RAPPORT D'AVANCEMENT POUR PAIEMENT D'ACOMPTE SUR CONVENTION DE SUBVENTION

Références

Accord général de partenariat pôle de recherche du Pôle d'Excellence Cyber, référence n° 14 81 000217

Accord spécifique (convention de subvention) n° 17810092 notifié le 29/11/2017 désignée T0

Raison sociale et adresse du bénéficiaire (UMR ou équipe d'accueil) :

Direction de thèse (Prénom et Nom) : Christophe Bidan

Post-doctorant (Prénom et Nom) : Mouad Lemoudden

Objet de l'accord spécifique (libellé du sujet) : Evaluation des IDS : Critères d'évaluation, mesures et plateformes

Jalon correspondant à l'état d'avancement du projet : T0 + 24 mois

Rapport d'avancement (format libre)

Evaluation of Security Monitoring

1. Introduction

With exponential growth in the size of computer networks and developed applications, the significant increase of the potential damage that can be caused is becoming prevalent. Intrusion detection is a common cyber security mechanism whose task is to detect malicious activities in host and/or network environments. Given the importance of intrusion detection, the research and industrial communities have designed and developed a variety of intrusion detection systems (IDSes). Due to the lack of adequate dataset, in addition to the fundamental shortcomings of this method, anomaly-based approaches in intrusion detection systems lack a strong evaluation methodology.

The goal of our work is to ultimately come up with an evolving platform for IDS evaluation that solves many of the issues that exist in the state of the art methods. We begin our work by surveying the relevant scientific literature. Traditionally, IDSes are evaluated based on their detection ability against a labeled dataset that contains normal and abnormal network traffic. Upon inspection, it is clear that datasets publicly available are usually obsolete in the span of a couple years in both anomaly types and background, benign Internet traffic. They also suffer from a lack of volume and diversity in traffic, and ultimately, lack of representativeness and realism.

Understanding the different aspects that go into consideration in creating a satisfactory IDS evaluation methodology has helped us in laying out design principles that enable us to remedy a bulk of the drawbacks existing in the state of the art. Consequently, we have designed a platform proposal to introduce a novel way to carry out IDS evaluation objectives. This work offers a generic overview of our proposal, detailing the different major components, and recounts the steps taken to implement our proposal, discussing the challenges and prospects.

This work will discuss related works that include IDS datasets as well as the different evaluation metrics in section 2. Section 3 gives a generic overview of our proposed platform. Section 4 focuses on normal and malicious traffic generation methods and tools. Section 5 details the different components for describing and automating our platform. Section 6 describes our proof of concept experiment. Section 7 discusses future works and perspectives. Finally, section 8 offers a conclusion of our work.

2. Related Work

2.1. IDS datasets

In this section, we discuss and critique the most important publicly available IDS datasets since 1998 to demonstrate their goals, merits, and issues.

DARPA (Lincoln Laboratory 1998-99):

In 1998 and again in 1999, the Lincoln Laboratory of MIT conducted comparative evaluations of intrusion detection systems (IDSs) developed under DARPA funding [1]. These evaluations contributed significantly to the intrusion detection research field by providing direction for research efforts. Offline datasets were made available by the Cyber Systems and Technology Group (formerly the DARPA Intrusion Detection Evaluation Group) to provide researchers with examples of attacks and background traffic.

The test network consisted of a mix of real and simulated machines; background traffic was artificially generated by the real and simulated machines while the attacks were carried out against the real machines. The dataset includes email, browsing, FTP, Telnet, IRC, and SNMP activities [2].

The main criticism against the DARPA IDS evaluation data set is that the test bed traffic generation software is not publicly available, and hence it is not possible to determine how accurate the background traffic inserted into the evaluation is [3]. Also the evaluation criteria does not account for system resources used, ease of use, or what type of system it is [4]. In his critique of DARPA evaluation [5], McHugh questioned a number of their results, starting from usage of synthetic simulated data for the background and using attacks implemented via scripts and programs collected from a variety of sources.

KDD’99 (University of California, Irvine 1998-99):

This dataset is an updated version of the DARPA98, by processing the tcpdump portion. Lee and Stolfo [6], one of the participating teams of the DARPA event, gave their feature extracted and preprocessed data to Knowledge Discovery and Data Mining yearly competition which was held in conjunction with

KDD'99. Since 1999, KDD'99 is the main dataset for the assessment of anomaly detection techniques [7].

KDD'99 is considered to be a heavily imbalanced dataset to attack instances [5]. Approximately 80% percent of flow is attack traffic (3925650 attack instances in total 4898430 instances). Normally, a typical network contains approximately 99.99% percent of normal instances. KDD'99 violates this principle. Most research papers need to re-sample the dataset to conform to typical network normality assumption, particularly anomaly detection papers. Another criticism is that U2R and R2L attacks are very rare, making up 00.001 and 00.02 percent of the total output classes.

Another issue in the dataset is that there is no exact definition of the attacks. For example, probing is not necessarily an attack type unless the number of iterations exceeds a specific threshold. Similarly, a packet that causes a buffer overflow is not always representative of an attack. Under such conditions, there should be an agreement on the definitions between the evaluator and evaluated. In DARPA'98, however, there are no specific definitions of the network attacks [8].

DEFCON (The Shmoo Group, 2000-2002):

The DEFCON datasets [9] contains data logged during the Capture the Flag Contest at DefCon. The Shmoo Group donated these datasets to promote the creation of more secure software and to offer data for research purposes. The DEFCON-8 dataset, created in 2000, contains port scanning and buffer overflow attacks, whereas DEFCON-10 dataset, which was created in 2002, contains port scan and sweeps, bad packets, administrative privilege, and FTP by Telnet protocol attacks.

Due to the nature of the DEFCON datasets, it is impossible to find any information about the attack scenarios within it. Moreover, the datasets are unusual and contain a huge number of attacks in a short period of time. As a consequence, intrusion detection systems will not exhibit good performance in real-world because attacks are usually less intensive [10]. This creates a gap between what the dataset represents and what real-world network traffic looks like [11].

CAIDA (Center of Applied Internet Data Analysis 2002-2016):

CAIDA, the Center for Applied Internet Data Analysis, collects several different types of data at geographically and topologically diverse locations, and makes this data available while preserving the privacy of individuals and organizations who donate data or network access [12].

The CAIDA dataset was found to be mostly populated by connections where only half the communication for any given flow was present. This is because the data was gathered on an Internet backbone link, where politics, financial concerns, and sheer practicality often work hand in hand to nearly guarantee that most flows will appear unidirectional. Researchers posit that this may account for the decreased consistency rates on the CAIDA traces [13].

LBNL (Lawrence Berkeley National Laboratory and ICSI 2004-2005):

Motivated by the fact that nearly all of the studies of enterprise traffic available in the literature focus on individual LANs rather than whole sites, the goal of LBNL/ICSI Enterprise Tracing Project [14] is to characterize internal enterprise traffic recorded at a medium-sized site, and to discover how modern enterprise traffic is similar to wide-area Internet traffic, and how it is quite different.

This project has collected 11GB of anonymized data that spans more than 100 hours of activity from a total of several thousand internal hosts. However, this is a dataset without payload and suffers from heavy anonymization. Also, it is not suitable to use with respect to IDS evaluation because of the lack of attack representations.

NSL-KDD (University of New Brunswick 2009):

To reduce deficiencies of KDD'99 dataset for machine learning algorithms, Tavallaee et al. [8] introduced NSL-KDD dataset. NSL-KDD has been generated by removing redundant and duplicate instances, also by decreasing the size of the dataset. The number of records in the NSL-KDD train and test sets are reasonable. This fact makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.

Since it is a re-sampled version of KDD'99, NSL-KDD dataset still suffers from some of the problems discussed by McHugh [5] and cannot be considered a perfect representative of existing real networks.

CDX (United States Military Academy 2009):

The CDX was held 21-24 April 2009 and consisted of teams from the U.S. Military Academy (USMA) and seven other military colleges. These teams built and secured operational networks within constraints specified by the exercise. Each team built their network from trusted operating system distributions (both Linux and Windows-based), but was required to integrate three untrusted workstations which were provided by the exercise organizers as VMWare images [15].

The weakness of the CDX revolves essentially around the lack of the volume and diversity of traffic normally seen in production networks. Due to the potential risks, the warfare competition games are normally conducted on isolated networks. Therefore, these exercises lack typical Internet background noise. Moreover, the limited duration of competitions (4 days) and periods of attacker inactivity are issues of concern when testing anomaly detection systems that require a training period [15].

Kyoto (Kyoto University 2009):

Kyoto 2006+ dataset was built during 3 years of real traffic data (Nov. 2006 to Aug. 2009) that has been generated using honeypots. The purpose of this dataset was to enable IDS researchers and operators to obtain more practical, useful and accurate evaluation results. It is important to note that the researchers regarded all traffic data captured from the honeypots as attack data and regarded all traffic data of the mail and DNS server as normal data.

Some of the weaknesses of the dataset include the fact that it has a limited view of network traffic because only attacks directed at the honeypots can be observed. It also has low benign traffic from different servers and network environments. Even regarding the attacks, they were not manually labeled, so the creators of the dataset relied on intrusion detection systems and other security solutions for detection and labeling. This renders the output classes (normal or attacks) unreliable.

Twente (University of Twente 2009):

The researchers from the University of Twente have created a labeled dataset for flow-based intrusion detection. The dataset aims to be representative of real traffic and complete from a labeling perspective. The goal here is to provide a rich dataset for tuning, training and evaluating ID systems [16].

With respect to the experimental setup, a honeypot was installed on a virtual machine running on Citrix XenServer 5. The decision to run the honeypot as a virtualized host is due to the flexibility to install, configure and recover the virtual machine in case it is compromised. In addition, a compromised virtual machine can be saved for further analysis. The virtual machine in the experimental setup ran for 6 days. The honeypot was hosted in the university network connected to the Internet. The data collection resulted in a 24 GB dump file containing 155.2 million packets [16].

One of the main drawbacks that is inherent in using a honeypot is the process of labeling attacks. Since the setup is connected to the internet, it is impossible to control and classify incoming attack traffic. Aside from the inherently flawed process of labeling, the presented traffic in the dataset mainly consists of malicious traffic. This means that during the evaluation of an IDS, this dataset allows to detect false negatives but not false positives. Also, there's the lack of volume and diversity of attacks.

ISCX 2012 (University of New Brunswick 2012):

This dataset was generated by a systematic approach following a guideline for generating realistic and useful IDS evaluation datasets. Their approach is based on the notion of different profiles, which contain detailed descriptions of intrusions and abstract benign traffic for applications, protocols, or lower level network entities. The two profiles are named Alpha and Beta profile; The Alpha profile carries out various attack scenarios to stream the anomalous segment of the dataset, while the Beta profile generates realistic benign network traffic with background noise [17].

The testbed network consists of 21 interconnected Windows workstations. The Windows operating systems are chosen so that it would be possible to exploit a diverse set of known vulnerabilities against the testbed environment. The potential vulnerabilities are exploited through third-party scripts or exploitation tools such as Metasploit (Rapid7) which allow for the automation of the attack process.

Despite its many advantages and sound incremental steps and guidelines to create a useful IDS evaluation dataset, this dataset suffers from a few shortcomings; it does not represent new network protocols (HTTPS) since nearly 70% of today's network traffic is HTTPS. It lacks attack diversity, and moreover, the distribution of the simulated attacks is not based on real world statistics [18].

ADFA-LD12 (University of New South Wales 2013):

The focus of this dataset is on host-based intrusion detection system (HIDS) evaluation. The experimental setup consists of a Linux local server (Ubuntu 11.04) which was selected as the host for the ADFA-LD12 [19]. To allow for a web based attack vector, Apache Version 2.2.17 running PHP Version 5.3.5 was installed and enabled. Several attacks are executed, including password bruteforce against FTP and SSH, client side attacks, java based payloads, PHP Remote File Inclusion attacks, etc.

The main issue with this dataset is associated with what is considered to be normal behavior. There exists only one computer (Ubuntu Linux OS) and there is no description about the services and software installed. As such, it is not possible to take it as a base to generalize a common behavior of other machines on the Internet. It also suffers from a lack of attack diversity.

UNSW-NB15 (University of New South Wales 2015):

This dataset has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. Existing and novel methods are utilised to generate the features of the UNSW-NB15 data set [20]. The authors used an attack automatic generation tool called IXIA PerfectStorm to implement nine families of real and updated attacks against several servers. The IXIA tool reportedly contains all information about new attacks that are updated continuously from a CVE site which serves as a dictionary of publicly known information security vulnerabilities and exposures.

Despite some promising ideas and approaches such as using a continuously updated list of known vulnerabilities to carry out attacks and proposing a hybrid of real and synthesized traffic, the dataset still has its shortcomings. The main problem being the completely synthetic generation of attack traffic. Also, the decision to use a commercial tool for simulating a part of the network traffic undermines the dataset in terms of transparency. Moreover, the authors did not model or describe normal or attack traffic.

NGIDS-DS (University of New South Wale 2017):

This dataset [21] is generated at the cyber range infrastructure of the Australian Centre of Cyber Security (ACCS) at the University of New South Wale (UNSW). The dataset consists of the collection of normal and abnormal host and network activities that are performed in a simulation. This dataset also uses the IXIA PerfectStorm tool to produce a mixture of modern normal and abnormal cyber traffic.

In the case of host-based IDS evaluation, the host logs of NGIDS-DS can be used. While in the case of network-based IDS, NGIDS-DS network pcaps can be used. It is certainly a good idea to include both host and network information in the dataset. However, the main issue here relates to using a commercial tool for simulating traffic. This creates a divide in understanding the ground-truth information as it pertains to attacks as well as to the process of data capture.

UGR‘16 (University of Granada 2017):

The dataset presented here is built with realistic background and synthetic attack traffics. The data is obtained from a real network of a Tier 3 Spanish ISP. The ISP is a cloud service provider, so that many of the services implemented in the network are virtualized. The total number of flows in the dataset is above 16,900M. The data is provided in anonymized flow format. The main advantage of this dataset is its usefulness for evaluating IDS that consider long-term evolution and traffic periodicity. Models that consider differences in daytime/night or labour weekdays/weekends can also be trained and evaluated with it.

The main drawback for this dataset is the lack of attack diversity. Due to the fact that only netflow traffic is being collected and, thus, payloads are not considered in the trace, the authors decided to not include attack types that are detectable through payload analysis.

CIDDS-001 (Coburg University of Applied Sciences 2017):

CIDDS-001 is a labelled flow-based data set for evaluation of anomaly-based NIDS. For its creation, a small business environment was emulated using OpenStack. This environment includes several clients and typical servers like an E-Mail server or a Web server. Python scripts emulate normal user behaviour on the clients. The CIDDS-001 contains unidirectional NetFlow data [22].

This dataset can be critiqued for not describing the distribution of user activities from real network traffic, which will allow realistic parameterization of the scripts that control simulated user behaviour. Additionally, attack scenarios are limited in scope and variation.

CICIDS2017 (University of New Brunswick 2017):

CICIDS2017 dataset is, in many ways, a continuation of the efforts of the University of New Brunswick when they generated the ISCX 2012 dataset. CICIDS2017 consists of labeled network flows, including full packet payloads in pcap format and the labeled flows (CSVs). It also includes the results of the network traffic analysis using CICFlowMeter (an open source tool that generates network traffic flow from pcap files, and extracts features from these flows) [18].

Despite the many merits of this work, it still has its shortcomings, mainly the lack of scale in terms of abstracting benign traffic based on only 25 users. Also, their paper [18] reveals a conceptual error in executing DoS attacks using Heartleech. Heartleech is used to carry out the Heartbleed attack, but this is not a Denial-of-Service attack by any means. Another curious point is whether or not this dataset has the ability to include sophisticated attack models such as cyber kill-chain.

CSE-CIC-IDS2018 on AWS (collaboration between CSE and CIC 2018):

This dataset is the result of a collaborative project between the Communications Security Establishment (CSE) and The Canadian Institute for Cybersecurity (CIC) that uses the notion of profiles to generate cybersecurity dataset in a systematic manner. It includes a detailed description of intrusions along with abstract distribution models for applications, protocols, or lower level network entities. The dataset includes seven different attack scenarios, namely Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and infiltration of the network from inside. The attacking infrastructure includes 50 machines and the victim organization has 5 departments including 420 PCs and 30 servers. This dataset includes the network traffic and log files of each machine from the victim side, along with 80 network traffic features extracted from captured traffic using CICFlowMeter-V3.

Although the intrusion detection studies using CIC-AWS-2018 Dataset [23] has not yet been much reported, there is plenty of research work has been done on an earlier version: CICIDS2017, which is mentioned above. It stands to reason that this dataset might suffer from the fundamental drawbacks of its predecessor.

2.2. Evaluation Metrics

Several metrics have been designed to measure the effectiveness of IDS. A lot of these metrics are derived from the confusion matrix. Next, we give an overview of the most commonly used basic and composite security-related metrics.

2.2.1. Confusion Matrix

The confusion matrix represents true and false classification results. The followings are the possibilities to classify events and depicted in Table 1 :

- True positive (TP): Intrusions that are successfully detected by the IDS.
- False positive (FP): Non-intrusive behavior that is wrongly classified as intrusive by the IDS.
- True Negative (TN): Non-intrusive behavior that is correctly labeled as non-intrusive by the IDS.
- False Negative (FN): Intrusions that are missed by the IDS, and classified as non-intrusive

		Predicted	
		Attack	Normal
Actual	Attack	TP	FN
	Normal	FP	TN

Table 1. Confusion matrix

2.2.2. Basic Security related Metrics

In spite of the representational power of the confusion matrix in classification, it is not a very useful tool for the sake of IDS comparison. To solve this problem, the basic metrics quantify various individual attack detection properties. These metrics produce numeric values that are easily comparable and are briefly explained in subsequent paragraphs.

- Classification rate (CR): It is defined as the ratio of correctly classified instances and the total number of instances.

$$CR = \frac{TP + TN}{TP + TN + FP + FN}$$

- Detection rate (DR) (Recall): It is computed as the ratio between the number of correctly detected attacks and the total number of attacks.

$$DR = \frac{TP}{TP + FN}$$

- False positive rate (FPR): It is defined as the ratio between the number of normal instances detected as attack and the total number of normal instances.

$$FPR = \frac{FP}{FP + TN}$$

- Precision (PR): It is the fraction of data instances predicted as positive that are actually positive.

$$PR = \frac{TP}{TP + FP}$$

- F-measure (FM): For a given threshold, the FM is the harmonic mean of precision and recall at that threshold. F-Measure is preferred when only one accuracy metric is desired as an evaluation criterion.

$$FM = \frac{2}{\frac{1}{PR} + \frac{1}{Recall}}$$

- Other basic metrics are the positive predictive value (PPV) and negative predictive value (NPV). The first quantifies the probability that there is an intrusion when an IDS generates an alert, whereas the latter quantifies the probability that there is no intrusion when an IDS does not generate an alert. PPV and NPV are expressed as follows:

$$PPV = P(I|A) = \frac{P(I)P(A|I)}{P(I)P(A|I)+P(\neg I)P(A|\neg I)}$$

$$NPV = P(\neg I|\neg A) = \frac{P(\neg I)P(\neg A|\neg I)}{P(\neg I)P(\neg A|\neg I)+P(I)P(\neg A|I)}$$

2.2.3. Composite Security Related Metrics

Area under the ROC curve (ROC): A receiver operating characteristic (ROC) curve plots true-positive rate against the corresponding false-positive rate exhibited by a detector. In the context of IDSes, a ROC curve depicts multiple IDS operating points of an IDS under test, and as such, it is useful for identifying an optimal operating point or for comparing multiple IDSes. Area under the ROC curve is used as a summary statistic. The problem with ROC curves is that they might be misleading and simply incomplete for understanding the strengths and weaknesses of the candidate system [5].

Cost-based metrics: the expected cost metric [24] proposes the measure of cost as an IDS evaluation parameter. The measure of cost is relevant in scenarios where a response that may be costly is taken (e.g., stopping a network service) when an IDS generates an attack alert. the cost ratio is used to calculate the expected cost C_{exp} of an IDS operating at a given operating point. In doing so, one can compare IDSes based on the estimated costs when each IDS operates at its optimal operating point.

3. Proposed Platform

3.1. Generic Overview

We have so far surveyed a good number of IDS evaluation datasets. Although they constitute important advancements in the area of IDS evaluation and a benchmark for future research, they all suffer from a number of recurring drawbacks. Namely, they become obsolete in the span of a couple years due to outdated attack definitions, and they contain unrealistic network traffic. There's also the question of generating a real or a synthetic dataset; a real dataset causes privacy issues and subsequently undergoes a process of anonymization that decrease the ground truth value of the dataset, on the other hand, a synthetic dataset gives the possibility of correctly labeling the traffic but also it is unrealistic by default. The lack of volume, diversity and representativeness is another recurring issue.

Due to the number of drawbacks discussed above, our aim in this work is to create an evolving platform for IDS evaluation. Bypassing a bulk of these drawbacks is an intrinsic design feature in our proposed platform. In order to create an evolutive platform, there is a need for dynamic infrastructure that allows continuous and automatic change. Here are a number of design principles that we want in our platform:

- Reproducibility: It should be possible to effortlessly and reliably rebuild the infrastructure of the platform or any element of it. Decisions about which software and versions to install and the like should be captured in the scripts.
- Repeatability: Building on the reproducibility principle, any action you carry out on your infrastructure should be repeatable. This is an obvious benefit of using scripts rather than making changes manually. This enables other researchers and practitioners to start up our platform and run it independently of the underlying hardware.
- Live evaluation: Traditionally, IDS evaluation is carried out using a static benchmark dataset. This method is prone to a number of shortcomings discussed above. We want to create an environment that resembles what IDS does in real life. We want researchers to be able to fire up our platform, plug in or upload their IDS solution, and run the experiment. Subsequently, they would be able to evaluate their solution and compare it with other benchmark IDS solutions. This is a novel way of carrying out IDS evaluation that is not present in scientific literature.
- Modularity: with the advent of new computing paradigms such as cloud computing, numerous IDS research now focuses on specific computing infrastructures. We expect our platform to be highly modular and flexible in evaluating security mechanisms according to their objectives. For example, an IDS in a cloud computing environment needs to be tested for virtual machine-related issues, whereas traditional systems don’t.
- Realism: It is natural that the most realistic traces are those collected in the real world. Unfortunately, these traces would compromise privacy and hence are rarely published, also they are not labeled and must undergo a manual process of labeling and anonymization that substantially decreases the ground truth present in the trace data. On the other hand, synthetic traces (traces that have not been collected but artificially generated) can avoid the problem of privacy and are correctly labeled but they usually require higher effort and deeper domain knowledge to achieve a realistic result. In that sense, we intend for our platform to be as realistic as possible in terms of traffic generation, real world attack representativeness, and system setup. This will surely be a continuous and evolutive effort to try to approach real world conditions as best as can be.
- Automatization: In order to be able to achieve the objectives of our IDS evaluation platform and have the possibility of consistently reproducing live evaluation independently of the underlying hardware, we need to take advantage of automation techniques in defining the infrastructure, as well as mapping out the traffic and interaction between different components.

Automation through scripts opens up the possibility of having a sufficiently complete description of our platform stored in script files that would be easily available for reuse. Automation also covers the aspect of normal/malicious activity generation inside our testbed architecture.

The figure below offers a global view of our proposed platform. This is not to be perceived as a detailed and actionable solution, instead it portrays a generic overview of the flow and interaction between the different components, showing basic internal processes and functions. As discussed above, our proposed platform relies on source code scripts to carry out all the major processes, conforming to the design principles we have laid out.

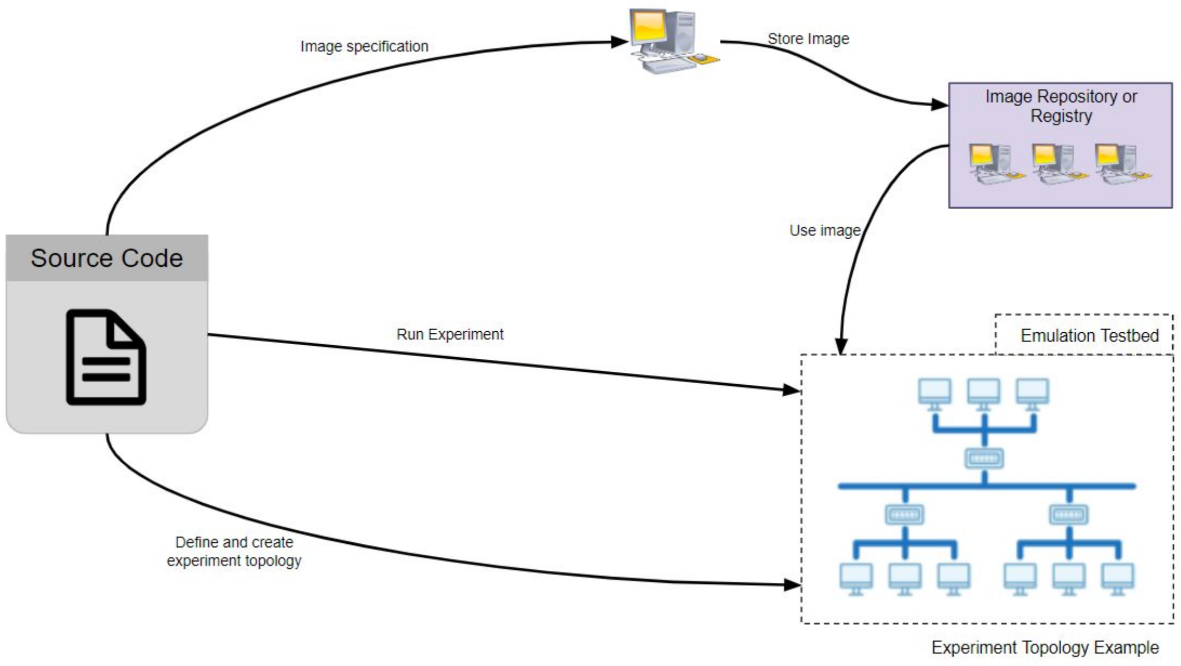
Image specification
Store Image
Image Repository or Registry
Use image
Source Code
Run Experiment
Emulation Testbed
Define and create experiment topology
Experiment Topology Example

Figure 1. Generic overview of our proposed platform

Our proposed platform is supposed to host an experiment topology consisting of a combination of different types of nodes in subnetworks, complete with necessary network equipment and functionalities. Figure 1 demonstrates a generic topology as an example of the many topologies that the emulation testbed might host.

Since our platform is designed to be modular and flexible, the network topology is subject to change according to objectives. It needs to be mentioned that real world enterprise or governmental agency networks are much more complex and bigger in scale than this example. This also brings to mind the consideration that with a sufficiently large and complex network, we need to think about where to station traffic capture points across the network.

The central intent of our platform is to put in place a network, put into motion the normal and malicious traffic, all of which are to be originated from automation scripts and dispersed across nodes. This gives us the ability to describe a network topology, as well as the traffic, in specific terms inside scripts and start the experiment without ambiguity. Conforming to the design principles discussed before, we intend for this experiment topology as well as the traffic to be synthetic and able to be replicated independently of the underlying hardware.

The image specification component in the figure above is in charge of producing an image or template of a system or application that can then be stored in a registry and subsequently instantiated. This is useful for our proposed platform because ultimately, the experiment is going to be composed of a number of computing nodes running an operating system image. For the purposes of transparency and having a consistent baseline in the evaluation processes, we need to provide researchers that will use our platform with either a ready-to-use OS image, pre-configured with all the necessary software, or a scripted way of building the OS instance with the necessary software and configuration.

It deserves to be mentioned that once a network topology and an information system is in place, we can run normal and malicious traffic. Since we control everything, we can monitor and collect network data as well as host data. This gives our platform the capability of being able to evaluate both HIDS and NIDS.

3.2. Choice of Testbed

In order to develop our platform, we need to consider the different emulation testbed options that can be used. To that end, we considered three main options: a public cloud, Grid'5000 [25] (a large-scale and flexible testbed for experiment-driven research), and Deterlab [26] (a scientific computing facility for cybersecurity research). The following is a table that compares these three options.

Table 1. A comparison of emulation testbeds

Platform	Pros	Cons
Public Cloud	- Computing services at a cost. - Ease of use. - No capacity limits.	- Security and privacy. - Limited control and flexibility. - Vendor lock-in.
Grid'5000	- Access to a large amount of resources: 800 nodes grouped in homogeneous clusters. - Funding from Inria and CNRS. - Highly reconfigurable and controllable. - Supports open science and reproducible research.	- Requires substantial engineering efforts. - Default max uptime of an experiment is 12h.
Deterlab	- Platform for repeatable cyber-security research. - Funding from NSF, DHS, and DARPA. - Control of a pool of interconnected nodes. - Ease of use and scalability. - Scripted high-level description of experiments.	- Default max uptime of an experiment is 24h. - Requires pre-planning and scheduling.

We find that Deterlab constitutes the best option for us moving forward with our proposed platform because of its built-in capabilities of using high level scripts to create network topologies and run experiments.

4. Traffic and Workload Generation

In this section, we will discuss the different relevant techniques in literature, as well as what we elected as our choice, for automated and dynamic generation of both normal and malicious traffic.

4.1. Normal Traffic

Generation of network workload is a fundamental component of security research. Here, we focus on the generation of realistic workload at network level. With the term realistic we mean synthetic workload able to replicate the main features of a real workload specifically targeted to a particular network scenario.

For us, synthetic network workload generation should be able to appropriately capture the complexity of real world workload in different scenarios and customly alter specific properties of such workload for the purpose of the experiment [27]. What follows is a (non-exhaustive) list of traffic generators that are most used in literature: several powerful tools exist, each of them with peculiar characteristics, but none of them full flexibility and configurability.

- RUDE/CRUDE [28] Rude/crude are two programs that can generate and collect UDP traffic. RUDE (Real-time UDP Data Emitter) is a small and flexible program that generates traffic to the network, which can be received and logged on the other side of the network with the CRUDE (Collector for RUDE).
- MGEN [29] MGEN provides the ability to perform IP network performance tests and measurements using TCP and UDP/IP traffic. The toolset generates real-time traffic patterns so that the network can be loaded in a variety of ways. The generated traffic can be logged for analysis. Script files are used to drive the generated loading patterns over the course of time.
- D-ITG [30] is a platform capable to produce traffic at packet level accurately replicating appropriate stochastic processes for both Inter Departure Time and Packet Size random variables (exponential, uniform, cauchy, normal, pareto). D-ITG supports both IPv4 and IPv6 traffic generation and it is capable of generating traffic at network, transport, and application layers.
- BRUTE [31] BRUTE is a user space Linux application designed to produce a high load of customizable IPv4 and IPv6 Ethernet traffic. The software has been designed to achieve high precision and performance in the traffic generation.
- OSTINATO [32] Ostinato is a packet generator and network traffic generator with a friendly GUI. Also a powerful Python API for network test automation. Craft and send packets of several streams with different protocols at different rates..

- SEAGULL [33] Seagull is an open-source multi-protocol traffic generator test tool. Primarily aimed at IMS (3GPP, TISPAN, CableLabs) protocols, it is conceived for functional, load, endurance, stress and performance/benchmark tests.
- Tmix [34] Tmix is a traffic generator for ns-2 that reads a packet header trace, derives a source-level characterization of each TCP connection, and then emulates in ns-2 the application that created the TCP connections in the trace.
- Harpoon [35] Harpoon generates TCP and UDP packet flows that have the same byte, packet, temporal and spatial characteristics as measured at routers in live environments. Harpoon is distinguished from other tools in that it can self-configure by automatically extracting parameters from standard Netflow logs or packet traces..
- KUTE [36] KUTE is aimed at being a maximum performance traffic generator and receiver mainly for use with Gigabit Ethernet. It is composed of Linux kernel modules which send/receive packets directly to the hardware driver (tested only with Ethernet hardware).
- LiTGen [37] LiTGen is an open-loop packet-level traffic generator, which statistically models IP traffic (resulting from Web requests) on a per user and application basis.
- Swing [38] is a closed-loop, network responsive traffic generator that accurately captures the packet interactions of a range of applications using a simple structural model. Starting from observed traffic at a single point in the network, Swing automatically extracts distributions for user, application, and network behavior. It then generates live traffic corresponding to the underlying models in a network emulation environment.
- NetSpec [39] NetSpec provides a fairly generic framework that can be used by a user to control multiple processes across multiple hosts from a central point of control for doing network testing. NetSpec consists of daemons that implement traffic sources/sinks and various passive measurement tools.
- LegoTG [40] is a modular framework for composing custom traffic generation. It makes it easy to combine different traffic generators and traffic modulators (e.g., delay models), and to port the same background traffic to different topologies.
- Iperf [41] Iperf is a tool for measuring maximum TCP and UDP bandwidth performance. Iperf allows the user to tune various parameters and UDP characteristics and it reports bandwidth, delay jitter, datagram loss.
- TCPivo [42] TCPivo is a tool that provides high-speed packet replay from a trace file using standard PC hardware and freely available open-source software
- TCPreplay [43] TCPreplay has the ability to use previously captured traffic in libpcap format to test a variety of network devices.
- TCPopera [44] TCPopera has been conceived to evaluate the performance of IDSs by replaying background traffic that mimics real-world behavior.
- ParaSynTG [45] ParaSynTG is a synthetic trace generator for source-level representation of Web traffic with different characteristics such as document size and type, as well as temporal locality of requests.
- UniLoG [46] UniLoG is a flexible tool to generate realistic and representative server and network loads, in terms of access requests to Web servers and creation of typical Web traffic.

- SURGE [47] SURGE is a tool to generate Web requests according to measured statistical properties.

Packet-level generators (Iperf, BRUTE, MGEN, Ostinato) are usually used for stress testing firewalls and servers or for end-to-end performance testing. Since all these solutions generate packets with dummy or random payload, they can’t be used for realistic traffic generation. Replay engines (TCPreplay, TCPivo, TCPopera), on the other hand, aim to reinject packets to the network as they were previously recorded with as accurate timing as possible. The collective drawback of these generators is that the measurement contains user sensitive information [48].

More sophisticated traffic generators like Harpoon, Swing, and Tmix are able to mimic the behavior of previously recorded traces by more complex traffic modeling. Although all these solutions can mimic the behavior of real network traffic in aspects of many different metrics, all these approaches miss to provide realistic packet payloads thus are not adequate in the context of our platform.

The idea of using GUI testing tools for controlling application in place of a human user was proposed in [49] where authors present a finite state machine model for piloting applications. However, their automation only covers basic applications (e.g., Internet Explorer, Outlook and Microsoft Word) using an isolated testbed and their goal is to present the effect of using anti-virus software on the system's performance. Our goal is to provide repeatable traffic generation in more versatile environments.

Our solution with respect to normal traffic generation builds on the idea of using GUI testing tools for automatically piloting applications. The experiment network which will be created using our platform will contain real services installed on nodes and will house versatile traffic, depending on the use case. Moreover, we will use GUI remote control tools that rely on scripts to trigger mouse and keyboard events and carry out user behaviour scenarios.

A user behavior scenario indicates a series of actions that a user does to interact with GUI applications. For example, the user opens a web browser, navigates to a torrent site, downloads a file, opens it in a torrent application and five minutes later the user closes it. Our platform needs to be able to emulate and mimic such a specific user behavior scenario. However, it is a crucial point that in order to achieve realistic results, we need to think about what we can do to construct a number of user behavior scenarios. One option is to create a model that monitors, or takes as an input, real world traffic in order to extract relevant and realistic user behavior scenarios. These scenarios will be stored and can be used repeatedly depending on the use case.

For the emulation of user behavior we will use PyAutoGUI [50]. PyAutoGUI is a cross-platform GUI automation Python module used to programmatically control the mouse & keyboard. Its primary goal is to make it possible to create automation scripts that pilot the operating system through GUI. Other tools for GUI controlling exist such as AutoIt [51], AutoHotKey [52] but they can only be used on Windows, others, such as Sikuli [57] and AutoKey [58], can only be used on Linux. We need to be able to host different operating systems on our platform, therefore, PyAutoGUI is the tool that we will use.

4.2. Malicious Traffic

To be able to evaluate an IDS, it is necessary to generate malicious activities on the monitored information system. Moreover, to obtain representative measures, the generated activities should have characteristics from real world malicious activities; they also should have goals that are realistic with relation to the monitored system and should cover a maximum of unitary attacks. These attack scenarios, once launched against one or more targets, would result in malicious traffic generation.

Recent advances in machine learning techniques allow us to consider their use to generate attack scenarios that try to mimic real world attacks. For example, the model of reinforcement learning [53] is based on an agent that observes his environment, executes an action, and gets a reward accordingly. This generic model seems to match this case of our objective here: the agent can model an attacker that tries a number of actions on the system and gets rewards when they are successful.

Other state-of-the-art attack scenario generation techniques [54] are based on the complete knowledge of the information system (topology, configuration, vulnerabilities, etc.) and on attack graph generation (with nodes representing a state of the system). Their objective was not to generate realistic malicious activities for IDS evaluation but to analyze and propose solutions to protect the system of the discovered attacks.

An attack graph is a model for analyzing the security of a network by modeling the way attackers combine and exploit vulnerabilities in a network to achieve their attack goals [55]. This model represents the state of systems using a set of variables such as the existence of vulnerabilities in the system, or connectivity between multiple machines. An attack graph displays the path of an attack to achieve its objectives, in other words, it illustrates how an attack could occur by exploiting existing vulnerabilities and configuration of systems.

MulVAL framework is used to generate attack graphs [56]. It is a network security analyzer that uses Datalog as its modeling language. Datalog collects a variety of information on the vulnerability of servers, file servers, web servers and clients, such as machine configuration, server configuration, and other relevant information. MulVAL uses this data to produce an attack graph in the form of text data and logical graph, which can be used to analyze how an attack occurs in a network.

In the context of our platform, we are capable of writing a file that describes a network topology of an information system and starts it up in a reliable and repeatable fashion. An even more detailed file describing the information system as well as data about configuration and intentionally pre-existing vulnerabilities can be created. This detailed file will subsequently be fed to the MulVAL framework, which will produce an attack graph. This attack graph can then be used as the blueprint for attack vectors which will be executed against the information system to generate malicious traffic.

MulVAL framework is a valuable tool for generating an attack graph; it ultimately gives us the path and multiple actions by which an attack can be carried out. However, implementing these multiple actions that lead to an attack is out of MulVAL's scope. Therefore, other tools can be used to execute the previously mentioned actions. Moreover, MulVAL framework doesn't provide information on the temporal aspect of carrying out an attack. Also, seeing as MulVAL relies on vulnerability databases provided by the bug-reporting community, it doesn't offer a mechanism for the reconnaissance step in an attack where an intruder selects a target, researches it, and attempts to identify vulnerabilities in the network. Ongoing work is investigating how to overcome these limitations.

5. Final Description

In this section, we will demonstrate the steps needed to deploy our platform using automation scripts. The following is a figure displays the different steps:

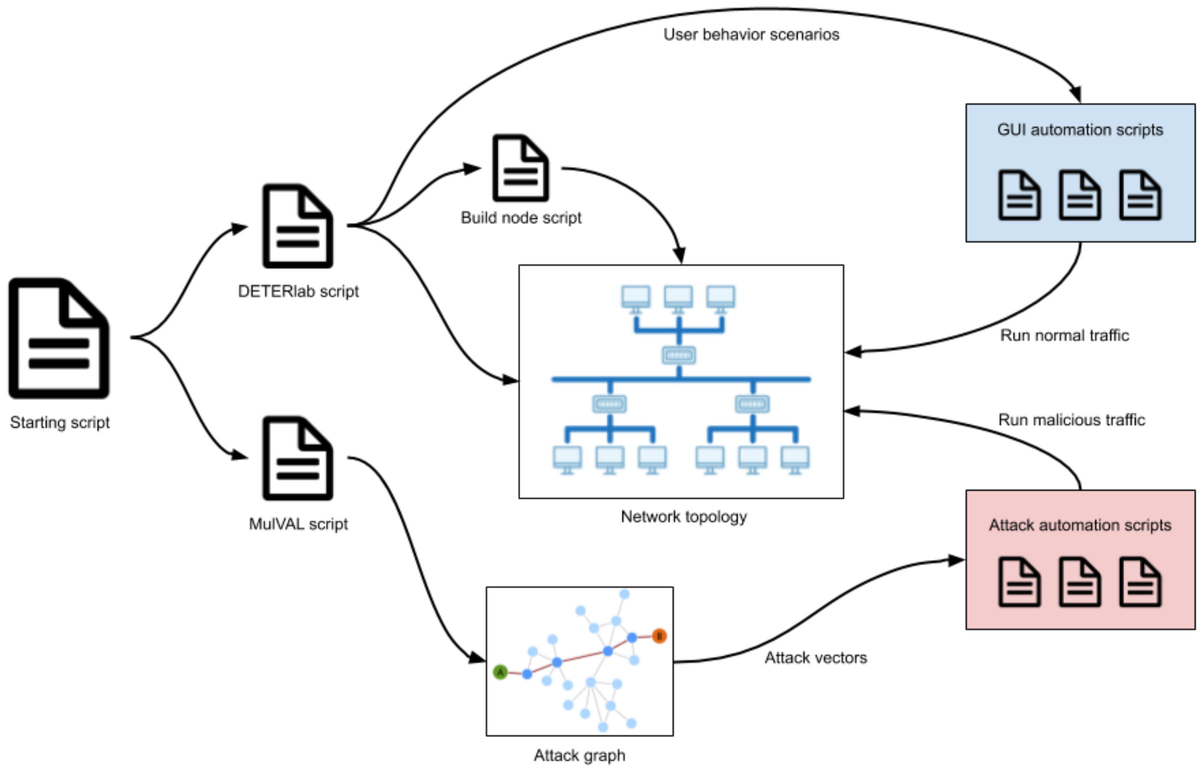

Figure2. Comprehensive description of the script-based usage of our platform

The main starting point in using our platform is the 'starting script', as shown in figure 2. This script contains detailed and exhaustive information about the network topology and the information system used in the experiment: machine nodes and their operating systems, node configurations, network configuration, node interaction, access policies, vulnerabilities and their properties, and so on. The format of this file could be JSON for its readability and capabilities.

From the starting script, we can create a 'MulVAL script' intended to be the input for MulVAL framework in order to generate the corresponding attack graph. This file will contain all the necessary information that MulVAL relies on. The resulting attack graph will contain different basic actions that combine to form a standalone attack. A repertoire of 'attack automation scripts' that can perform these basic actions will be available on our platform so that the attack graph can be interpreted and the basic actions can be successively executed against the information system, generating the malicious activity.

From the starting script also, we can generate the 'DETERlab script' which will be used on the deterlab testbed in order to create the experiment. DETERLab uses the "NS" ("Network Simulator") format to describe network topologies so that decides the format of this file. Inside this script, we can define numerous details about the nodes and how they connect to each other. From the DETERlab web application, connected with our credentials, we swap in the experiment. Individual nodes used in the experiment can be accessed through SSH if necessary.

The 'DETERlab script' can call the 'Build node script' which can be executed on specific nodes. This script has the goal of installing and setting up all the necessary software and dependencies onto nodes. After this step, the nodes will be able to run automation scripts for generating normal and malicious traffic. Note that the 'Build node script' only sets up the node, the 'DETERlab script' would be responsible for telling which node to run which user behavior scenarios and in what order.

6. Experiment Design

In this section, we will outline an experiment design that we are currently implementing. The purpose of this implementation is to provide a proof of concept that demonstrates the feasibility of our platform. We are considering a single use case that consists of a number of subnetworks and resembles the network topology of a small or medium-sized enterprise in terms of functionality and network segmentation. The following figure represents the experiment design.

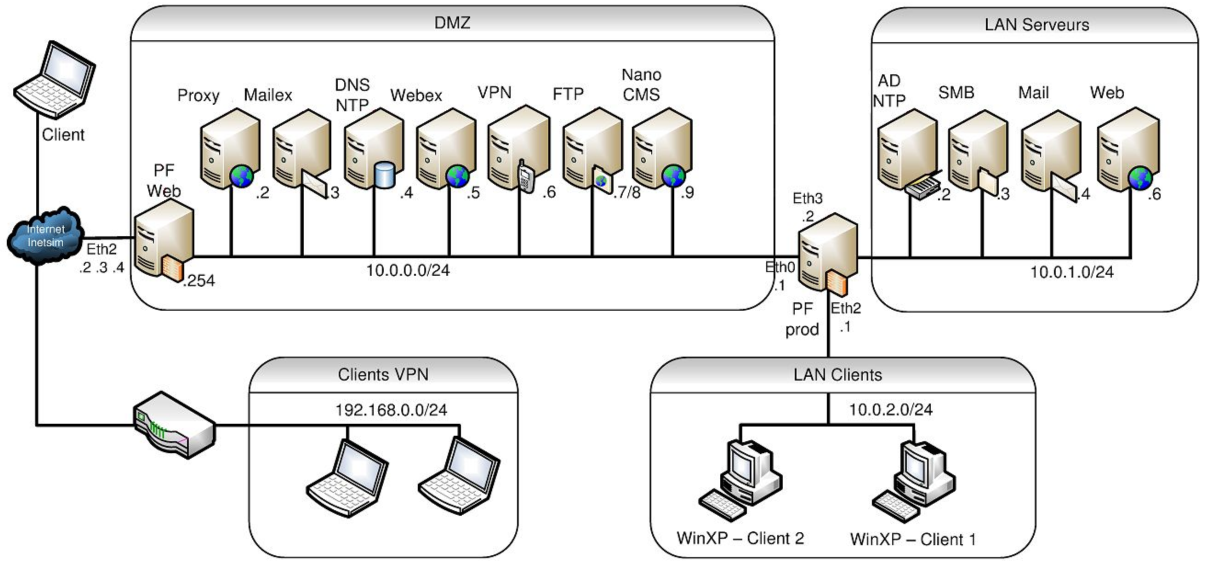

Figure 3. Network topology of our experiment design

In the illustration above, we have a well-defined network structure that includes a secure internal network zone and an external untrusted network zone, with intermediate security zones. Security zones are groups of servers and systems that have similar security requirements and consist of a Layer3 network subnet to which several hosts connect. The firewall offers protection by controlling traffic to and from those hosts and security zones, whether at the IP, port, or application level.

In our experiment design, we use firewall security zone segmentation to keep servers separated. In our example we use two firewalls (PF Web and PF Prod) and one DMZ (demilitarized) zone and internal zones for LAN servers and LAN clients. A DMZ zone is an isolated Layer3 subnet.

The servers in the DMZ zone are Internet facing in order to function. Because they face the internet, these servers are the most vulnerable to attacks so they should be separated from servers that do not need direct Internet access. By keeping these servers in separate zones, one can minimize damage if one of the Internet facing servers is compromised. Our example also has an internet client that can connect to the network structure, as well as VPN clients connecting through the Internet.

A network topology very similar to this experiment design is currently being hosted on DETERlab and can readily be instantiated thanks to the relevant description script. We are currently in the process of setting up the services that each machine node should have so that normal traffic that flows from one node to another can be as realistic as possible.

In terms of malicious traffic, we are currently elaborating the file that MulVAL framework will take as an input. Inside the file, an end goal should be explicitly defined. The first attack objective that we will include in this file is for the client connected through the internet to have unauthorized access to the web server inside the LAN server. Other attack objectives should follow shortly in the future of the developing this experiment.

7. Future works

7.1 Platform infrastructure

For prototyping an IDS test solution, we chose the DETERlab platform. While being dedicated to repeatable cyber-security research work, DETERlab presents the important advantage to get as close as possible to a real system, with real (i.e., non-virtualized) machines implementing real services. In addition, DETERlab uses high-level scripts to create network topologies and run experiments.

In our study, we defined a script format that allows us to define a test environment. This script is converted into two other scripts, one allowing to configure DETERlab (definition of the information system: machines, services, network), the other serving as an input to Mulval which is able to find an attack graph according to assigned objectives.

For the future, two avenues of work appear interesting.

Firstly, the setting up of a platform similar to DETERlab for the French (or European) security community. The aim here is to create a new sovereign research tool, which could be used for IDS testing, but more broadly for experiments in computer security. This platform will have to respect the design principles that we want:

reproducibility, repeatability, modularity, realism, automation, live evaluation. Its specification, design, deployment and operation is clearly an engineering work that must be done in cooperation with its future users (researchers and research engineers). Industrial cooperations will thus have to be sought.

A possible alternative, but to be studied, would be to come back to research platforms not necessarily dedicated to security, which do not necessarily follow to date the requirements listed above, but which could be complemented by an "over-layer" to correct these shortcomings. This work, although close to operational, should then be led by the academic world, which usually pilots this type of platform (we are thinking in particular of Grid5000). One advantage that we see in this approach is the benefit that will be derived from the successive evolutions of the "basic" environment. In the case of Grid5000, the possible link with systems not from the classical IT field, but relative to the IoT or the IoE, also seems very interesting to us.

In either case, the final result will have to be administered on a day-to-day basis and govern over a long period of time. Again, this is clearly engineering work, but it could be carried out by engineers in a variety of contexts: academic, industrial or governmental.

Finally, it should be noted that synergy could be found with security training facilities (cyber range). The objectives of these platforms are of course different, but the infrastructure, the properties needed, and the tools deployed to achieve these properties are quite similar.

7.2 Normal activity generation

In this section, we focus on the generation of realistic normal network traffic over real networks and using software platforms. Synthetic network traffic generation should be able to appropriately capture the complexity of real network workload in different scenarios and customly change specific characteristics of the workload according to the requirements of the experiment.

Our solution with respect to normal traffic generation builds on the idea of using GUI testing tools for automatically piloting applications. Different scenarios can be executed to generate versatile traffic. Essentially, we will use GUI remote control tools that rely on scripts to trigger mouse and keyboard events and carry out user behavior scenarios.

Creating a good model for what constitutes normal activity is a broad and challenging scientific question in itself and should be pursued. For the purposes of our research, two approaches appear to be promising:

In the short term, we can construct a Markov chain model for the way real users interact with each application [49]. Then, during the experiment, we can use the Markov chains to generate new scenarios similar in distribution to those generated by the real users, and use these to drive the applications on the testbed.

This approach offers a reasonably realistic set of workloads on the host, in terms of running processes, files and directories accessed, open network ports, etc. In order to emulate a human user, inputs generated by real human users are recorded and then a hierarchical Markov chain model is trained for the events sent to each application. Since we have a system in place in our experiment that automates GUI remote control tools, this approach can be implemented once we have a sufficient record of real human user events.

In the medium term, we can look at novel AI-based generative models such as Generative Adversarial Networks (GAN). In most cases, GANs are used for the purpose of generating images, so there is a challenge in applying this technique to generate network traffic that contains continuous and categorical data. However, there is an interesting work [59] about the generation of flow-based network data using GANs. This is achieved using techniques for transforming categorical attributes of flow-based network data (IP addresses, port numbers) into continuous attributes, considering that GANs can only process continuous data.

This approach can learn the characteristics of the training flow-based network traffic and generates new flow-based traffic with the same underlying characteristics. However, in our case, rather than flow-based data, we want to generate packet data that contains a complete record of all network activity, including packet payload that is transferred across the network. There is an inherent challenge in creating a GAN model that generates packet data. Moreover, we will need a training dataset that contains sufficiently rich packet data capture. Other generative models that could be impactful exist, such as Variational Autoencoders, and need to be studied as well.

7.3 Malicious activity generation

We can divide the future work for that domain in 3 parts. In the short term, it is necessary to build a knowledge base relating high-level actions and low-level implementations of the actions. In the medium term, evolutions to the attack graph method used in our work should be implemented to get a more realistic attack scenario. In the long term, the use of machine learning may bring some advantages over the use of an attack graph. We explain these possibilities in the next 3 subsections.

7.3.1 Knowledge base of high-level actions and low-level implementations

The attack graph produced by Mulval gives the steps that the attacker can implement to achieve his/her goal by exploiting vulnerabilities in the information system. The low-level implementation of the exploitations is not given by the attack graph. In our work, we only developed the low-level implementations of the vulnerability exploitations linked to our PoC. In a more realistic setting, it is necessary to build a knowledge base relating vulnerabilities, high-level actions given by the attack graph and low-level implementations. This knowledge base can be used to choose a right low-level implementation for the scenario.

The main difficulties to build up this knowledge base are linked to the many parameters for the low-level implementations. Some parameters are linked to the information system. For example, a buffer-overflow vulnerability will need different low-level implementations following the CPU architecture, the operating system and its configuration (the presence of ASLR, for example). Some parameters are not constrained by the information system but still must be defined when the action is executed. For example, a buffer-overflow vulnerability in a network service allows the attacker to execute code. Different low-level implementations can realize this high-level action: one can open a shell listening on a tcp port, another one can connect to a tcp port on the attacker computer, and a last one can directly execute the code if the code is small enough to fit in the buffer. These different low-level implementations will have impacts on the detection: in the previous example, the first two implementations will create new network connections between the attacker and the target.

To have a more realistic malicious activity, in our PoC, we added reconnaissance and post-exploitation actions to the attack graph produced by Mulval. In the same way, there are many low-level implementations possible for these high-level actions. For example, listing the user accounts on a Windows computer or on a Linux one is not implemented in the same way. The Att&ck matrix and the red automation tools developed around this matrix can be helpful to build this part of the knowledge base.

Building a large knowledge base of low-level implementations of high-level actions is a tremendous task (certainly more related to engineering than research). However, this knowledge base is necessary to be able to test the monitoring tools in different attack scenarios.

7.3.2 Evolutions to the attack graph

Some evolutions to the attack graph are necessary to automate the execution of the scenarios and to make the malicious activity more realistic.

The attack graph produced by Mulval is a graph that gives the steps that an attacker can implement to achieve his/her goal. That means there can be multiple paths to achieve this goal (depending on the information system, the vulnerabilities and the goal of the attacker). One path must be chosen for the execution of the scenario. Different options are available. The actions can be tagged with some values representing the cost and/or the stealthiness of the actions. The path chosen will be the path minimizing this cost. Another possibility is to choose the path following the actions that a specific group of attackers prefers.

In our PoC, the reconnaissance and post-exploitation actions have been added manually to the attack graph. To fully automate the execution of the scenarios, it is necessary to define what are the reconnaissance and post-exploitation actions and when they will be executed. A first intuition is that the attacker will try the actions whenever he/she will gain some privileges (access to a computer or to a user account with more privileges). Moreover, some post-exploitation actions need some specific privileges and will be executed by the attacker only when he/she gains the required privileges. So, first the properties of the different actions must be defined and then, an algorithm must be developed that outputs the actions to be executed based on their properties and the stage of the attack.

To make the attack scenario more realistic, it is necessary to complement the attack graph with realistic timings between the actions. First, an attacker needs time to process the results of reconnaissance actions and decide what his/her next step will be. Moreover, to remain stealthy, an attacker will not execute all the steps in a hurry and will certainly execute some specific actions at different times of the day. For example, it is stealthier to exfiltrate some amount of data during day work hours.

Another drawback of the attack graph is that the attacker does not do any "useless" exploitations. Since Mulval has all the knowledge of the information system, the attack graph is optimized and an attacker following one path in the graph will not exploit a vulnerability that will not lead to the goal defined. That hypothesis is clearly false in the real world, an attacker will certainly exploit any vulnerability that he/she can find to gain more knowledge about the information system. It is necessary here to model the knowledge of the attacker so as to choose which "useless" vulnerabilities he/she can exploit.

The attack graph is clearly a first step in the automatization of malicious activity. However, it must be complemented with many evolutions so that this activity is more realistic.

7.3.3 Artificial intelligence approach

Another way to generate the malicious activity is to use machine learning techniques. With this kind of technique, it is possible to encode directly in the model all the parameters that need to be defined when using an attack graph. The path that the attacker will follow, the reconnaissance and post-exploitation actions, the timing of the actions and the deviations from the optimal attack path can be directly learned by the model. Another interesting possibility is that a machine learning approach can be used not only for security monitoring evaluation but also for blue team training, since it can react to the remediation actions that the blue team can take.

Since examples of attack scenarios are scarce, reinforcement learning seems to be the only technique available to generate malicious activity. Reinforcement learning is based on one agent that observes its environment, chooses one action to execute and gets a reward based on the result of its action on the environment. This paradigm matches quite well our use case: an attacker will do reconnaissance, exploitation and post-exploitation actions on an information system to achieve his/her goal. The main difficulties with reinforcement learning are the definition of the reward function and the time needed to learn an interesting behavior. One possibility to speed up the learning phase is to combine the reinforcement learning algorithms with expert rules.

8. Conclusion

In order to defend IT systems from potential threats and attacks, security monitoring mechanisms are needed. With continued innovation in the area of intrusion detection, there needs to be a way in which these solutions can be evaluated. However, there’s no clear evaluation methodology which can be used to efficiently achieve the evaluation objectives.

After reviewing the scientific literature on IDS evaluation, we were able to point out a number of design and functional flaws in the datasets available or their generation process, and basically, uncover fundamental drawbacks in relying on static datasets. Our proposal is to create a novel way to carry out IDS evaluation, and not only overcome the most immediate drawbacks of traditional ways, but also have an evolutive platform that allows us to evaluate an IDS in a live manner and to tackle the challenges of reproducibility and representativeness of normal and malicious traffic.

In this work, we have discussed in detail the evaluation metrics that are used in research, we proposed a generic platform designed to bypass the several shortcomings that exist in traditional methods, and we made concrete steps to implement an experiment design. This work also discusses the different avenues for future work in the short, medium, and long term.

References

1. Lincoln Laboratory, MIT. DARPA intrusion detection datasets. accessed May 3rd 2018. https://www.ll.mit.edu/ideval/data/.
2. Nehinbe, J. O. (2011, September). A critical evaluation of datasets for investigating IDSs and IPSs researches. In Cybernetic Intelligent Systems (CIS), 2011 IEEE 10th International Conference on (pp. 92-97). IEEE.
3. Brown, C., Cowperthwaite, A., Hijazi, A., & Somayaji, A. (2009, July). Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on (pp. 1-7). IEEE.
4. K. J. Pickering, Evaluating the viability of intrusion detection system benchmarking, Bachelor Thesis, University of Virginia, US, 2002
5. J. McHugh, Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA IDS evaluations as performed by Lincoln Laboratory, ACM Transactions on Information and System Security, vol.3, No.4, Nov. 2000
6. Lee, W., & Stolfo, S. J. (2000). A framework for constructing features and models for intrusion detection systems. ACM transactions on Information and system security (TiSSEC), 3(4), 227-261.
7. Meena, G., & Choudhary, R. R. (2017, July). A review paper on IDS classification using KDD 99 and NSL KDD dataset in WEKA. In Computer, Communications and Electronics (Comptelix), 2017 International Conference on (pp. 553-558). IEEE.
8. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009, July). A detailed analysis of the KDD CUP 99 data set. In Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on (pp. 1-6). IEEE.
9. Group, T. S. (2000). Defcon 8, 10 and 11.
10. Al-Mamory, S. O., & Zhang, H. (2009). IDS alerts correlation using grammar-based approach. Journal in computer virology, 5(4), 271.
11. Nehinbe, J. O. (2009, September). A simple method for improving intrusion detections in corporate networks. In International Conference on Information Security and Digital Forensics (pp. 111-122). Springer, Berlin, Heidelberg.
12. Center for Applied Internet Data Analysis. “CAIDA Data - Overview of Datasets, Monitors, and Reports.” CAIDA, www.caida.org/data/overview/.
13. Proebstel, E. P. (2008). Characterizing and Improving Distributed Network-based Intrusion Detection Systems (NIDS): Timestamp Synchronization and Sampled Traffic (Doctoral dissertation, University of California, Davis).
14. LBNL/ICSI Enterprise Tracing Project - Traces Project Papers, www.icir.org/enterprise-tracing/papers.html.
15. Sangster, B., O'Connor, T. J., Cook, T., Fanelli, R., Dean, E., Morrell, C., & Conti, G. J. (2009, August). Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In CSET.

16. Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., & Nakao, K. (2011, April). Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation. In Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (pp. 29-36). ACM.
17. Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection." computers & security 31, no. 3 (2012): 357-374.
18. Sharafaldin, I., Gharib, A., Lashkari, A. H., & Ghorbani, A. A. (2017). Towards a Reliable Intrusion Detection Benchmark Dataset. Software Networking, 2017(1), 177-200.
19. Creech, G., & Hu, J. (2013, April). Generation of a new IDS test dataset: Time to retire the KDD collection. In Wireless Communications and Networking Conference (WCNC), 2013 IEEE (pp. 4487-4492). IEEE.
20. Moustafa, N., & Slay, J. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Military Communications and Information Systems Conference (MilCIS), 2015 (pp. 1-6). IEEE.
21. Haider, W., Hu, J., Slay, J., Turnbull, B. P., & Xie, Y. (2017). Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. Journal of Network and Computer Applications, 87, 185-192.
22. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., & Hotho, A. (2017, June). Flow-based benchmark data sets for intrusion detection. In Proceedings of the 16th European Conference on Cyber Warfare and Security (pp. 361-369).
23. I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., in: ICISSP, 2018, pp. 108–116.
24. John E. Gaffney and Jacob W. Ulvila. 2001. Evaluation of intrusion detectors: A decision theory approach. In Proceedings of the 2001 IEEE Symposium on Security and Privacy. 50–61.
25. Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., ... & Mornet, G. (2006). Grid'5000: A large scale and highly reconfigurable experimental grid testbed. The International Journal of High Performance Computing Applications, 20(4), 481-494.
26. Mirkovic, J., Benzel, T. V., Faber, T., Braden, R., Wroclawski, J. T., & Schwab, S. (2010, November). The DETER project: Advancing the science of cyber security experimentation and test. In 2010 IEEE International Conference on Technologies for Homeland Security (HST) (pp. 1-7). IEEE.
27. Botta, A., Dainotti, A., & Pescapé, A. (2012). A tool for the generation of realistic network workload for emerging networking scenarios. Computer Networks, 56(15), 3531-3547.
28. http://rude.sourceforge.net/
29. http://cs.itd.nrl.navy.mil/work/mgen/
30. A. Botta, A. Dainotti, A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," Computer Networks (Elsevier), 2012, Volume 56, Issue 15, pp 3531-3547.
31. http://code.google.com/p/brute/
32. http://code.google.com/p/ostinato/
33. http://gull.sourceforge.net/#Open+Source
34. M.C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, F.D. Smith, Tmix: a tool for generating realistic TCP application workloads in ns2, ACM SIGCOMM Computer Communication Review (CCR) 36 (3) (2006) 67–76.
35. Sommers, J., & Barford, P. (2004, October). Self-configuring network traffic generation. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (pp. 68-81). ACM.
36. http://caia.swin.edu.au/genius/tools/kute/
37. Rolland, C., Ridoux, J., & Baynat, B. (2007, May). Catching IP traffic burstiness with a lightweight generator. In International Conference on Research in Networking (pp. 924-934). Springer, Berlin, Heidelberg.
38. Vishwanath, K. V., & Vahdat, A. (2009). Swing: Realistic and responsive network traffic generation. IEEE/ACM Transactions on Networking (TON), 17(3), 712-725.
39. http://www.ittc.ku.edu/netspec/

40. Bartlett, G., & Mirkovic, J. (2015, June). Expressing different traffic models using the LegoTG framework. In 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops (pp. 56-63). IEEE.
41. http://sourceforge.net/projects/iperf/
42. Feng, W. C., Goel, A., Bezzaz, A., Feng, W. C., & Walpole, J. (2003, August). TCPivo: A high-performance packet replay engine. In Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research (pp. 57-64). ACM.
43. http://tcpreplay.appneta.com/
44. Wong, F. S. Y. (2003). TCPOpera: A Software Tool to Actively Manipulate Recorded TCP Traffic. University of California, Davis.
45. Khayari, R. E. A., Rucker, M., Lehmann, A., & Musovic, A. (2008, June). Parasyntg: A parameterized synthetic trace generator for representation of www traffic. In 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (pp. 317-323). IEEE.
46. Kolesnikov, A. W., & Wolfinger, B. E. (2011). Web workload generation according to the UniLoG approach. In 17th GI/ITG Conference on Communication in Distributed Systems (KiVS 2011). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
47. Barford, P., & Crovella, M. (1998). Generating representative web workloads for network and server performance evaluation. ACM SIGMETRICS Performance Evaluation Review, 26(1), 151-160.
48. Megyesi, P., Szabó, G., & Molnár, S. (2015). User behavior based traffic emulator: A framework for generating test data for DPI tools. Computer Networks, 92, 41-54.
49. Wright, C. V., Connelly, C., Braje, T., Rabek, J. C., Rossey, L. M., & Cunningham, R. K. (2010, September). Generating client workloads and high-fidelity network traffic for controllable, repeatable experiments in computer security. In International Workshop on Recent Advances in Intrusion Detection (pp. 218-237). Springer, Berlin, Heidelberg.
50. https://pyautogui.readthedocs.io/en/latest/
51. http://www.autoitscript.com/site/autoit/
52. http://www.autohotkey.com/
53. R. S. Sutton and A. G. Barto. Introduction to reinforcement learning, volume 135. MIT press Cambridge, 1998.
54. X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In Proceedings of the 13th ACM conference on Computer and communications security, pages 336–345. ACM, 2006.
55. Sembiring, J., Ramadhan, M., Gondokaryono, Y. S., & Arman, A. A. (2015). Network security risk analysis using improved MulVAL Bayesian attack graphs. International Journal on Electrical Engineering and Informatics, 7(4), 735.
56. Ou, X., Govindavajhala, S., & Appel, A. W. (2005, July). MulVAL: A Logic-based Network Security Analyzer. In USENIX security symposium (Vol. 8, pp. 113-128).
57. http://sikulix.com/
58. https://github.com/autokey/autokey
59. Ring, M., Schlör, D., Landes, D., & Hotho, A. (2019). Flow-based network traffic generation using generative adversarial networks. Computers & Security, 82, 156-172.

Date et signature du post-doctorant

12/11/2019

[illegible]

Date et signature de la direction de thèse