Labelled network capture generation for anomaly detection

Maël Nogues
David Brosset
Hanan Hindy
Xavier Bellekens
Yvon Kermarrec

Labelled Network Capture Generation For Anomaly Detection

Maël Nogues[1], David Brosset[1,2], Hanan Hindy[3], Xavier Bellekens[4], and Yvon Kermarrec[1,5]

[1] Chair of Naval Cyber Defence[**], École Navale - CC 600, F29240 Brest Cedex 9, France
[2] Naval Academy Research Institute, École Navale - CC 600, F29240 Brest Cedex 9, France
[3] Division of Cyber-Security, Abertay University, Dundee, United-Kingdom
[4] Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, United-Kingdom
[5] Institut Mines-Télécom Atlantique, Lab-STICC CNRS UMR 6285, F-29238 Brest, France

Abstract. In the race to simplify man-machine interactions and maintenance processes, hardware is increasingly interconnected. With more connected devices than ever, in our homes and workplaces, the attack surface is increasing tremendously. To detect this growing flow of cyber-attacks, machine learning based intrusion detection systems are being deployed at an unprecedented pace. In turn, these require a constant feed of data to learn and differentiate normal traffic from abnormal traffic. Unfortunately, there is a lack of learning datasets available. In this paper, we present a software platform generating fully labelled datasets for data analysis and anomaly detection.

Keywords: Network Traffic Generation · Data Analysis · Intrusion Detection Systems · Cyber Security · Network Security

1 Introduction

As our lives are ever more connected, from our smartphones to our homes, the attack surface increases exponentially. These attacks generally influence the working state of the systems, often triggering their detection. To describe cyber-attacks, models defining the paths they take to reach their target through different networks are often used. However, to train and test these attack recognition models, real or simulated data is needed. The data, generally consisting of network capture, must be sufficiently documented and reflect the reality of network exchanges. Leading to the classification of well-formatted datasets with defined classes for each flow through the scenarios of these captures.

[**] Funded and supported by École Navale, IMT Atlantique, Naval Group, Thales and ENSTA Bretagne

Network captures available on the Internet are often provided without complementary information other than the context in which they were generated. Therefore, the exact content of these network captures is absent. Hence, users can only infer the classifications of the packets they contain.

Moreover, network captures are often transformed into CSV formatted datasets for intrusion detection systems. The most used dataset according to the taxonomy presented by Hindy *et al.* [4] is the KDD'99 dataset, generated for the information discovery and data mining tool competition [2] of KDD-CUP 1999. However, this dataset remains generic, does not contain IP addresses and lacks lot of key information defining an attack path and threat vectors. In addition, dating back to 1999, the dataset contains attacks that are outdated and not complex enough to reflect modern attacks. Al Tobi and Duncan also identified a number of errors in the dataset in [1]. The main concern raised in their manuscript is the dissimilarity between the number of attacks initially published by DARPA and KDD'99.

To this end, this paper presents a software platform to generate network traffic scenarios using a combination of real and/or virtual hosts. The platform automatically generates network capture in a reliable manner, with the property to identify and tag packets to the corresponding action.

The remainder of the paper is organized as follows; Section 2 presents a survey of network traffic generation tools and associated network captures. The platform to generate network captures is presented in Section 3. Section 4 provides a description and analysis of the results obtained and finally Section 5 concludes the paper and presents future work.

2 Background

This section describes related work as well as different captures and existing network traces available on the Internet.

2.1 Network capture generation

To the best knowledge of the authors, there is no software able to generate labelled packet captures based on specific scenarios. However, there exists network simulators and packet generators. Neither the simulators nor the packet generators are able to generate scenarios for the networks they emulate. Hence, this work is complementary to existing solutions and provides a large range of solutions to increase the flexibility and accuracy of dataset generation for machine learning based intrusion detection systems.

Three software for generating network traffic in different ways have been analysed, Swing [12], Sourcesonoff [11] and the network traffic generation tool of "*Realistic Global Cyber Environment*" [5]. In this subsection, their different features and network traffic generation are reviewed.

Swing Vishwanath *et al.* presented Swing, a network generator based on traffic captures [12]. Swing allows you to explore different options, recreating the capture through different parameters. Swing can be used to generate network captures for network capacity planning, broadband router design, queue management rule design, or bandwidth estimation. Despite the flexibility that allows exploring network parameters, scenarios are not explicitly defined.

Sourcesonoff This software is focused on accurate traffic generation and includes multiple protocols. The software is based on ON/OFF sources that allow the random generation of packets while following a well-defined distribution. It is developed in C and contains several different distributions to vary the generations of packets. The uniform distribution is based on the Linux functions *drand48()* and *random()*, with corrections to ensure perfect uniformity across all intervals. The Normal or Gaussian distribution is calculated using the Box-Muller transformation. Knuth's algorithm is used for Poisson distribution. Pareto, Weibull and Exponential distributions use a uniform distribution transformation. An additional pseudo-distribution is available: the Constant distribution. In this case, all generated values are equal to a user-defined constant. This method allows the user to generate more predictive behaviour. Factor multiplication allows users to convert randomly generated values into bytes and nanoseconds. Distributions can be limited by user-defined minimum and maximum values. Once converted, the data is used for communication in the network. The data flows can be parameterised by the user to refine the behaviour of the software for its use. Different sets of sources can work at the same time to produce more traffic consistent with the traffic seen on the Internet. Each set of sources will then be associated with its own distribution and parameters to allow greater control over the data generated to the user.

Traffic Generation for the *Realistic Global Cyber Environment* The traffic generation tool that has been developed for the *Realistic Global Cyber Environment* (RGCE) is an Internet traffic simulation tool. It allows control of the traffic generation from a central point of the network while generating packets on different machines through this same network. This tool allows the generation of tailored traffic. RGCE use a complete simulation approach of network clients consisting of a hierarchical network of nodes forming a tree in which each node represents a client on the network.

The nodes are divided into several levels, each of which has a different role. The root node is the *King*, serving as a bridge between the network and the user interface of the tool. The user interface works through a web server.

The *Slavemaster* nodes represent network routers. They know the entire subnet and can be chained to create a tree of arbitrary depth. When they receive a message from the root node, if the message is destined for the *Slavemaster* it is then broadcast to all its children nodes, however, if the message is destined to one of its sub-nodes, then it sends the message in the direction of that sub-node.

The *Botmaster* nodes are the leaves of the tree. They represent the host machines of the system and can execute one or more *Bots* allowing to generate traffic on the network. *Botmasters* receive messages from the network traffic generation *Bots* and their status, passing them to the root node (*King*) to update the user interface. The *Bots*, meanwhile, are responsible for generating network traffic. Each *Bot* is responsible for generating a particular type of traffic (e.g., *HTTPBot* generates HTTP traffic). If a *Bot* encounters an error, it sends a notification to its *Botmaster* to send to the *King* to update the user interface.

2.2 Online Network Captures

Network captures can be found online, mostly provided as raw captures or processed such as *KDD'99*. However, network captures almost never come with well-defined protocols allowing users to understand the scenarios at hand and the action taken by the generators. This subsection explores two processed packet capture, the *KDD'99* dataset and the *UNSW-NB15* dataset. For the purpose of this manuscript, all network captures that does not come with a scenario describing its generation has an equivalent value. It seems then reasonable to limit ourselves to the most commonly used.

KDD'99 KDD'99 is a dataset provided by the University of California Irvine (UCI) for the third international knowledge discovery contest and data mining tools. The goal of the competition was to build a network intrusion detection system, based on a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" connections, called normal. The dataset contains a standard set of data to audit, including a number of simulated intrusions in a military network environment.

A list of the different attacks contained in this dataset is presented in table 1.

Type	dos	u2r	r2l	probe
Attacks	back land neptune pod smurf teardrop	buffer_overflow loadmodule perl rootkit	ftp_write guess_passwd imap multihop phf spy warezclient warezmaster	ipsweep nmap portsweep satan

Table 1. Attack list of the dataset from *KDD'99*

This dataset contains attacks of several different types that are:

– Denial of Service (DoS) Attack: An attack in which the attacker makes a computing resource or memory busy or full disabling him from handling legitimate requests, or denying legitimate users access to a machine.

– Privilege Elevation Attack (U2R): is an exploit class in which the attacker first accesses a normal user account on the system (e.g. through a dictionary attack, or social engineering) and exploits one or more vulnerabilities of the system to gain privileged access (root) on the system.
– Local Network Infiltration Attack (R2L): Occurs when an attacker has the ability to send packets to a machine on the network, but does not have an account on that machine, exploiting a vulnerability to obtain a local access as a user of a machine.
– Network Discovery Attack (Probe): is an attempt to collect information on a computer network for the purpose of circumventing security checks.

As aforementioned, KDD'99 no longer represents the current scene of cyber attacks found on the Internet. It remains , however, one of the most used datasets for detection of intrusions [4].

The features that have been chosen for *KDD'99* are presented in the table 2 [1]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	root_shell	su_attempted	num_root	num_file_creations	num_shells	num_access_files	num_outbound_cmds	is_host_login	is_guest_login	count	srv_count	serror_rate	srv_serror_rate	rerror_rate	srv_rerror_rate	same_srv_rate	diff_srv_rate	srv_diff_host_rate	dst_host_count	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_rerror_rate	dst_host_srv_rerror_rate
Basic features (Total : 9)									**Content features (Total : 13)**													**Time-based features (Total : 9)**									**Connection-based features (Total : 10)**									

Table 2. Features of *KDD'99* [1]

The *KDD'99* dataset is a transformation of a network trace generated by DARPA in 1998. This transformation was performed using the intrusion detection system *Bro* and deletes 5 essential information[6] to be able to link the records obtained from their sources in the trace of DARPA. In this transformation, ICMP packets are treated differently than other packets. Indeed, each ICMP packet is treated as an entire connection (so-called *stateless*) whereas the UDP and TCP connections consist of a sequence of packets exchanged between 2 machines (called *stateful*).

Many of *KDD'99*'s critics come from the network trace from which it was made, DARPA. Indeed, this network trace has a large number of attacks that are detectable using only the *time to live* (TTL) field in the headers of the packets. This over-representation of an attack characteristic can cause intrusion detection patterns, trained with this dataset, to increase detection bias.

[6] the start time of the connection, the source IP address, the source port, the destination IP address and the destination port

According to A. M. Al Tobi and I. Duncan [1] the processing of *KDD'99* has introduced additional errors. For example, the resulting dataset contains more TCP traffic, less ICMP traffic, and less UDP traffic than the *DARPA* network trace from which it is made.

To solve some of *KDD'99* issues, M. Tavallaee et. al. present a new dataset based on *KDD'99*, called *NSL-KDD* [10], in their article [9].

UNSW-NB15 The Australian Centre for Cyber Security (ACCS) has generated a new dataset named *UNSW-NB15* presented in [6], simulating a modern environment allowing to generate network traffic resembling a modern traffic and containing a large variety of recent attacks, especially low-profile intrusions that may be difficult to detect and that do not exist in the *KDD'99* dataset.

The *UNSW-NB15* dataset was created using the *IXIA PerfectStormOne* tool, owned by the ACCS Cyber Range, using CVE services to retrieve information about the latest types of attacks used and discovered in cyberspace. CVE services are a kind of public dictionary of security vulnerabilities disclosed on various systems and applications that are accessible via the Internet. With this tool, researchers were able to generate more than 100 GB of data usable by intrusion detection systems. This data was retrieved by the network trace capture tool *tcpdump* deployed on one of the routers used in the network exchanges generation.

The configuration of the IXIA PerfectStorm tool for generating the *UNSW-NB15* dataset is detailed in figure 1.

For this dataset, the tool virtualises three servers. Servers 1 and 3 are set to generate basic network traffic, while Server 2 generates malicious and abnormal activity in the network. The servers are connected to the user machines via two routers. These routers are connected to a firewall that is configured to pass traffic from one router to another, whether normal or abnormal. Then they recovered the traffic generated by the *IXIA PerfectStormOne* tool thanks to *tcpdump*, installed on router 1, to save the data in an exploitable format. Finally, the *IXIA PerfectStormOne* tool is used here to generate and corrupt normal traffic with recent attacks thanks to its link with online CVE services.

This dataset thus proposes nine different types of attacks, representing the different methods of attacks existing in the cyberspace.

A study was conducted to evaluate the performance of the dataset *UNSW-NB15* [7]. In this study, the dataset was divided into two parts, one for learning and the other for testing classifications. These two parts were evaluated statistically, on the correlation of characteristics and finally on the complexity of the dataset. The results were then compared to those of the *KDD'99* dataset, particularly in terms of accuracy complexity and false alarm rate. The results of the evaluation show that the *KDD'99* dataset shows better results in false alarm rates. This is due to the lack of diversity of attacks and the over-representation of certain data characteristics in these attacks. In conclusion, the *UNSW-NB15* dataset is more complex than the *KDD'99* dataset because of the different types of modern attacks it uses.

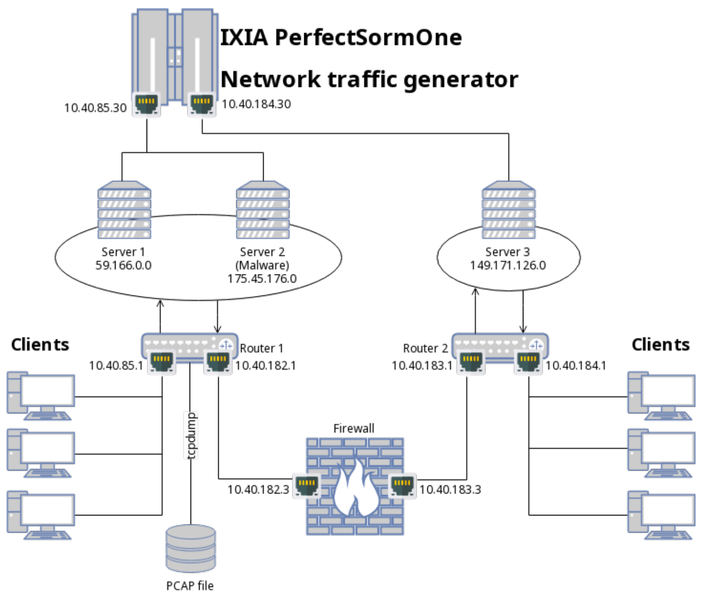

Fig. 1. Configuration of the *IXIA PerfectStormOne* tool for the dataset generation (inspired by [6])

Moustafa and Slay have thus been able to generate a synthetic network dataset, modern enough to answer current network analysis issues as shown by Moustafa et al. in the article [8], thanks to the tool *IXIA PerfectStromOne* using these nine types of attacks and by translating the network traces obtained in 49 characteristics that were extracted using tools of network activity audits such as *Argus* and intrusion detection systems such as *Bro*. They have shown that the *KDD'99* and *NSL-KDD* datasets do not represent modern network traffic that contains modern, low-impact attacks on network traces.

3 Network captures generation platform

The goal of the platform presented within this manuscript is to allow the creation of replicable network captures based on pre-defined scenarios, from which one can easily extract sets of training data for IDS or for research.

For now this platform is still a proof of concept that aims to show that it is possible to create training datasets that are already labelled, for IDS and research, almost completely automatically. And that the generated datasets are easily reproducible thanks to their adjunct scenario file.

Figure 2 presents a schema of the operating principle of the platform.

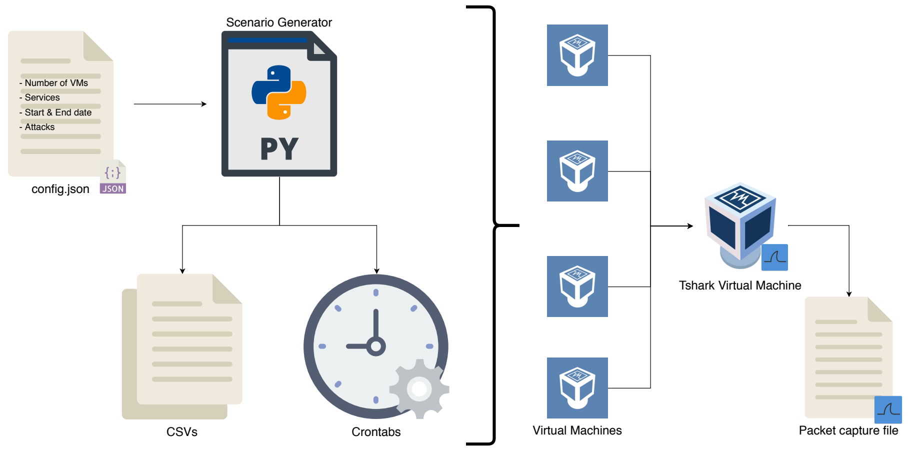

Fig. 2. Creations of the packet captures through the platform

3.1 Design and Implementation

This section describes the operating principles of the network capture generation platform from a software design perspective.

Description The platform is based on a python implementation of the model developed, abstracting the concepts to be represented in the network (i.e., the virtual machines, the actions they perform, their services, the host machines, and their behaviours).

This platform allows to randomly generate scenarios of normal uses of a network using parameters specified by the user. Making it possible to create experiments with their protocols very simply. Once generated, the scenarios are transposed to the real and virtual hosts for the deployment and dataset generation phase. The deployed hosts are configured to be assigned to a crontab generated by the algorithm adding it to the list of devices to be listened to for the network capture.

After the capture, the platform transforms the PCAP file to a CSV before adding the labels to the newly generated dataset. The labels are added automatically thanks to the crontab files generated by the platform that can be correlated to the packets in the CSV file. This automatic labelling of the generated dataset is the main contribution of this manuscript as it allows for the production of ready to use datasets for data analysis or data mining without the hassle of labelling datasets manually. These datasets could also be used for training anomaly detection models.

Modelling of the program For the network capture generation platform, the network model used in this platform presents networks as a number of machines (virtualised in this implementation) and VLANs that allows for the creation of a complex network.

The virtual machines were modelled with their IP and MAC addresses, the enabled services, their behaviours (to simulate a user's choice of actions), and the actions they will perform during the generation. The services are identified with a name and relate to commands.

The actions are made from the commands available in the services, these commands have a list of different parameters in which the actions choose the final command to execute, the actions also have a time stamp defining when the command will be executed during the experiment.

Special virtual machines are also defined in the model, called "hacker". These virtual machines are different from the others as no services are executed, however, a list of commands with parameters are used to attack the legitimate virtual machines in the network.

Generating the scenarios The program uses a configuration file in JSON format. Listing 1.1 provides an example of the configuration, to define the services available for virtual machines, the number of virtual machines per configuration group, the number of configuration group changes, the prefix of the IP addresses to be used (e.g. "192.168.10."), the start date of the experiment, the end date of the experiment, the maximum number of actions per machine virtual, and attack commands available to the attacker.

By varying these parameters, the behaviour of the machines can be modified on the simulated network and thus the network traces that are recovered during the experiments.

```
{
  "network": {
    "number_of_vms": 2,
    "number_of_changes": 1,
    "prefixe": "192.168.10.",
    "services": [{
      "name": "sshd",
      "commands": [{
        "name": "ssh",
        "parameters": ["tester@&ip"]
      }]
    }]
  },
  "experiment": {
    "start_date": "2018-08-02 13:00",
    "end_date": "2018-08-02 17:00",
    "max_actions_per_vm": 500
  },
  "hacker": {
    "attacks": [{
      "name": "nmap",
      "parameters": ["-T0 -sV --top-ports 5 &ip"]
      }]
   }
}
```

Listing 1.1. Configuration file example

To generate random behaviours of virtual machines, a list of virtual machines are generated (with their IP and MAC addresses, and their available services). From this list, configuration groups are created. Hosts in a configuration group are available at the same time during the experiment. The configuration group further allows defining the interactions between the machines, randomly choose the services to contact and the time of contact.

The definition of the configuration group is essential to the working of the platform. A misconfiguration in the file would essentially render the trace ineffective, hence, limitations were implemented to ensure appropriate hosts are contacted at appropriate times.

3.2 Generating the attacks

This section describes the existing attack tools and algorithms. The attacks used by the attacker during the experiments are also presented in this section.

Attacks To test the resilience of networks, a large number of tools exist. Nmap is undoubtedly the best-known tool for network discovery. Nmap has many possibilities to identify enabled services on a computer and identify open ports. Combined with the Nmap Scripting Engine (NSE), many vulnerabilities can be exploited directly by Nmap.

Other tools such as Metasploit were also used in this study. Metasploit is a framework for the development and execution of exploits and can execute attacks against remote machines. Metasploit features a commercial version, as well as, various other projects related to the framework. Among the most important, the Opcode database and The shellcode archive.

The platform also uses OpenVAS, which is a framework of different services and vulnerability scanning tools focusing on effective vulnerability management. The framework is used to execute a set of scheduled attacks on one or more machines and draw a detailed report of the vulnerabilities of the machines tested.

Malicious User The generation of cyber attacks, in the network capture, is achieved through the integration of a virtual machine executing a set of attacks on the simulated network. The attacker host has a pre-defined IP and MAC address and is generated last.

The attacker's behaviour uses predefined commands from a list of attacks provided in the configuration file. These attacks are executed by the attacker's virtual machine, targeting randomly selected virtual machines from the list of all the virtual machines on the network.

The commands available for the attacker are those included by default on the Kali Linux distribution that is used as a base operating system. In particular, the commands used by the attacker are defined in the *config.json* configuration file.

4 Experiments

To test the network experiment generation platform, 3 experiments were conducted with different attacker behaviors:

- An attacker scans the entire network on all ports.
- An attacker scans the network looking for an SSH server and then connects to it by forcing the password of the root user.
- A very cautious attacker uses long interval settings to scan without raising suspicion of intrusion detection systems.

Several categorizations of attackers exist in the scientific literature. The taxonomy, proposed by SLN Hald and JM Pedersen, presents 8 different types of attackers namely; Script Kiddies, Cyber-Punks, Insiders, Petty thieves, Gray Hats, Professional Criminals, Hacktivists and Nation states [3]. According to this taxonomy, the first attacker of the list is the equivalent of a Script Kiddie, the second attacker is the equivalent of a Petty thieve and the last striker is seen as a Professional criminal.

A Windows 10 version 1803 machine with an i7-4720HQ processor and 20GB of RAM was used to host these experiments. As this platform is a proof of concept, the test network was kept simple with 1 network in which 5 virtual machines, one being the attacker, were communicating.

Virtual machines in the test network are Arch Linux virtual machines, up to date at the time of the experiments. They include SSH, FTP, and HTTP services through the openssh, inetutils, and apache packages. The machine listening to the network is an Arch Linux virtual machine that is also up to date and contains the wireshark-cli package to use the tshark command to retrieve the packets transmitted over the network. Finally, the attacking machine is the virtual machine provided by the Kali Linux website for the VirtualBox platform.

The network capture files contains 22 different IP addresses including the 16 IP addresses belonging to the virtual machines that make up the observed network, the IP address of the machine running the network capture, the IP address of the machine running the attacks on the network, the 2 broadcast IP addresses (255.255.255.255/24 and 0.0.0.0/24) and 2 operating IP addresses (224.0.0.252/24 and 224.0.0.22/24).

The most used ports by the network machines are ports 21, 22 and 80 corresponding to the ports used by the services chosen to use for this experiment (FTP, SSH and Apache). Port 5355 is also used as it corresponds to the LLMNR protocol allowing for name resolution on the local network.

4.1 Experiment 1: Script Kiddie

During the experiment with the attacker repeatedly scanning the entire network, a total of 1,724,531 packets were collected over a period of 4 hours. The average size of these packets is 69 bytes. Such a low average packet size is due to the network scan as discovery packets are fairly small.

A total of 790 034 packets exchanged between the hacker and other machines (IP address 192.168.10.48/24). It, therefore, represents more than 48% of the packet exchanges of the network capture.

Figure 3 presents the normal traffic compared to the attacker's traffic.

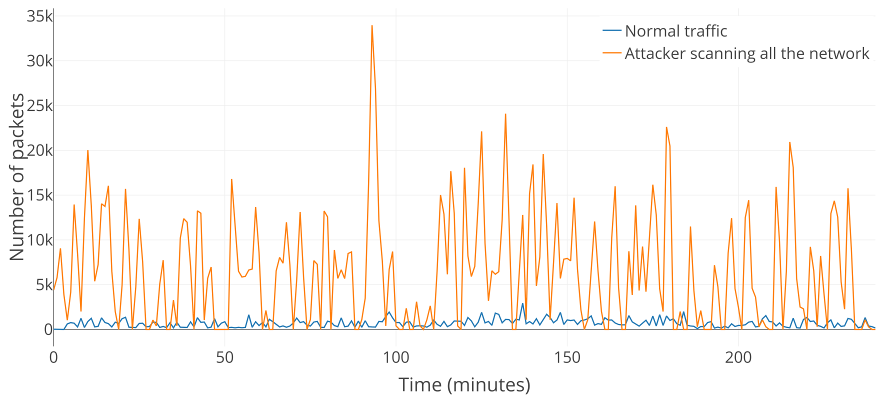

Fig. 3. Number of packages per minute over the duration of the experiments

4.2 Experiment 2: Petty thief

During the experiment with the attacker repeatedly scanning the network for a machine with port 22 open, a total of 514,308 packets were collected over a period of 4 hours. The average size of the packets obtained during this experiment is of 123 bytes.

The attacking machine did numerous recognition through Nmap. It is the one that communicates the most within the network trace with a total of 146 936 packets exchanged (IP address 192.168.10.210/24). It, therefore, represents more than 34% of the packet exchanges of the network capture.

The attacker, while focusing on a particular service, participates less in the network trace as it only represents 34% of the trace whereas in the previous experience it represented 48% .

This attacker could also represent a beginner using scripts, but with a specific goal that is to connect in ssh on machines whose port 22 is open. Figure 4 presents the normal traffic compared to the attacker's traffic.

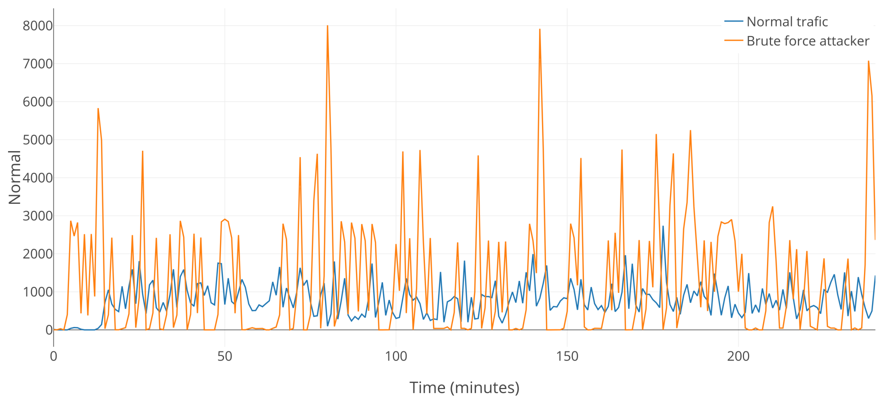

Fig. 4. Number of packages per minute over the duration of the experiments

4.3 Experiment 3: Professional criminal

During the experiment with the attacker scanning the network very cautiously, a total of 123,626 packets were collected over a period of 4 hours. The packets obtained by this experiment have an average size of 99 bytes.

The attacking machine having made very cautiously its recognition with the command Nmap and the good parameters of time, it is the one communicating least in the network trace with a total of 832 packets exchanged (IP address 192.168.10.147/24). It, therefore, represents less than 1% of packet exchanges of the network capture.

When these results are compared with the previous ones, it is noticed that the cautious attacker is much quieter than the others. This attacker could go unnoticed if it wasn't known that he was in the network trying to discover the computers that are accessible. Indeed, with a participation of 1% in the network trace, this attack could appear as a machine little talkative to an external observer. He therefore represents an attacker who knows exactly what he is doing and who is very organized. Figure 5 presents the normal traffic compared to the attacker's traffic.

4.4 Results analysis

To fully understand the impact of the different behaviors of attackers on the generated data, Figure 6 explores the difference in the number of packets exchanged per minute at the same time of the different experiments.

It can be pointed out that the attacker scanning the entire network generates a lot more packets than the other two attackers. Packets are grouped into peaks of activity, synonymous with wild recognition and noisy attack.

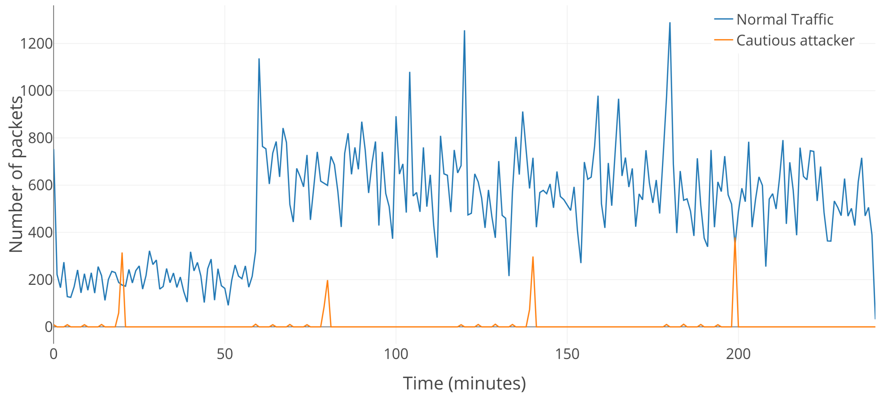

Fig. 5. Number of packages per minute over the duration of the experiments

Attacker scanning cautiously does not change the packet line of its network trace, remaining relatively flat. This shows that this attack technique is rather difficult to spot in a network trace because it does not stand out from the rest of the supposedly normal traffic.

The different experiments performed show that the network traffic generation platform allows the creation of specific attacker's comportment which can be used to generate any kind of specific threats to get datasets representing them.

This approach of the generation allows for an easy automation of the transformation process, from network capture files to a readable format for any machine learning algorithm, as it is possible to relate all packets from the network capture files to its sources from the files containing the scenarios, generated by the platform.

5 Conclusion

The results obtained show that the proposed solution is able to generate pertinent network datasets. Traffic is clearly identified, unlike many datasets available. The tool developed makes it easy and quick to configure complex scenarios. The traces produced by the tool are also realistic as they are generated from fully fleshed systems.

The modelling of the types of attackers is achieved through relatively simple actions. The randomness of the scenario generation makes the attacker's actions disparate and could therefore affects the quality of the generated network traces, however, several differences can be observed in the network traces when the behaviour of the attacker is changed making the solution modular and flexible.

Future work include the move from a purely random behaviour to a simulated behaviour, managed by a multi-agent system, for example. the network

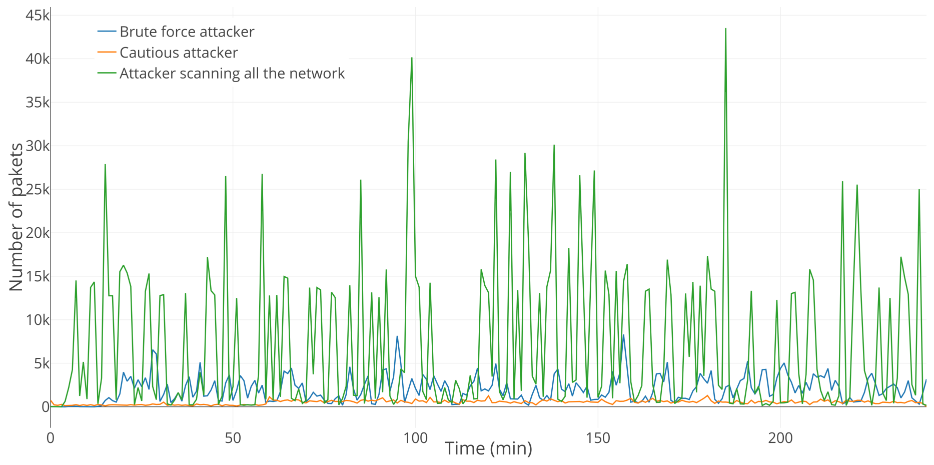

Fig. 6. Number of packages per minute over the duration of the experiments

trace generation platform could also be improved by integrating communication functions with other machines and virtual machine management functions to simplify the use of the virtual machine solution, enabling load balancing on several machines of the different virtual machines as well as manage the configuration of the different virtual machines necessary for the experiment requested automatically.

References

1. Al Tobi, A.M., Duncan, I.: KDD 1999 generation faults: a review and analysis. Journal of Cyber Security Technology pp. 1–37 (Sep 2018). https://doi.org/10.1080/23742917.2018.1518061, `https://www.tandfonline.com/doi/full/10.1080/23742917.2018.1518061`
2. Bay, S.D., Hettich, S.: UCI KDD Cup 1999. University of California, Irvine, School of Information and Computer Sciences (1999), `https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/kddcup99.html`
3. Hald, S.L.N., Pedersen, J.M.: An Updated Taxonomy for Characterizing Hackers According to Their Threat Properties. In: 2012 14th International Conference on Advanced Communication Technology. pp. 81–86. IEEE (2012)
4. Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., Bellekens, X.: A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets. arXiv:1806.03517 [cs] (Jun 2018), `http://arxiv.org/abs/1806.03517`, arXiv: 1806.03517
5. Kokkonen, T., Hämäläinen, T., Silokunnas, M., Siltanen, J., Zolotukhin, M., Neijonen, M.: Analysis of Approaches to Internet Traffic Generation for Cyber Security Research and Exercise. In: Balandin, S., Andreev, S., Koucheryavy, Y. (eds.) Internet of Things, Smart Spaces, and Next Generation Networks and Systems, vol. 9247, pp. 254–267. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-23126-$6_2$3, `http://link.springer.com/10.1007/978-3-319-23126-6_23`

6. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS). pp. 1–6. IEEE, Canberra, Australia (Nov 2015). https://doi.org/10/gf234f, `http://ieeexplore.ieee.org/document/7348942/`
7. Moustafa, N., Slay, J.: The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Information Security Journal: A Global Perspective **25**(1-3), 18–31 (Apr 2016). https://doi.org/10/gfz6v4, `http://www.tandfonline.com/doi/full/10.1080/19393555.2015.1125974`
8. Moustafa, N., Slay, J., Creech, G.: Novel Geometric Area Analysis Technique for Anomaly Detection using Trapezoidal Area Estimation on Large-Scale Networks. IEEE Transactions on Big Data pp. 1–1 (2017). https://doi.org/10/gf234h, `http://ieeexplore.ieee.org/document/7948715/`
9. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6. IEEE, Ottawa, ON, Canada (Jul 2009). https://doi.org/10/c9qtzd, `http://ieeexplore.ieee.org/document/5356528/`
10. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB (Jul 2009), `https://www.unb.ca/cic/datasets/nsl.html`
11. Varet, A., Larrieu, N.: How to generate realistic network traffic ? In: IEEE COMPSAC 2014, 38th Annual International Computers, Software & Applications Conference. p. pp xxxx. Väster\a as, Sweden (2014), `https://hal-enac.archives-ouvertes.fr/hal-00973913`
12. Vishwanath, K., Vahdat, A.: Swing: Realistic and Responsive Network Traffic Generation. IEEE/ACM Transactions on Networking **17**(3), 712–725 (Jun 2009). https://doi.org/10.1109/TNET.2009.2020830, `http://ieeexplore.ieee.org/document/4914755/`