable environment for users, therefore we also provide a way to add additional previously captured data from users. The data for simulating HTTP protocol were obtained from a virtual network which consists of computers with different operation systems. We used virtual network because of better scalability of deployed systems and also it gives us opportunity to automate this task in the future. The Wi-Fi part of the traffic was generated separately on different physical network adapters and then the final data of HTTP and Wi-Fi protocol had to be merged together.

In this paper we aim to simulate HTTP traffic, so that we create real network flows manually. Every single captured flow consists of one access request and one browsing request

of specific web page, and we have also slightly changed virtual environment between single experiments, for example maximum speed of the network, load of the virtual machine or error rate. Packets were captured directly on virtual network adapter by Wireshark [16] network sniffing tool and then they were saved as PCAP file. Each captured file has to be named by domain name of the corresponding web page and then saved to the database of relevant generator module.

V. HTTP SUBMODULE

This module represents the core part of our actual work. HTTP protocol was selected because of its complexity which allows us to better demonstration of achieved results. The main goal of this module is to generate valid atomic HTTP traffic based on input behavioral patterns. Behavioral patterns can specify the duration of communication and size of data traffic.This generated atomic HTTP communication has to be merged to final PCAP file based on information gathered from the calendar. The submodule can be divided into independent parts which are described as follows:

A. Randomizer

This part provides low level cooperation with database of stored PCAP files. It establishes a layer between stored files and generator engine which allows easy and transparent reading of stored packets. The randomizer reads packets from randomly selected file. The randomization is used in selection process of source PCAP file. The decision to change the source file is made by other modules; based on state of the generated communication. This approach preserves information about single flows in HTTP traffic i.e. if some query is processed, the context should not be switched.

B. Selector

Selector module is used to filter packets incoming from randomizer. Packet have to correspond to behavioral pattern and also the state of generated communication, so if the packet do not fit this rules it is dropped out until the best fitting packet is selected.

C. Header rewriter

Header rewriter is used to retransform packet data. Rewrite rules are applied on internet, transport and application layer of TCP/IP protocol stack. In this case all IP addresses in source incoming packets from randomizer are replaced based on generated communication parameters. This module is not strictly aimed for HTTP traffic but also resolves DNS queries and replaces all affected data. HTTP header is transformed as well, so all HTTP GET or POST messages are also modified.

D. Finalizer

The last step of workflow in HTTP submodule is finalizing generated packets. This part performs modifications of CRC checksums and timestamps of generated packets. This operation is crucial for preserving the validity of generated network traffic. The modification of timestamp allows the system to merge generated packets together and the stream of random valid network traffic for supported protocols is created.

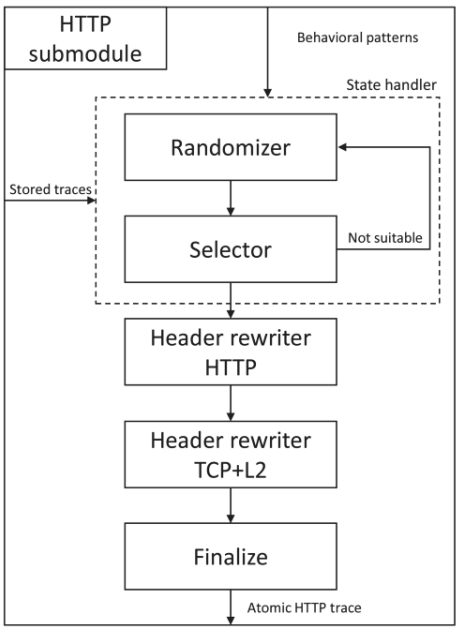

Fig. 2. HTTP submodule

VI. TEST AND RESULTS

The main goal of testing process is to prove that generated network traffic satisfies the key characteristics which are correspondence with behavioral pattern and validity of generated packets. We used two approaches to verify the communication is valid. The first approach is based on comparison of data distribution of packets in time, the second one is based on statistical analysis. We also manually verified generated PCAP files in Wireshark if they are valid and flows are not corrupted.

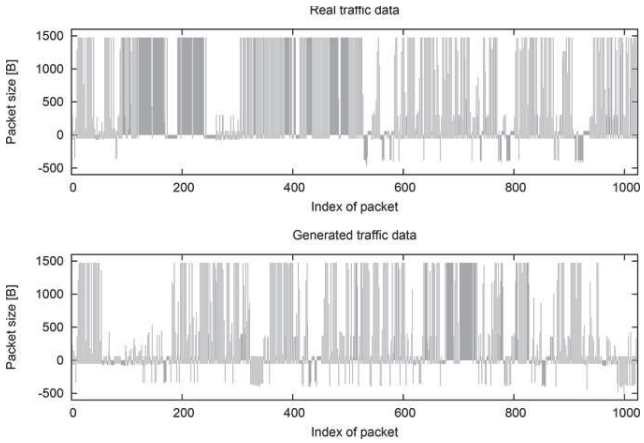

Fig. 3. Comparison of the real network traffic with generated traffic

The fig. 3 shows packet distribution of real captured data compared to generated traffic. Real data was captured by visiting single page of web application and on the other hand simulated traffic was generated with parameters (size of communication, length of communication) corresponding

to the real communication. The positive value on the axis y represents inbound communication and the negative value represents outbound communication from host machine. It is important that the amount of inbound and outbound communication and the time distribution is comparable with realistic traffic. The fig. 4 shows packet distribution of simulated traffic generated with the same input parameters. The distribution contains a small deviation between each traffic, but on the other hand, communications are still very similar.

The statistical analysis was based on computation of the average of packet size and time difference between consecutive packets. The results of tests are shown in table I. It contains the set of real captured packets and three traffic simulations which tend to follow the real traffic. In this case deviations from real communication are notable, but they still satisfy predefined requirements.

	Real traff.	Sim. traff.	Sim. traff.	Sim. traff.
Avg. in. size [B]	455.334	576.006	1134.556	657.991
Avg out. size [B]	105.846	114.800	94.283	118.375
Avg. time diff. [s]	0.0078	0.0033	0.0030	0.0034

TABLE I
STATISTICAL ANALYSIS

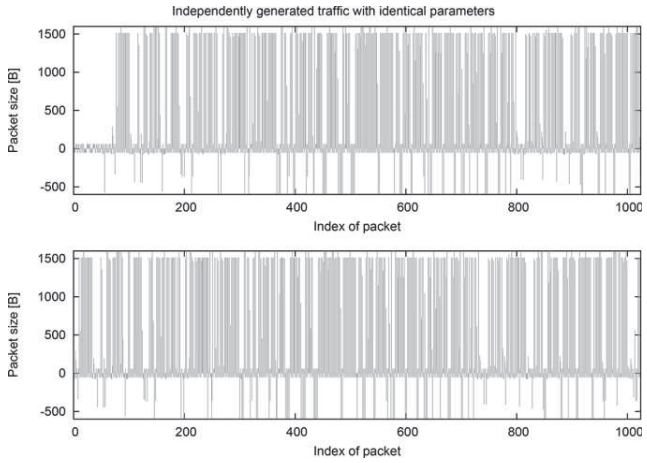

Fig. 4. Comparison of two generated traffic with the same behavioral parameters

VII. CONCLUSION

In this paper we presented the prototype of traffic generator, which generates network traffic based on predefined behavioral patterns, further we shortly described the architecture of proposed solution and results of generating HTTP communication. The generated traffic shows a similarity with genuine traffic, but each generated data has a small deviation from sample traffic,which is intentionally altered to increase diversity and randomness of generated traffic, but in a way to keep consistency of the communication.

We use this generator to test our behavioral reputation system. It can generate network traffic with specific behavioral pattern and then the reputation system detects or does not detect this pattern and evaluation of entity reputation can begin.

As a future work, we plan to develop new submodules for the generator core, such as SMTP, IMAP, SSH, in order to cover the most used network protocols to simulate real network traffic with high accuracy. This data will be used to verify applications such reputation systems for which is the valid network traffic very important part.

ACKNOWLEDGMENT

The work was supported by the EU/Czech IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070 and the internal BUT projects FIT-S-12-1 and FIT-S-14-2486.

REFERENCES

[1] C. V. Wright, C. Connelly, T. Braje, J. C. Rabek, L. M. Rossey, and R. K. Cunningham, "Generating client workloads and high-fidelity network traffic for controllable, repeatable experiments in computer security," in *Recent advances in intrusion detection*. Springer, 2010, pp. 218–237.

[2] C.-Y. Ku, Y.-D. Lin, Y.-C. Lai, P.-H. Li, and K.-J. Lin, "Real traffic replay over wlan with environment emulation," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. IEEE, 2012, pp. 2406–2411.

[3] C. Rolland, J. Ridoux, B. Baynat, and V. Borrel, "Using litgen, a realistic ip traffic model, to evaluate the impact of burstiness on performance," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 26.

[4] S. Molnár, P. Megyesi, and G. Szabo, "Multi-functional emulator for traffic analysis," in *Communications (ICC), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2397–2402.

[5] M. C. Weigle, P. Adurthi, F. Hernández-Campos, K. Jeffay, and F. D. Smith, "Tmix: a tool for generating realistic tcp application workloads in ns-2," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 3, pp. 65–76, 2006.

[6] S. Zander, D. Kennedy, and G. Armitage, "Kute–a high performance kernel-based udp traffic engine," *CAIA (Center for Advanced Internet Architectures) Technical Report*, 2005.

[7] J. Laine, S. Saaristo, and R. Prior, "Real-time udp data emitter (rude) and collector for rude (crude)," 2003.

[8] B. Adamson and S. Gallavan, "The multi-generator (mgen) toolset." [9] A. Botta, A. Dainotti, and A. Pescape, "Do you trust your software-based traffic generator?" *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 158–165, Sept 2010.

[10] G. Antichi, A. Di Pietro, D. Ficara, S. Giordano, G. Procissi, and F. Vitucci, "Bruno: A high performance traffic generator for network processor," in *Performance Evaluation of Computer and Telecommunication Systems, 2008. SPECTS 2008. International Symposium on*, June 2008, pp. 526–533.

[11] "Ieee standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks- specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Medium access control (mac) security enhancements," *IEEE Std 802.11i-2004*, pp. 1–175, 2004.

[12] M. Kacic, P. Hanacek, M. Henzl, and P. Jurnecka, "Malware injection in wireless networks," in *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on*, vol. 01, Sept 2013, pp. 483–487.

[13] S. Bhatia, D. Schmidt, G. Mohay, and A. Tickle, "A framework for generating realistic traffic for distributed denial- of-service attacks and flash events," *Computers & Security*, vol. 40, no. 0, pp. 95 – 107, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404813001673

[14] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

