jamk.fi

Kokkonen T., Hämäläinen T., Silokunnas M., Siltanen J., Zolotukhin M., Neijonen M. (2015). Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise. In S. Balandin, S. Andreev, Y. Koucheryavy (eds.) Internet of Things, Smart Spaces, and Next Generation Networks and Systems. ruSMART 2015. Lecture Notes in Computer Science, vol. 9247, 254-267.

DOI: https://doi.org/10.1007/978-3-319-23126-6_23

URL: https://link.springer.com/chapter/10.1007%2F978-3-319-23126-6_23

Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise

Tero Kokkonen[1,2], Timo Hämäläinen[2], Marko Silokunnas[1], Jarmo Siltanen[1], Mikhail Zolotukhin[2], Mikko Neijonen[1]

[1] JAMK University of Applied Sciences, Institute of Information Technology, Jyväskylä, Finland
{tero.kokkonen, marko.silokunnas, jarmo.siltanen, mikko.neijonen}@jamk.fi
[2] University of Jyväskylä, Department of Mathematical Information Technology Jyväskylä, Finland
{timo.t.hamalainen, mikhail.m.zolotukhin}@jyu.fi
tero.t.kokkonen@student.jyu.fi

Abstract. Because of the severe global security threat of malwares, vulnerabilities and attacks against networked systems cyber-security research, training and exercises are required for achieving cyber resilience of organizations. Especially requirement for organizing cyber security exercises has become more and more relevant for companies or government agencies. Cyber security research, training and exercise require closed Internet like environment and generated Internet traffic. JAMK University of Applied Sciences has built a closed Internet-like network called Realistic Global Cyber Environment (RGCE). The traffic generation software for the RGCE is introduced in this paper. This paper describes different approaches and use cases to Internet traffic generation. Specific software for traffic generation is created, to which no existing traffic generation solutions were suitable.

Keywords: Internet Traffic Generation, Cyber Security Research and Exercise, Cyber Security, Network Security

1 Introduction

The JAMK University of Applied Sciences has built a closed Internet-like network called Realistic Global Cyber Environment (RGCE). RGCE mimics the real Internet as closely as possible and contains most services found within the real Internet, from tier 1 Internet Service Providers (ISP) to small local ISPs and even individual home and corporate ISP clients. The fact that RGCE is completely isolated from the Internet allows RGCE to use accurate GeoIP information for all IP addresses within RGCE. This allows the creation of exercises or research cases where the attackers and the defenders are seemingly in different parts of the world and any device (real or virtual) will assume that it is actually operating within the real Internet. RGCE also contains various web services found in the real Internet [1, 2].

Due to the fact that RGCE is isolated from the real Internet, RGCE does not contain background user traffic of its own. This poses a problem: how can you realistically train for a scenario where your public services are being attacked by an unknown party, and the attack traffic is concealed within normal user traffic if there is no normal user traffic? This is the basic problem to be solved in order to efficiently use RGCE for cyber security exercises or research.

Traffic generation has an important role when characterizing behaviour of the Internet. Behaviour of the real Internet consists of the rapid changes of the network, network traffic and user behaviour as well as the variables of characterization vary from the traffic links and protocols to different users or applications [3]. In addition changing nature of connections in Internet is influenced by the behaviour of the users, which determines the page level, and the connection level correlation that should be included to the traffic generation models [4]. According to the study [4] this is neglected by the scientific literature.

There are two fundamental approaches to Internet-like traffic generation, trace-based generation and analytical model-based generation. In trace-based generation the content and the timings of the captured real traffic are retransmitted and in analytical model-based generation the traffic is generated based on the statistical models [5, 6].

Due to increasing amount of traffic, applications and users deep analysis of real Internet traffic is essential for planning and managing networks [7]. Deep analysis of real Internet traffic also gives an efficient viewpoint for realizing the extensive processes of the Internet [8]. Thus the deep analysis of the real network traffic can be used for developing Internet traffic generation software using realistic traffic patterns from both humans and machines.

In this paper the Internet traffic generation software is introduced. First, the requirements, existing solutions and different approaches for traffic generation are presented. Then the developed solution is introduced and evaluated.

2 Found Requirements

The main purpose of developed Internet traffic generation software is to generate user traffic for the cyber security exercises conducted within RGCE. To meet the requirements for cyber security exercises the Internet traffic generation software was implemented according to the following self-generated requirements:

- Centralized control; the system shall have a single point of control and the control mechanism shall enable the generation of a large volume of traffic with minimal user interaction.
- Ability to generate legitimate traffic; the generated traffic shall adhere to the generated protocol.
- Ability to generate meaningful traffic on several layers of the OSI model; the system shall be able to generate meaningful traffic on OSI layers 3-6. This shall include IP, TCP, HTTP and other application layer protocols.

- Ability to generate attack traffic; the system shall be able to generate traffic for various attacks commonly encountered on the Internet. Examples of such attacks include SYN flood, NTP and DNS amplification DDoS attacks.
- Generated traffic shall look like real Internet traffic; the traffic shall be as indistinguishable as possible from real traffic for both humans and machines.
- Ability to make the traffic look like it is coming from anywhere within RGCE; it shall be possible to deploy parts of the system to various parts of RGCE to make the geolocation information look realistic.
- Generated traffic shall not be a replay; replaying previously recorded traffic would make it easy to distinguish generated traffic from normal user traffic, unless the recorded captures are of significant length.
- Generated traffic shall work with existing servers; the system shall be able to use normal, non-modified servers as targets for traffic generation. A simple example would be HTTP: the system shall be able to generate legitimate non-identical requests to a given HTTP server, with varying HTTP headers and make those requests at human-like intervals.
- The system shall be highly autonomous; the system shall be able to recover from errors without human intervention as much as possible. The system shall be able to generate traffic without human intervention for extended periods of time.

3 Existing solutions for traffic generation

There are a number of proprietary and open source tools available for Internet-like traffic generation, such as TG Traffic Generator [9], NetSpec [10], Netperf [11], Packet Shell [12] and D-ITG [13, 14]. A detailed listing and analysis of available tools can be found from the study [5]. Those mentioned tools approach the problem from the viewpoint of workload generation through statistical models. Their goal is to generate repeatable workloads for networks and monitoring tools.

Such tools suffer from the fact that they are often implemented on top of non-real-time operating systems (OS). This causes their behaviour to be un-deterministic due to various scheduling decisions made by the OS as introduced in study [15]. Performance of D-ITG is also analysed in [14]. Netbed has a different viewpoint, it is a tool for integrating three experimental environments: network emulator, network simulator and real networks [16].

Developed Internet traffic generation software avoids many of above-mentioned problems, mainly because the goal is not in the generation of realistic workloads, but rather in meaningful payloads and good integration with existing off-the-shelf products with minimal customization.

4 Approaches

There are different approaches to Internet traffic generation with their pros and cons. These described approaches were analysed for the development of Internet traffic generation software.

4.1 Network layer traffic generation

Generating traffic on the network layer is a simple approach to traffic generation. It is trivial to implement using, for example, Linux raw sockets [17], and can be implemented for both IPv4 and IPv6.

This approach works by generating a large number of IP packets with randomized payloads. The use of Linux raw sockets also allows the source IP address of the packet to be spoofed, which allows a single machine to simulate a huge number of individual hosts. The machine sending the IP packets could be considered to be the default gateway for a large organization, such as university or a company.

An example system could work by requiring a definition of a range of source IP addresses to use (e.g. 10.0.0.1-10.0.0.255 for IPv4) and then generating a large number of IP packets with the source field set to one of the IP addresses within the source range.

The generated packets are only meaningful when analysed on the IP layer. If the requirements for the generated traffic are such that the traffic has to be meaningful on higher layers (e.g. TCP), this approach is not suitable without a considerable amount of effort. This means that implementing a custom TCP stack on top of Linux raw sockets is required. The only benefit over regular Linux TCP sockets [18] is the ability to spoof source IP addresses for individual TCP segments [19].

Various analytical model-based network traffic generation tools utilize this approach. Such tools put emphasis on IP traffic characteristics (e.g. packet size and timing), rather than the transmitted data itself [13, 9, 10].

This is not feasible for the purposes of Internet traffic generation within RGCE (see Section 2). But it is relevant for testing various other aspects of network performance.

4.2 Transport layer traffic generation

Using existing TCP stacks found in operating systems to handle the TCP connections significantly reduces the complexity of the implementation but makes IP spoofing [20] difficult. It is still possible to use a single machine to simulate a larger amount of hosts by using IP aliasing.

An elementary approach to traffic generation on the transport layer would be to utilize TCP stack provided by the operating system. This greatly simplifies the implementation of the traffic generator, as the OS TCP stack will take care of retransmission and other TCP details. As a downside, this approach does not allow for much control over the generated traffic characteristics.

As with the network layer approach (Section 4.1) this method works as long as meaningful exchanges on higher layers of the OSI model are not required (e.g. HTTP). It is possible to overcome this problem for the simplest of cases, such as creating multiple identical HTTP requests and always expecting an identical reply. More complex transmissions are also possible to implement but in most cases it would be more straightforward to just implement the approach described in Section 4.6

4.3 Replaying traffic

When considering approaches to Internet traffic generation, replaying PCAP files [21] is a rather natural option. Typically, traffic replay aims to generate repeatable workloads for systems under test [6]. This is achieved by replaying recorded data [22] or synthesizing [23] traffic traces and then replaying them through the network. Tcpreplay [22] is existing software solution that is able to replay captured TCP traffic from files.

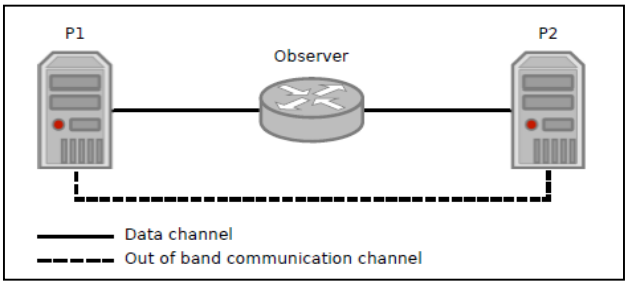

Fig. 1. Simple traffic replaying environment

It is necessary to use Out of band communication channel (Fig. 1) if the orchestration should not interfere with the system under test. Orchestration could include communicating the roles, timing, and bandwidth quotas associated with the replay [5].

In order to make this approach work, some processing is required for the PCAP file:

- Filtering out all unnecessary data streams. Unless the PCAP file is captured with the intention of replaying it, it is likely that the file contains a lot of unnecessary packets.
- Compiling the payload bytes from the TCP segments. Sending individual TCP segments from the PCAP file is not a feasible approach because network conditions are very likely to differ between the recording environment and the replaying environment. When the bytes are properly extracted and sent over the operating system's TCP stack the implementation does not have to concern itself with TCP details. It also makes it easier to detect and handle networking problems in the replaying environment.

- Constructing an intermediate presentation of the PCAP file that describes what to send and what to receive for each participant of the conversation.

It is worth noting that replaying PCAP files is only feasible for reliable transport layer protocols (e.g. TCP and SCTP). While it is possible to just extract the sent UDP (or other unreliable transport level protocols) packets and resend them, it will require extra steps to ensure that the packets get to their destination due to the nature of UDP [24]. There are two approaches to overcome this problem:

- Protocol awareness. The system needs to be aware of the protocol it is replaying and in case of lost packets mimic the simulated protocol's behaviour in such situations (if any). This requires considerable effort to duplicate the protocol's functionality and the solution starts to resemble the approach detailed in section 4.4.
- Out of band communication channel. An out of band communication could be utilized to transfer information about sent and received packets between participants. While this approach makes sure that all packets get delivered, it does not reliably reproduce the simulated protocol, because it is acceptable to lose packets in some UDP based protocols.

The following subsections will detail the out of band communication channel approach and its limitations. It is worth noting that the out of band communication channel must use a reliable transport layer protocol, such as TCP. The out of band communication channel also introduces additional latency to the replaying caused by TCP.

4.4 Replaying in a reliable network.

It is assumed in Fig. 2 that the replaying environment does not suffer from packet loss, thus introducing no unexpected side effects.

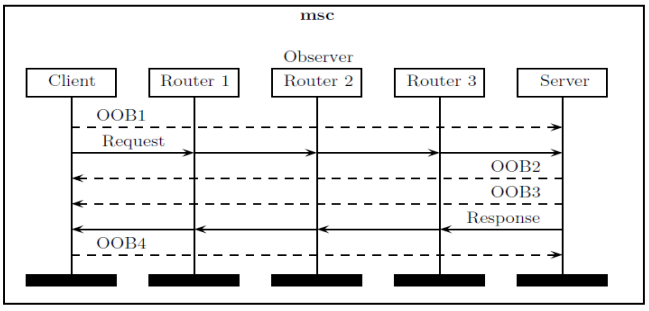

Fig. 2. No network problems

Replaying in a reliable network is processed as follows: client notifies Server through the OOB channel that it is about to send a request, client sends the actual request, server notifies client that it received the request, server notifies client that it is about

to send a response, server sends the response and finally client notifies server that it received the response successfully.

This scenario works as expected and does not introduce any additional side effects; the observer sees a single request and a single response and thus cannot tell the traffic apart from the real traffic.

4.5 Replaying in an unreliable network.

The fact that the out of band communication channel introduces some reliability features to the system can cause the observer to see responses without requests, this can be seen from Fig. 3.

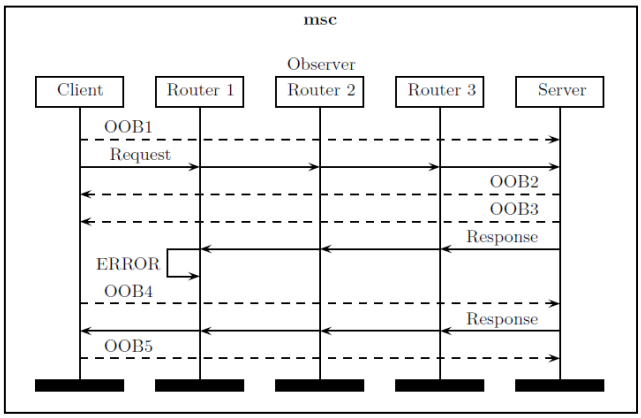

Fig. 3. Network problems

Replaying in an unreliable network is processed as follows: client notifies server through the OOB channel that it is about to send a request, client sends the actual request, server notifies client that it received the request, server notifies client that it is about to send a response, server sends the response, router 1 fails to deliver the packet, client notifies Server that it did not receive a response, server resends the response and Client notifies the server that it received the response successfully

In the case of a network failure the observer observes multiple identical responses without corresponding requests. This allows an observer familiar with the protocol in question, to conclude that this traffic is not authentic.

Even though replaying PCAP files is problematic for protocols that are implemented on top of unreliable transport protocols, it is still robust for protocols utilizing reliable transport protocols. But still this approach cannot be used with existing servers and will end up repeating the same conversation over and over again, thus not fulfilling the requirements listed in section 2.

4.6 Simulating clients

Simulating full clients for Internet traffic generation offers a flexible solution to traffic generation, as it allows fine-grained control over the generated traffic and the depth of simulation. This approach does not allow precise traffic generation on packet level or control of the various packet characteristics that are available in other traffic generation solutions, such as Inter Departure Time (IDT) and Packet Size (PS).

This approach can fulfil the requirements listed in section 2; it can be used with any server and the generated traffic is sufficiently diverse. It is also possible to implement very specific types of traffic (e.g. deliberately broken TCP traffic). If the client simulation is sufficiently sophisticated, it is very difficult for the server and observers to distinguish it from the realistic clients.

It is worth noting, that this approach requires a significant amount of effort, as it is difficult to create a single solution that could simulate multiple clients and protocols in a convincing manner. This means that the system will require multiple protocol specific modules.

5 Implemented solution

Implemented solution aims to generate traffic that looks meaningful to a human observer. It was decided to implement Internet traffic generation software using the full client simulation approach. Solution consists of a hierarchical network of nodes. The network forms a tree like structure. The network forms an opt-in botnet, where each individual node is a host.

5.1 Terminology

The network consists of three different node types: King, Slavemaster and Botmaster. Bots are not nodes (hosts), but are run on the same host as the Botmaster.

King is the root node of the tree. King acts as a bridge between the UI and the rest of the network. The UI is running on a webserver on this node. Both Slavemasters and Botmasters can connect to King. Every message sent into the network by the user passes through King.

Slavemaster connects to the King or another Slavemaster and acts as a router between nodes. Slavemasters can connect to other Slavemasters and thus the depth of the tree representing the network can be arbitrary. Slavemasters have full knowledge of the tree underneath themselves. When a Slavemaster receives a message it checks the message recipient. If the recipient is the Slavemaster it broadcasts that message to all of its children, who then broadcast it to their children and so on. If the recipient is one of the descendants of the Slavemaster message is forwarded towards it.

Botmaster is a leaf node of the tree. Botmasters are charged with performing the actual traffic generation. Botmasters run one or more Bots. Multiple Bots can be running simultaneously. Botmaster receives messages and status updates from its Bots and forwards them to King, which will then update the UI accordingly. In the current implementation Bots are ran in the same process as the Botmaster.

Bot handles the actual traffic generation. Each Bot is tasked with generating traffic of a certain type (e.g. HTTPBot generates HTTP traffic). If a Bot encounters an error it sends a notification to the UI about it.

5.2 Implementation

Current implementation of the system contains traffic generation profiles for various protocols and services, such as HTTP, SMTP, DNS, FTP, NTP, IRC, Telnet, SSH, CHARGEN and ICMP. Each protocol or service is capable of containing different profiles. For example there are five different bots for HTTP protocol: HTTPBot mimics an user that is browsing the internet, SlowlorisBot performs the slowloris HTTP DoS attack, SlowPOSTBot performs the slow POST HTTP DoS attack, HTTPAuthBot repeatedly attempts to authenticate using HTTP Basic Auth and HTTPDDoSBot repeats the same HTTP request continuously.

Implementation is done for GNU/Linux using the Go programming language [25]. Each node of the system (King, Slavemaster and Botmaster) has its own binaries. In the current implementation Bots are ran in the same process as the Botmaster, but this is required to change in the future development. Go was chosen as the implementation language due to the fact that it has native support for coroutines (called goroutines in Go) and easy interfacing with C programming language.

Go also provides a way to facilitate communication between goroutines using channels, which are derived from Hoare's CSP [26]. The first few versions of our traffic generation software utilized the C interface significantly, but the current version contains no C code. The need for interfacing with C reduced as the Go ecosystem grew and more libraries became available. Most of the C code in the early versions was related to utilizing raw sockets to conduct IP spoofing, which can now be achieved using Go.

Bots are run inside the Botmaster as goroutines. A Bot can contain multiple goroutines. Naturally, the Botmaster contains goroutines that are not Bots as well, such as goroutines related to communication with the rest of the network.

Bots communicate with the Botmaster using Go's channels. Botmasters, Slavemasters and King communicate between each other using a custom text based protocol implemented on top of TCP.

The UI is implemented as a single-page web application served by a web server running on the King. The UI communicates with the King using a custom protocol on top of WebSockets [27]. The King acts as a translator between the WebSocket protocol and the protocol used by the rest of the network. User interface (UI) of developed Internet traffic generation software can be seen in Fig. 4.

Fig. 4. UI of developed Internet traffic generation software

5.3 Evaluation

The evaluation for the developed solution is studied as follows. There were three different networks and three different amount of data generation Botmasters chosen. The different networks chosen were Localhost, Local Area Network (LAN) and Internet. The webserver with webpage including text pictures and links to 30 sub-pages was installed and HTTPBots browsed the contents of that webserver on each network cases. The LAN was in the JAMK University of Applied Sciences and the Internet scenario was between Netherlands and Finland (the webserver located in the Netherlands). The amount of Botmasters was 5, 25 and 125. The network data was captured on both sides, server side and client side. Time period of every single capture is 30 minutes long.

Evaluation data was generated using the HTTPBot, which mimics a browser by first downloading an HTML page from the targeted HTTP server. HTTPBot downloads all images, JavaScripts and CSS files referenced in the HTML document. Once all of the files are downloaded, the HTTPBot searches for a link to another page within the same domain or another domain. A link is chosen at random and the same process is repeated again.

The network traffic (PCAP data) was captured from the client and server side of the connection and also log data from the server was collected.

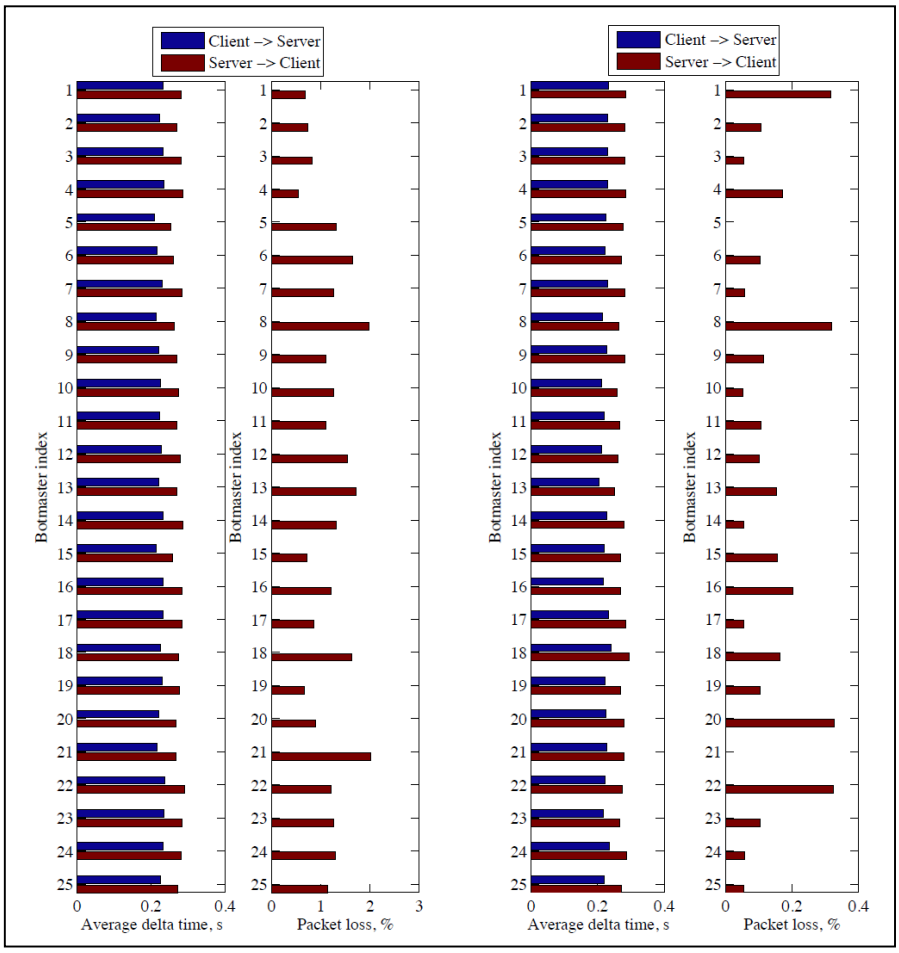

Fig. 5. Left: Internet 25 Botmasters, average delta time (s) and packet loss (%)
Right: LAN 25 Botmasters, average delta time (s) and packet loss (%)

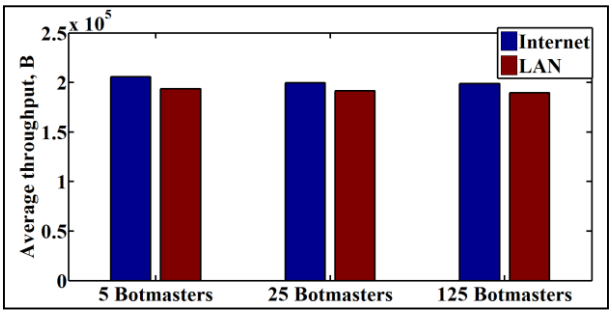

Fig. 6. Average throughput for generated traffic

Fig. 5 shows packet loss (%) and average delta time for 25 Botmasters in Internet and LAN. Average delta time in those figures is an average time difference between sent packets from the same source during the same conversation. Fig. 6 shows average throughput from all clients to server in all measured cases captured in client sides of the connections.

Evaluation shows that developed system is scalable and capable of producing significant amount of traffic. Scalability was proofed using different amount of Botmasters and different network topologies. In all tested network topologies the generated traffic behaves as expected based on the calculated characteristics.

Since Internet traffic generation software is designed for conducting research in network security and cyber attack detection, it is also capable to produce different sorts of attacks e.g. Denial of Service DoS/DDoS attacks based on volumetric traffic, resource exhausting or exploits and also bruteforce attacks.

Developed solution is also tested in the National Cyber Security Exercises organized by Finnish Defence Forces [28, 29]. Initial version of Internet traffic generation software is also being used for Internet traffic generation in an anomaly detection study [2].

All of those experiments show that developed traffic generation solution can be used to generate different kind of data patterns, and it is appropriate for different kind of cyber security analysis for example for big scale National cyber security exercises.

5.4 Lessons learned

The generation of the Internet traffic is extremely important for research and development of cyber security. For example research and development of Anomaly Detection algorithms or Intrusion Detection Systems requires an environment with realistic legitimate background traffic and design made attacks [30, 2]. Generation of Internet traffic has an important role in cyber security exercises and training.

There were some lessons learned from the use cases that caused extra development for the data generation software. OOB communication for control traffic is very important when generating lot of traffic (e.g. HTTPDDoSBot). Generated data might block the outgoing data and if the command communication data uses the same interface it is also blocked. That causes situation where one Bot blocks the Botmaster out from the network. Another lessons learned that required changes for development was CPU bound meaning that if there is Bot doing resource intensive processing it might harm the whole process and block the Botmaster out of communication. Bots that are CPU resource intensive (e.g. SYN flood Bot) cannot have permission to generate traffic as fast as they are capable of processing, thus there must be a limit e.g. 5000 packets/second/Bot.

6 Conclusion

In this study, approaches to realistic Internet traffic generation for cyber security research and exercise were considered. First requirements for traffic generation were

analysed. After that different solutions and approaches were described. Finally suitable approach was chosen and developed Internet traffic generation software was introduced. As a conclusion it can be said that the developed Internet traffic generation software met the requirements and it is suitable for modelling the Internet traffic as a part of the cyber security research and exercise. Requirements for next phase were found and in future the development of those requirements are planned to execute. The deployment of the several Botmasters shall be automated and replaying of PCAP data (limited only for TCP) between Botmasters shall also be developed.

References

1. JAMK University of Applied Sciences, Jyväskylä Security Technology (JYVSECTEC), Realistic Global Cyber Environment (RGCE), http://www.jyvsectec.fi/en/rgce/
2. M. Zolotukhin, T. Hämäläinen, T. Kokkonen, J. Siltanen, "Analysis of HTTP requests for anomaly detection of web attacks", 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, Aug. 2014, pp. 406–411
3. S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", IEEE/ACM Trans. Netw. 9.4, Aug. 2001, pp. 392–403
4. E. Casilari, F.J. Gonzblez, and F. Sandoval, "Modeling of HTTP traffic", Communications Letters, IEEE 5.6, June 2001, pp. 272–274
5. A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios", Computer Networks (Elsevier) 14.15, 2012, pp. 3531–3547
6. S. Hong and S.F. Wu, "On Interactive Internet Traffic Replay", Recent Advances in Intrusion Detection, 8th International Symposium, RAID 2005, Seattle, WA, USA, September 7-9, 2005
7. R. Pries, F. Wamser, D. Staehle, K. Heck and P. Tran-Gia, "On Traffic Characteristics of a Broadband Wireless Internet Access". Next Generation Internet Networks, 2009. NGI '09. July 2009, pp. 1–7
8. T. Li, J. Liu, Z. Lei and Y. Xie, "Characterizing Service Providers Traffic of Mobile Internet Services in Cellular Data Network", Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on. Vol. 1. Aug. 2013, pp. 134–139
9. The University of Southern California (USC-ISI), The Information Sciences Institute, TG Traffic Generation Tool, http://www.postel.org/tg/
10. The University of Kansas, The Information and Telecommunication Technology Center (ITTC), NetSpec Tool, http://www.ittc.ku.edu/netspec/
11. Netperf homepage http://www.netperf.org/netperf/
12. pksh -the Packet Shell http://tecsiel.it/pksh/index.html
13. Universita' degli Studi di Napoli "Federico II", D-ITG, Distributed Internet Traffic Generator http://traffic.comics.unina.it/software/ITG/
14. L. Angrisani, A. Botta, G. Miele and M. Vadursi, "An experimental characterization of the internal generation cycle of an open-source software traffic generator", Measurements and Networking Proceedings (M N), 2013 IEEE International Workshop on. Oct. 2013, pp. 74–78
15. A. Botta, A. Dainotti, and A. Pescapè. "Do You Trust Your Software-Based Traffic Generator", IEEE Communications Magazine (2010), pp. 158–165

16. B. White et al. “An Integrated Experimental Environment for Distributed Systems and Networks”, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002. USENIX Association. Dec. 2002
17. A. Kleen. Linux Programmer's Manual RAW(7). http://www.manpages.info/linux/raw.7.html
18. A. Kleen, N. Singhvi, and A. Kuznetsov’s. Linux Programmer's Manual TCP(7) http://www.manpages.info/linux/tcp.7.html
19. J. Postel, Transmission Control Protocol. RFC 793 (INTERNET STANDARD). Updated by RFCs 1122, 3168, 6093, 6528. Internet Engineering Task Force, Sept. 1981, http://www.ietf.org/rfc/rfc793.txt
20. M. Tanase, “IP Spoofing: An Introduction”, The Security Blog Mar. 2003, http://www.symantec.com/connect/articles/ip-spoofing-introduction
21. WireShark Wiki, Libpcap File Format http://wiki.wireshark.org/Development/LibpcapFileFormat/
22. Tcpreplay homepage, http://tcpreplay.synfin.net/
23. R. E. A Khayari, M. Rücker, A. Lehman and A. Musovic, “ParaSynTG: A Parameterized Synthetic Trace Generator for Representation of WWW Traffic”, SPECTS (2008), June 16-18 2008, pp. 317–323
24. J. Postel, User Datagram Protocol. RFC 768 (INTERNET STANDARD). Internet Engineering Task Force, Aug. 1980. http://www.ietf.org/rfc/rfc768.txt
25. The GO Programming Language homepage https://golang.org/
26. C. A. R. Hoare, “Communicating Sequential Processess”, Communications of the ACM, Volume 21 Issue 8, Aug. 1978, pp. 666-677
27. I. Fette and A. Melnikov, WebSocket Protocol. RFC 6455 (INTERNET STANDARD). Internet Engineering Task Force, Dec. 2011. http://www.ietf.org/rfc/rfc6455.txt
28. Ministry of defense press release 8.5.2013, “Cyber Security Exercise in Jyväskylä May 13-17 2013”, “kyberturvallisuusharjoitus Jyväskylässä 13.-17.5.2013”, http://www.defmin.fi/ajankohtaista/tiedotteet/2013/kyberturvallisuusharjoitus_jyvaskylassa_13.-17.5.2013.5502.news
29. Finnish Defence Forces Press Release 3.6.2014, ”Performance of Cyber Security is developed by co-operation between Government Authorities and University”, “Kybersuorituskykyä hiotaan viranomaisten ja korkeakoulujen yhteistyöllä”, http://www.fdf.fi/wcm/su+puolustusvoimat.fi/pv.fi+staattinen+sivusto+su/puolustusvoimat/tiedotteet/kybersuorituskykya+hiotaan+viranomaisten+ja+korkeakoulujen+yhteistyolla
30. S. Luo and G. A. Marin, “Realistic Internet traffic simulation through mixture modeling and a case study”, Proceedings of the 2005 Winter Simulation Conference, Dec. 4-7 2005, pp.2408–2416