On generating network traffic datasets with synthetic attacks for intrusion detection

CARLOS GARCIA CORDERO, Technische Universität Darmstadt
EMMANOUIL VASILOMANOLAKIS, Aalborg University
AIDMAR WAINAKH and MAX MÜHLHÄUSER, Technische Universität Darmstadt
SIMIN NADJM-TEHRANI, Linköping University

Most research in the area of intrusion detection requires datasets to develop, evaluate or compare systems in one way or another. In this field, however, finding suitable datasets is a challenge on to itself. Most publicly available datasets have negative qualities that limit their usefulness. In this article, we propose ID2T (Intrusion Detection Dataset Toolkit) to tackle this problem. ID2T facilitates the creation of labeled datasets by injecting synthetic attacks into background traffic. The injected synthetic attacks blend themselves with the background traffic by mimicking the background traffic's properties to eliminate any trace of ID2T's usage.

This work has three core contribution areas. First, we present a comprehensive survey on intrusion detection datasets. In the survey, we propose a classification to group the negative qualities we found in the datasets. Second, the architecture of ID2T is revised, improved and expanded. The architectural changes enable ID2T to inject recent and advanced attacks such as the widespread EternalBlue exploit or botnet communication patterns. The toolkit's new functionality provides a set of tests, known as TIDED (Testing Intrusion Detection Datasets), that help identify potential defects in the background traffic into which attacks are injected. Third, we illustrate how ID2T is used in different use-case scenarios to evaluate the performance of anomaly and signature-based intrusion detection systems in a reproducible manner. ID2T is open source software and is made available to the community to expand its arsenal of attacks and capabilities.

CCS Concepts: •**Security and privacy → Intrusion detection systems;** *Network security;*

Additional Key Words and Phrases: intrusion detection systems, datasets, attack injection, synthetic dataset

ACM Reference format:
Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Aidmar Wainakh, Max Mühlhäuser, and Simin Nadjm-Tehrani. 2018. On generating network traffic datasets with synthetic attacks for intrusion detection. 0, 0, Article 39 (June 2018), 31 pages.
DOI: 0000001.0000001

1 INTRODUCTION

Evaluating the detection capabilities of NIDSs (Network Intrusion Detection Systems) has become a crucial task in today's Internet age [36]. It is not only due to our dependency on the Internet, but also because of the Internet's threat landscape that NIDSs have become an almost mandatory line of defense against attacks. This need to develop NIDSs that can keep up with evolving attacks and motivated adversaries has yielded much research in the direction of identifying old and new, previously unobserved, threats. Evaluating NIDS has intrinsic complexities and challenges that need to be addressed irrespective of which intrusion detection method they use. The evaluation of

NIDSs relies on quality datasets to assess their capabilities. Quality datasets are also what enables accurate comparison of different intrusion detection methods. Reliable datasets, however, are not readily available.

Reliable datasets useful for the evaluation of NIDSs are hard to obtain. Widely available datasets tend to be outdated, often lack labeled attacks and usually contain overlooked defects. Furthermore, these datasets are often not publicly available or difficult to obtain [18]. Most datasets are distributed as PCAP (Packet Capture) files. These files, being in essence just an ordered collection of network packets, can be thought as of having packets originating from one of two sources: a benign or a malicious one. Packets originating from benign sources compose the *normal* or *background* traffic of the dataset. Packets created by malicious activities on a network compose the *attack* traffic of the dataset. To overcome the seemingly inherent difficulties of creating and sharing datasets, we have developed ID2T. ID2T is a tool that creates and injects synthetic attack traffic into background traffic. Rather that just naïvely merging together the background and attack traffic, ID2T attempts to replicate the properties of the background traffic in the synthetic attack traffic. The level by which properties are replicated is controlled by the user to suite their needs.

Users of ID2T are not confined to only injecting the set of attacks provided. The instruments to develop new attacks are provided. These come in the form of an API (Application Programming Interface) to easily extract, compare and replicate traffic properties.

The design philosophy of ID2T is simple: background traffic is provided by the user and ID2T adds attacks to the background traffic following the specifications of the user. The injected attacks are labeled and clearly identified for intrusion detection mechanisms to use; for instance, for evaluation purposes. Figure 1 shows the main architectural components that together interact to create datasets with synthetic attacks that replicate background traffic's properties. The inputs of ID2T are a PCAP file containing background data and user supplied parameters. Its output is a PCAP file injected with synthetic attacks, a labels file specifying the time and location of the injected attacks, and a report on the statistical analysis of the background traffic.

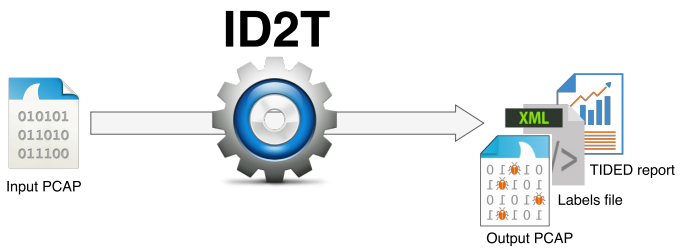


Figure 1. ID2T provides all what is needed to create datasets that contain attacks that replicate the properties of arbitrary background traffic. The output dataset, in the form of a PCAP file, contains labeled attacks that suite the needs of the user.

1.1 Contributions

This work substantially extends two earlier publications [7, 35]. The first publication presents the idea of ID2T in the form of a poster [7]. In the poster, we presented a basic architecture for injecting attacks that replicate background traffic's properties along with the basic requirements to do so. In the second work, we formalized the requirements for injecting attacks, we built a prototype, and we evaluated the prototype's performance and ability to inject attacks [35]. This work has the following additional contributions:

- We present a comprehensive survey of NIDS datasets and synthetic dataset generation tools as well as a categorization of the problems found in these.
- We release ID2T as an open source[1] software to help NIDS researchers to create, distribute and replicate datasets.
- We develop the ID2T module named TIDED that calculates quantitative characteristics of network traffic that help researchers determine whether background network traffic has abnormal characteristics.
- We implement a number of recent attacks. For instance, these include the popular Wannacry attack, based on the *EternalBlue* exploit, as an injectable attack within ID2T. This attack highlights how synthetic attacks can be easily created in contrast to generating and publishing whole new datasets that contain the same attack. We further demonstrate use-case scenarios that show how ID2T can be used to evaluate NIDSs in a reproducible way.

1.2 The Limitations of ID2T

ID2T limits itself to replicating the properties of user-supplied background traffic into synthetically generated attack traffic. Many attack scenarios are not affected by this limitation, others, however, may be negatively affected. If an attack is not expected to alter the state of a network, the replication strategy employed by ID2T is sufficient. In this category of attacks we find most exploit attempts and network reconnaissance scans. In contrast, if an attack is expected to change how packets are produced and distributed in a network, ID2T will only approximate the real effects of the attack. Most denial of service and botnet attacks fall into this category.

A second limitation exists that is related to the labeling of attack traffic. A dataset suitable for evaluating NIDS needs to have labeled attacks. ID2T indeed labels all synthetically injected attacks. However, if the provided background traffic contains attacks, not all attacks would be labeled. To try and leverage this problem, ID2T analyses the background traffic provided by the user so as to highlight or make existing attacks stand out.

1.3 Outline

The remainder of this article is structured as follows. In Section 2, we propose a set of requirements for datasets that are suitable for the evaluation of NIDSs as well as for injecting synthetic attacks. Section 3 discusses the state of the art in static and dynamic datasets for the evaluation of NIDS and algorithms. We further categorize some of the defects found in these datasets. With Section 4 we begin the analysis of our toolkit (ID2T), its architecture and components. In addition, Section 5 touches one important component of ID2T that is responsible for the testing and analysis of intrusion detection datasets. Subsequently, the attacks that can be generated by our toolkit are presented in Section 6. Section 7, shows use cases that demonstrate how ID2T can be applied to evaluate anomaly and signature-based NIDSs. Finally, Section 8 concludes this article.

2 REQUIREMENTS

Many requirements concern the creation and injection of synthetic attacks. On the one hand, there are requirements related to the tool that injects the attacks. On the other hand, there are requirements inherent to the resulting datasets, containing injected attacks, that are meant to be used for the evaluation of IDSs (Intrusion Detection Systems). This section emphasizes this distinction and divides the requirements into functional and non-functional ones.

[1]The source code of ID2T can be found in https://github.com/tklab-tud/ID2T.

2.1 Requirements of Datasets Suitable for the Evaluation of IDSs

Irrespective of whether datasets contain synthetic or real data, they need to conform to certain requirements if they are to be used for the evaluation of NIDSs. From observations made in our research in the field of IDS (i.e., [7, 34–37]), as well as from the research of others [3, 18, 22, 29], we have derived different requirements that make datasets usable in the context of evaluating NIDSs. In the following, these requirements are split into functional and non-functional ones.

2.1.1 Functional Requirements. These functional requirements focus on what is needed to construct or assemble datasets that enable different NIDSs to compare their performance against each other.

(1) Payload availability — Due to privacy, network payloads are often unavailable. This limits the capability of NIDSs to detect attacks at the payload level (e,g., application layer exploits). To serve NIDSs that aim at detecting attacks at the payload level, one needs datasets that include payloads.
(2) Labeled attacks — Datasets need labels that distinguish malicious from benign network traffic. Labels enable NIDSs to determine their detection accuracy and to establish a direct comparison against other NIDSs.
(3) Ground truth — Labeled traffic is not enough if it cannot be guaranteed that the background traffic does not contain unlabeled attacks. Without ground truth, a direct comparison between different NIDSs is not possible.
(4) Growing — A growing dataset is one that is constantly updated with traces of recent network traffic patterns and attacks.
(5) Attack diversity — Datasets usually focus on certain types of attacks. A dataset suitable for the evaluation of different NIDSs requires a diversified set of network attacks, from DoSs (Denial of Services) to remote exploitation attempts.

2.1.2 Non-Functional Requirements. The following non-functional requirements specify the criteria datasets need to satisfy to be of practical use.

(1) Public availability or ease of reproducibility — Although being obvious, this requirement needs to be explicitly stated. Datasets need to be created with the goal of becoming public or easily reproduced. If a dataset is used for multiple evaluations, for the purpose of replicating experiments, it must be made public. If privacy is a concern, anonymizing the data should be carried out carefully so as to not introduce *artifacts* (see Section 3.3 for a definition of the term). If the dataset cannot be public, reproducing it must be possible.
(2) Interoperability — Network data needs to be shared using a common format. Although not a standard, the most commonly used format is the PCAP file format.
(3) Good quality — Through our empirical experience of generating synthetic attacks, as well as through the analysis of related work (see Section 3), we have identified diverse quality issues that need to be taken into account when synthetic attacks are injected into real traffic. These quality issues need to be addressed one by one, each one requiring special attention. In Section 3, we further discuss these issues (termed *artifacts*) and categorize all observations of these in other datasets.

2.2 Requirements for Injecting Synthetic Attacks

Tools for injecting synthetic attacks need to be built with the goal of crating useful datasets for the evaluation of IDSs. These tools must be capable of working with arbitrary PCAP files; the de facto standard for recording and sharing network communications. There should be no hard assumptions regarding the provenance of these files nor how they are generated. It is for this

reason that considerable care needs to be taken when designing a tool that injects synthetic attacks into PCAP files.

2.2.1 Functional Requirements. The following functional requirements have been derived from observations of how datasets are manually created. To alleviate the manual task of creating tailor-made labeled datasets with attacks, a tool that injects attacks needs to take several functional requirements into account. These next requirements focus on conferring tools with the ability to customize already existing attacks and create new ones without much additional effort.

(1) Packet level injection — In order to evaluate any type of network IDS, attacks must be injected at the lowest common denominator that all network IDSs can use, i.e., packets. Many IDSs directly use packets to perform intrusion detection. There is, however, another family of IDSs that work on higher abstraction levels, such as network flows [3]. If attacks are injected at the packet level, both types of IDSs can benefit from the generated datasets.
(2) Querying interface for network properties — In order the generate realistic-looking synthetic attacks, an input PCAP file needs to be analyzed to extract properties. These properties amount to quantitative characteristics such as *average used bandwidth per host*, *total number of packets sent by a specific host*, or *the open ports observed for a given subnetwork*. These properties need to be made available to users that wish to inject attacks as well as to the program that creates the synthetic attacks. An API is required to directly query the input files into which attacks will be injected. This enables, for example, a synthetic port scan to determine which ports, for a given host, were open in the input PCAP file.
(3) Attack diversity — Many network attack classifications exist. These classifications should be taken into account when providing tools for generating and injecting synthetic attacks. Rather than only focusing on a single type of attack, creating a diverse set of attacks is required.
(4) Applicability to arbitrary PCAP files — As long as a PCAP file contains network traffic alone, the file should be suitable for injecting synthetic attacks. To achieve this requirement, PCAP files need to be analyzed to replicate their properties onto injected attacks. For instance, if a network only contains local traffic (from the 192.168.1.0 subnet), by default attacks should not originate from IP addresses outside this subnet (unless this is desired and manually configured).
(5) Headless operation — Working with arbitrary PCAP files implies that their size is unbounded. Packet captures of crowded networks can easily reach a size of hundreds of gigabytes. Analyzing and injecting attacks into these files might take considerable time and memory resources. It is assumed, therefore, that cluster environments or headless systems are used to process large files. For this reason, tools that inject attacks need to work in headless environments with small or no user interaction.
(6) Modeling of packet behavior and payload — Attack injection tools need to model attacks at the packet and payload levels. Attacks at the packet level are more concerned about the quantity of the sent packets rather than on the payload. Port scans and DDoS (Distributed Denial of Service) attacks are examples of these. In contrast, attacks at the payload level may carefully craft the data sent by each packet. Exploit attacks against a web server are examples of these. Some attacks are concerned with both levels, such as when creating synthetic botnet generated traffic. This traffic sometimes contains important payload information that defines the attack, other times, the traffic is used as a DDoS attack in which the payload is not important.

2.2.2 Non-Functional Requirements. The following non-functional requirements are needed to enable synthetic attack injection tools to process arbitrary PCAP files. The main challenge when not being limited to certain PCAP files is to consider the different scenarios from where these files may originate. Home and office environments generate limited traffic belonging to only few sub-networks. In contrast, large enterprise or university networks may generate massive quantities of data with heterogeneous characteristics.

(1) Scalability — When mimicking the properties of network data onto a synthetic attack, many different statistics need to be collected. Calculating statistics on top of network data is typically demanding both in terms of memory and processing resources. Tools that inject synthetic attacks need to take this into account and calculate statistics as efficiently as possible. This guarantees the scalability of the tool when large PCAP files are analyzed and injected with synthetic attacks.
(2) Extensibility — New attacks are developed each day. A set of tools and libraries are needed to easily and rapidly model these new attacks for injection.
(3) Usability — One of the main reasons static datasets are widely used is due to their ease of use. Static datasets have many disadvantages: they never change and become easily outdated. Their advantages, however, outweigh the disadvantages: they can be used immediately without extensive setup. Tools that generate either real or synthetic traffic often require complicated hardware and setups. To be as useful as static datasets, tools that inject synthetic attacks need to be usable. The user should be involved as little as possible in the injection process but have a detailed control over the injections.
(4) Open source and public availability — Researchers often build custom tools to evaluate their work without making the tools available in open platforms. It is also common that tools are lost when researchers change subject areas or institutions. For example, the tools proposed by Shiravi et al. [29] and Brauckhoff et al. [4] can no longer be easily found online although they used to be available.

3 RELATED WORK AND DEFECT ANALYSIS

A good quality dataset is one that allows researchers to identify the ability of an IDS to detect anomalies [3], preferably allowing to draw valid conclusions about the appropriateness of the IDS (its efficiency, accuracy, validity scope, etc). Although many of the available datasets nowadays are valuable to the research community, they unfortunately fail to fulfill all the requirements proposed in Section 2. The existing datasets can be classified into two categories: *static* datasets and *dynamically generated* datasets (see Figure 2 for a historical overview). In this section, first, we provide basic information about some of the most popular static datasets. Afterwards, we present a description about the dynamically generated datasets. Lastly, this section discusses the deficiencies and artifacts that can be found in datasets.

3.1 Static Datasets

A static dataset is a dataset that was generated once during a limited period of time. Such a dataset can be collected from a real-world network or it can be generated synthetically. Currently, there are several static datasets available for IDS evaluation purposes. Unfortunately, research suggests that these datasets are not adequate for mainly two reasons. First, many of them are out of date. They were created many years ago based on old versions of protocols, old applications, and attacks that exploit old vulnerabilities. Thus, these datasets are not realistic datasets at the present time. Second, these datasets contain many defects that indicate synthetic generation. In the following, we briefly discuss some well-known static datasets. It is worth to mention that similar datasets can be

found in other fields, such as the ICS (Industrial Control Systems) field, where building testbeds is a challenging task [15, 31]. However, the testbeds in such fields are built under application-specific conditions, and are therefore considered out of the scope of this article.

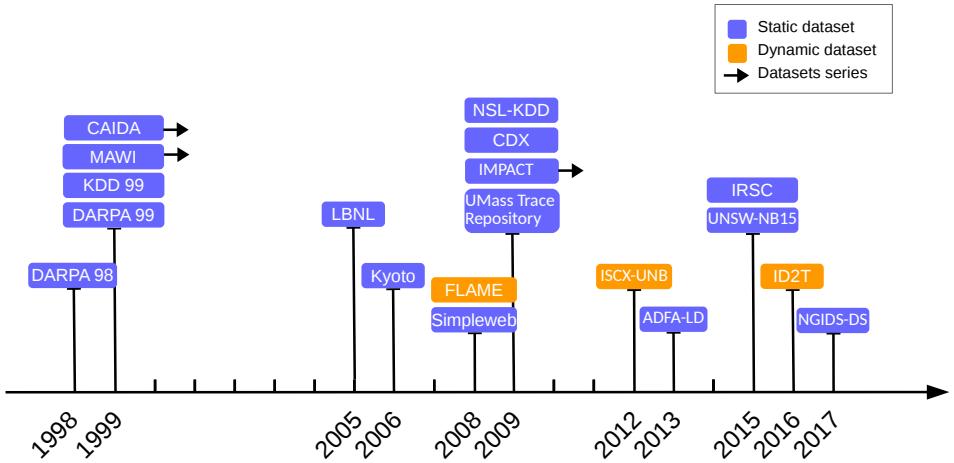


Figure 2. A historical timeline overview of datasets intended for intrusion detection.

3.1.1 DARPA 99. The DARPA 99 is one of the most well-known datasets in the field. In fact, DARPA 99 is an upgraded version of the DARPA 98 dataset. The two datasets were collected during two intrusion detection evaluations [9, 20]. A testbed was developed to generate normal and malicious traffic. The normal traffic tended to be similar to that seen between a military base and the Internet. A custom software automaton was used to simulate hundreds of users running UNIX applications. In DARPA 98, the included attacks cover four categories, namely, DoS, R2L (Remote to Local), U2R (User to Root), and Surveillance/Probing. The traffic was collected for seven weeks with labeled attacks and two weeks with unlabeled attacks. In DARPA 99, Windows NT hosts were included in the evaluation as victims and more attacks types were covered. The dataset consists of two weeks of normal traffic with no attacks, one week with a few labeled attacks, and two weeks of unlabeled traffic. DARPA 98 – 99 datasets were highly innovative for their time and had been widely used for IDS evaluation. However, later on, several studies have marked these datasets as unreliable due to a number of limitations and issues (see Section 3.3).

3.1.2 KDD 99. This dataset was created on the basis of the DARPA 98 during a competition for network intrusion detectors [10]. KDD 99 contains data records, rather than packets, where each record contains 31 features that describe a connection. A connection is a sequence of TCP packets that flow between two defined IP addresses, under defined protocol and within defined time interval. The raw data of DARPA 98 was used to extract seven million connection records. All connections were labeled as either normal or malicious with one specific attack type. This dataset was used heavily to evaluate IDSs at that time. Later on, several shortcomings in this dataset were pointed out by various studies.

3.1.3 MAWI. The MAWI (Measurement and Analysis on the WIDE Internet) dataset corresponds to a family of datasets that are collected from an operational testbed network, connecting Japanese universities and research institutes with various networks all over the world [30]. The traffic is stored in the *tcpdump* raw format, so the header information is available and can be used for

further analysis. However, for privacy reasons, IP addresses are scrambled and packets' payloads are removed. Unfortunately, MAWI datasets are not labeled, thus, there is no ground truth to be used in IDS evaluation. Fontugne et al. [12] proposed a method to label the anomalies in a subset of MAWI datasets. The labeled dataset is known as MAWILab dataset. The labels are obtained using an advanced graph-based methodology that compares and combines several independent anomaly detectors. The dataset is updated constantly to include new traffic and anomalies from upcoming applications. Nevertheless, as the detectors are anomaly-based, false positives are expected to be present in the dataset.

3.1.4 NSL-KDD. Tavallaee et al. [32] conducted a statistical analysis on the KDD 99 dataset and found two deficiencies. First, more than 75% of the data records are duplicated, which makes the learning algorithms less sensitive for infrequent records, which can be malicious. Second deficiency is the low level of difficulty. Using typical classifiers, the authors were able to label the dataset correctly with an unreasonably high accuracy rate. In NSL-KDD, the duplicated records were removed and the difficulty level was increased by selecting records out of KDD 99 based on the inverse of classification accuracy rate.

3.1.5 CDX. This dataset was collected from the CDX (Cyber Defense Exercise) 2009 competition [28]. The knowledge about the locations of the defenders and the attackers in the network was used to label the traffic as normal or malicious. In contrast to the DARPA 98 – 99, the CDX dataset is more realistic, since it presents the live traffic of real human activities. Nevertheless, the ratio between the normal and malicious traffic was almost equal, not representing real world observations. In addition, the dataset has a small volume because of the limited duration of the competition. This is problematic for anomaly detectors that require a long training period.

3.1.6 Other Datasets. In the following, we briefly discuss other IDS public datasets. An overview of static datasets and their properties can be found in Table 1 [2, 18, 28, 29, 38].

- **CAIDA**: CAIDA (Cooperative Association for Internet Data Analysis) collects data at different locations, and provides this data to the research community, taking into account the privacy of individuals and organizations who participate in generating the data [5].
- **LBNL**: LBNL (Lawrence Berkeley National Laboratory) collected packet traces for more than 100 hours of internal enterprise traffic, and released this data publicly in an anonymized form [19].
- **Simpleweb**: A dataset was collected, Twente university network, via a honeypot in September 2008. Several network services were hosted on that honeypot, which was directly connected to the Internet. The honeypot only captured suspicious traffic [2].
- **IMPACT**: The IMPACT (Information Marketplace for Policy and Analysis of Cyber-risk & Trust) project provides an open platform for dataset exchange. This platform connects producers and consumers of cyber-risk-relevant data. The existing data was collected by various organizations and universities [16].
- **UMass Trace Repository**: UMass university provides a trace repository that contains network, storage, and other data traces. The data was extracted during various experiments and hence is specialized; it reflects specific network behavior and attacks that were captured for experimental purposes [33].
- **ADFA-LD**: This dataset is provided by the university of New South Wale and consists of normal and malicious Linux based system call traces. Only host logs had been collected [14].
- **IRSC**: The dataset, created by the IRSC (Indian River State College), consists of packet captures and network flow data and includes labels [38].

- **UNSW-NB15**: This dataset contains real modern normal traffic and synthetic attacks. It was specifically generated over a commercial penetration testing environment [14, 24].

Dataset	Availability	Synthetic	Payload	Ground truth	Labeled attacks	Updates
DARPA 98-99	✓	✓	✓	✓	✓	x
KDD 99	✓	✓	✓	✓	✓	x
MAWILab	✓	x	x	x	✓	✓
CAIDA	✓[1]	x	✓	x	x	✓
SimpleWeb	✓	x	x	x	x	x
NSL-KDD	✓	✓	✓	✓	✓	x
CDX	x	✓	✓	✓	✓	x
IMPACT	✓[2]	✓	x	x	x	✓
UMass	✓	✓[3]	✓	x	x	x
RIPE	✓[4]	x	✓[5]	x	x	✓[6]
IRSC	x	✓	✓	✓	✓	x
UNSW-NB15	✓	✓[7]	✓	✓	✓	x

1 Access to some datasets requires special permissions.
2 Access is limited to U.S. researchers.
3 A variety of datasets, most of them synthetic.
4 Registration is required.
5 Not available in all datasets.
6 Not updated regularly.
7 Real benign traffic and synthetic attacks.

Table 1. Comparison of static datasets.

3.2 Dynamically Generated Datasets

Network behavior and traffic patterns change continuously and attacks evolve rapidly. This created a need for datasets that are modifiable, extensible, and reproducible. Therefore, researchers have proposed tools that are capable of generating synthetic datasets dynamically. The main idea of these tools is to consider the contemporary traffic characteristics and replicate them in synthetic traffic, which in turn emphasizes the realism of the generated datasets. In this section, we present the two existing dynamic datasets, followed by a brief comparison of them in Table 2.

3.2.1 FLAME. Bauckhoff et al. [4] proposed FLAME, an application that generates and injects parameterized anomalies into a given flow trace. Three classes of anomalies are offered by FLAME: (*i*) Additive anomalies where synthetic flows are added to a background trace, but without interacting with the existing flows, e.g., network scans; (*ii*) Subtractive anomalies in which selected flows are removed from background traffic, e.g., ingress shifts; (*iii*) Interactive anomalies where synthetic flows are added to a background trace, and selected flows are removed, e.g., DoS attacks that cause a network congestion. In particular, three use cases were implemented: a TCP SYN network scan, ingress shifting, and TCP SYN DoS attack.

The usage of network flows, rather than payloads, as background traffic makes datasets generated by FLAME usable only for flow-based intrusion detection algorithms. In addition, many attacks cannot be detected by only using network flows. In this context, FLAME is capable of creating a limited range of attacks which have a flow footprint. Moreover, the implementation of FLAME is not publicly available anymore due to its discontinuation.

3.2.2 ISCX-UNB. Shiravi et al. [29] proposed a mechanism to generate dynamic datasets based on the concept of profiles (i.e., abstract representations of certain features and events in network traffic). Two classes of profiles were proposed, namely, α and β. The α profiles describe attack behavior and are used to generate malicious traffic. The β profiles represent legitimate agents' activities and are used to generate normal traffic.

Real network traffic was analyzed to extract mathematical distributions of various properties, such as packet sizes, sizes of payloads, etc. These distributions were encapsulated in β profiles to create models of normal traffic. Several β profiles were created to simulate different legitimate agents that use various network protocols. The α profiles were created manually based on human knowledge. Each profile represents an attack scenario and a number of attacks were implemented. A testbed network was built and a combination of α and β profiles were executed to generate traffic.

The quality, of the profile models in this work, however, raises questions. The β profiles were created based on real traffic that was not verified to be free of attacks, and this degrades the confidence degree in the correctness of the profiles. Moreover, human assistance is required to execute α profiles, which leads to variances on how attacks are carried out. Thus, the reproducibility of the generated dataset is reduced. Lastly, the authors provide only the generated dataset to the research community, but not the implemented profiles.

Dataset	Availability	Synthetic	Payload	Ground truth	Labeled attacks	Tool updates
FLAME	x	✓	x	x[1]	✓	x
ISCX-UNB	x[2]	✓	✓	✓	✓	x

1 Users are responsible for providing benign traffic as an input.
2 One output dataset is available but not the tool.

Table 2. Comparison of dynamic datasets.

3.3 Defect Classification

In this section, we expose and categorize defects, that we as well as other researchers have detected. ID2T actively tries to avoid committing the same mistakes discovered in other datasets, either static or dynamic. We use the term *deficiency* to refer to a problem that appears in a static dataset. In contrast, we use the term *artifact* to indicate a problem that has appeared as a side-effect of creating synthetic network traffic. *Artifacts* are associated to dynamic datasets only. Synthetic data is any data that is not acquired through direct measurements [25]. In the context of networks, synthetic data refers to network traffic that is not captured directly from real-world networks, instead, it is generated by software to mimic real-world traffic.

3.3.1 Categorization of Dataset Defects. We have categorized the dataset defects according to Figure 3. Dataset defects can be divided in two classes, namely *Invalid Network Traffic* and *Inconsistent Network Traffic*. Invalid Network Traffic defects are problems that relate to the incorrect usage of network protocols or specifications. Inconsistent Network Traffic defects are those pertaining the creation of inadvertent traffic patterns or network packets. These two defect classes are characterized by different symptoms depending on two different perspectives. From a *Network Biased Perspective*, certain traffic presents defects only when the background traffic is taken into account; otherwise, it looks normal. From a *Network Agnostic Perspective*, traffic patterns present defects regardless of the background traffic. In the following, each piece of the classification is explained and the defects found are classified.

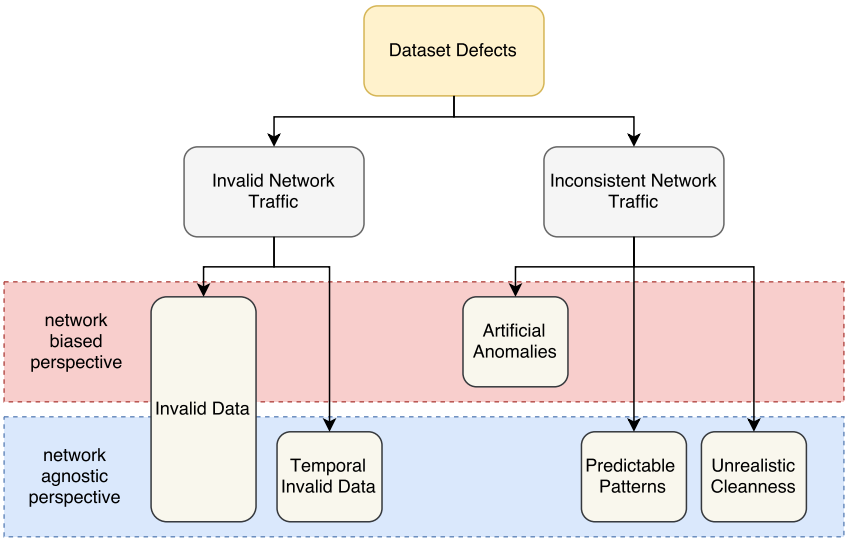


Figure 3. Categories of defects found in static or dynamic datasets.

(1) **Invalid Network Traffic**: Traffic patterns or individual packets that are invalid because they contradict a network protocol specification. We encountered two reasons concerning these class of defects, namely *Invalid Data* and *Temporary Invalid Data*.
   (a) **Invalid Data**: Invalid traffic according to a protocol specification. These defects have a slight change of meaning depending on the perspective. From a network specific perspective, traffic data in a dataset is invalid if the characteristics and physical limitations of a network (captured by the dataset) are not respected. For example, if packets have a larger MTU (Maximum Transmission Unit) than what the network hardware supports; also, if packets use more bandwidth than what is available. These two examples are cases of *artifacts*. From a network agnostic perspective, traffic data is invalid if protocols are violated that should or cannot be. For example, any TCP packet using port zero.
   (b) **Temporary Invalid Data**: Traffic data that used to be valid according to an old protocol but are not anymore. These defects are only relevant when looked from a network agnostic perspective. An example of such a defect is found in the DARPA 99 dataset [20]. Here, the IPv4 ToS (Type of Service) field uses an old standard found in RFC 791 [27]. The ToS field has a different meaning according to the more recent RFC 3260 [13].

(2) **Inconsistent Network Traffic**: Network traffic that does not reflect the same characteristics of some background network traffic. Inconsistencies can originate from wrong assumptions of how a network behaves or unforeseen problems in the process of generating synthetic traffic.
   (a) **Artificial Anomalies**: Anomalous data patterns inadvertently added to network traces. These can result from either the generation of synthetic attacks or the incorrect set-up of hardware. As an anomalous pattern depends on the specific network, this is a defect found only from a network biased perspective. For example, Mahoney et

al. [21] were able to detect 33 attacks in the DARPA 99 dataset [20] due to the use of the same TTL (Time To Live) values in the packets of all attacks.

(b) **Predictable Patterns**: Patterns that appear repeatedly such that network traffic diversity is constricted, making the traffic incorrectly correlate with bogus information. These defects are not expected in any network and are therefore taken into account from a network agnostic perspective. As an example, Brauckhoff et al. [4] generated synthetic SYN port scans with this defect. The authors, without knowing the repercussion of doing so, used the same delay between the reply packets of the attack. This led to a predictable pattern of inter-arrival times that highly correlate with the attack. Similarly, the DARPA 99 dataset [20] utilized TTL values with only nine different values across the entire dataset, making some values highly correlate with the attacks alone.

(c) **Unrealistic Cleanness**: An excessively clean dataset can be a sign of network traffic that does not have the characteristics of real-world network traffic. Most networks will periodically observe issues such as incorrect packet checksums or TCP packet retransmissions and duplication. From a network agnostic perspective, a complete lack of these issues signals a dataset that does not conform to a real-world and live network.

We now go on to describe our dataset generation framework followed by the tools and applications of it.

4 THE INTRUSION DETECTION DATASET TOOLKIT (ID2T)

The ID2T (Intrusion Detection Dataset Toolkit) is software that generates and injects synthetic attacks into a PCAP file. The main aim of ID2T is to dynamically create high quality datasets for IDS evaluation purposes. ID2T was designed to provide the research community with datasets that meet the aforementioned requirements of IDS datasets (see Section 2.1). At a glance, ID2T takes a PCAP file as input and provides a new PCAP file as output; the latter contains the original input traffic along with synthetic attacks.

The main architecture of ID2T was presented in [7, 35]. In this work, we extend the *Attacks* module to include a wider range of attacks. Generating additional attacks with different characteristics requires considering further statistical properties of the background traffic, therefore, we extend as well the *Statistics* module to cover this aspect. In addition, we develop a new module *TIDED*, which analyzes the input traffic properties in order to indicate potential problems. In this section, we provide a detailed description of ID2T and its extended architecture.

4.1 ID2T Architecture

Figure 4, illustrates the architecture of ID2T in which arrows represent flow of data between the computation modules. In the following, we describe in detail the architecture by first focusing on the input and the output of ID2T and subsequently on the core components of the system.

4.1.1 Input. ID2T receives two inputs from the user: a PCAP file and the attack parameters. As mentioned in Section 1, the main assumption behind ID2T is that the user is responsible for the input PCAP file. That is, the PCAP may or may not contain attack data. In both cases, the toolkit replicates the input traffic characteristics in the synthetic attacks. Hence, ID2T does not examine the input for signs of attack data. It does, however, provide a test module that conducts a quantitative analysis of the input PCAP file. This analysis produces several statistical properties, which can be used to infer potential problems in the input traffic (e.g., incorrect TCP checksums or unexpected values in packet headers). The second input is the attack parameters, which specify the properties of the synthetic attack. Examples of such parameters can be the IP addresses of the

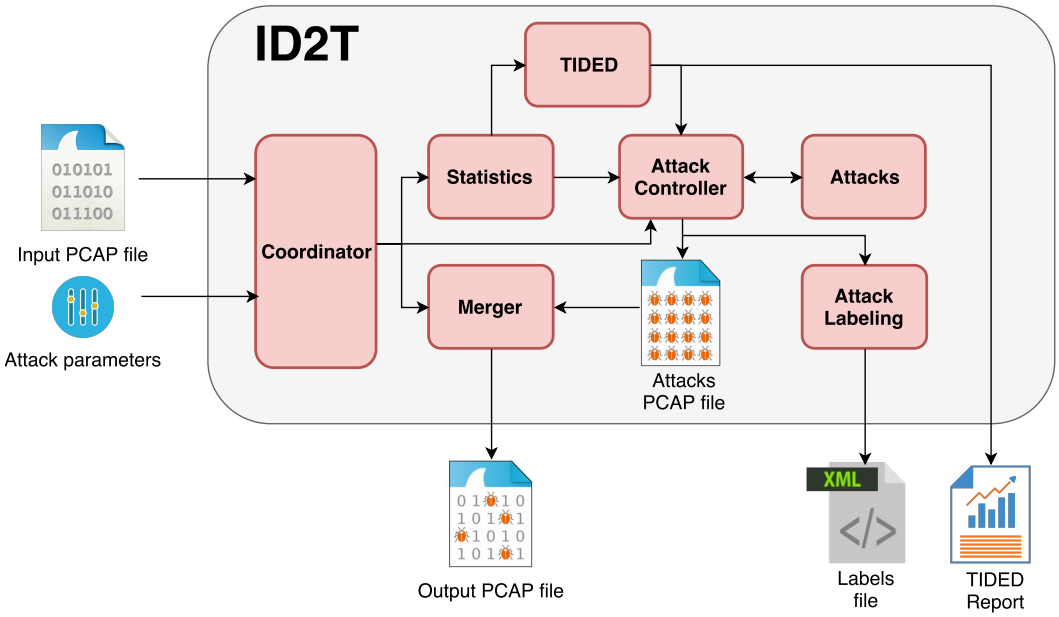


Figure 4. An overview of the ID2T architecture

attacker and the victim, the utilized ports, and the location in which the attack will be injected in the input traffic.

4.1.2 Output. As depicted in Figure 4, the output of the system is threefold. First, ID2T creates a PCAP file that includes the input traffic along with the synthetic attacks. Second, the toolkit can (optionally) perform a multitude of statistical tests to explore the characteristics of the input traffic. The results of the tests are presented in a set of graphs and a text file denoted "TIDED report" in Figure 4. Finally, ID2T generates a labels file that contains information about the injected attacks and their locations in the output PCAP file.

4.1.3 ID2T-Modules. In the following, we present the core modules of the system, namely the *Coordinator*, the *Statistics*, the *Attack Controller*, the *Merger*, the *Attack Labeling* and *TIDED*.

Coordinator. This module initiates the process, triggers other modules, and passes them the required parameters. More specifically, the *Coordinator* receives the path of the input PCAP file and the attack parameters from the user, and forwards these two inputs to the *Statistics* and the *Attack Controller* modules, respectively.

Statistics. This module collects statistical properties of the input PCAP file. The module is also responsible for calculating the hash [2] of the input PCAP, which is used to check whether the file was previously loaded and analyzed. In this case, the recalculation of the file's properties is not required. Otherwise, the *Statistics* module analyzes the packets of the file sequentially and collects four types of features.

First, properties of the file as a whole are collected and taken into account; these can be features such as the total packet count, the capture duration and the average packet size. Second, properties that relate to each host in the network, such as the sent and received packets are determined. Third,

[2]SHA-224 hash function

the distribution of some header fields, e.g., the protocols, the ports , the TTLs, and the window sizes are calculated. Finally, features with regard to the TCP connections, such as the average packet rate and the packet inter-arrival time are taken into account. The aforementioned properties are mainly used by the *Attack Controller* module to guarantee that the synthetic attacks exhibit similar characteristics to the input traffic when this is expected from an attack that would be created in real world. In addition, the statistical properties provide the main input for the traffic analysis that is conducted by the *TIDED* module.

Attack Controller. The main tasks of this module is to validate the attack parameters provided by the user, and to call the corresponding *Attacks* module to generate the synthetic attack. The *Attack Controller* module provides the *Attacks* module with the statistical properties of the input PCAP file. Afterwards, it produces a temporary PCAP file that contains the generated attack.

Attacks. This module contains the attacks that ID2T can generate and inject. This module was designed to be extensible, in the sense that a user can implement and add their own attacks. Currently, the module contains four classes of attacks, namely, probing and surveillance, exploits, resource exhaustion, and botnet infection. For each attack, this module receives and processes a different set of parameters and, in some case, additional input files. An in-depth discussion of the attacks that are supported by ID2T is given in Section 6.

Merger. After generating a temporary attack PCAP file by the *Attack Controller* module, the *Merger* module injects the attacks by merging that file with the input PCAP file according to the timestamps of the packets.

Attack Labeling. The task of this module is to create a file that contains labels of the injected attacks. The *Attack Controller* module sends information about the injected attacks to the *Attack Labeling* module, which in turn writes it in XML format into a text file. The information includes the names of the attacks and the timestamps where they start and end in the output PCAP file.

TIDED. This module provides users with a quantitative analysis of the input traffic characteristics. A detailed description of the *TIDED* module is given in Section 5.

4.2 Modules' Performance

This section presents an overview of the performance of two modules, namely, the *statistics* and the *attacks* modules. Our experiments show that the ability of these modules to generate the internal data needed for ID2T is acceptable from a timing perspective, meaning that no bottlenecks have so far been discovered in running the modules when feeding each other with data. In the following, we show two charts: the statistics generation performance indicator, and the DDoS attack data creation indicator.

a) Generation of Statistics: One of our proposed requirements (see Section 2) is the ability to handle arbitrary files. Hence, it is important to examine how ID2T's architecture handles large files. The only bottleneck to take into account, for handling large files, is located in the *statistics* module. Figure 5 shows the time required to handle files of different sizes. The PCAP files utilized in the experiment are from the MAWI archives [3]. In particular, four different files are examined, ranging from approximately 11 million to 212 million packets. For each file, two different measurements were examined. First, for collecting the so-called regular statistics, which encompass the features that were presented in the *Statistics* module description. Second, measurements are taken for further data analysis. These extra tests include, but are not limited to, calculating IP addresses' entropies,

[3] http://mawi.wide.ad.jp/mawi/samplepoint-F/2015/201506021400.html

examining the correctness of TCP checksums, and checking the availability of packet payloads. These tests are parts of the *TIDED* module, and are further explained in Section 5. At a glance, Figure 5 suggests that moderate network files, e.g., a 14GB PCAP file, can be analyzed by ID2T in a reasonable time frame.

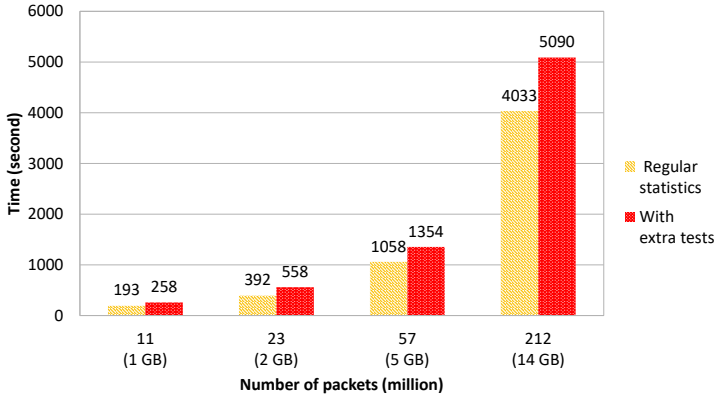


Figure 5. Performance of collecting statistics

b) DDoS Attack Creation: The toolkit provides attacks such as DDoS attack, which usually produces a large number of packets. Therefore, the performance of generating packets is important for such toolkit. Figure 6 shows the results of several performance experiments with regard to the time needed to generate 10^4, 10^5, and 10^6 packets of a DDoS attack. Although the experiments show that the time increases exponentially, it is illustrated that ID2T is able to generate a big number of packets in a reasonable amount of time.

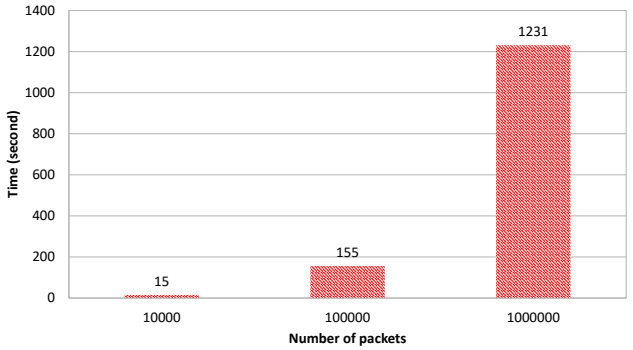


Figure 6. Performance of creating DDoS attack packets

5 TESTING INTRUSION DETECTION DATASETS (TIDED)

Datasets are needed to accurately evaluate the detection capabilities of NIDSs. For evaluations to be accurate and unbiased, datasets need to satisfy the requirements presented in Section 2.1. If these requirements are not satisfied, NIDSs might detect or learn artificial patterns that do not

typically occur in networks. And while most requirements are easily verified, the *Good Quality* non-functional requirement is of a more complex nature. In order to analyze datasets to identify potential *quality* problems, we have developed the ID2T module named TIDED. The goal of TIDED is to identify potential sources of *artifacts* (synthetic defects, see Section 3.3), inherent deficiencies of the background traffic (a user defined PCAP file), or both.

There are two sources of *artifacts* or deficiencies that contribute to the fallibility of a dataset when evaluating NIDSs. The tools that inject synthetic attacks or manipulate traffic are the first source. The second source is the dataset itself and the environment where it is generated. ID2T actively avoids creating *artifacts* throughout the process of injecting synthetic attacks. In spite of these efforts, it is still possible that, after using ID2T, a generated dataset is unsuitable for evaluation because of its own inherent or unforeseen issues.

In the past, many datasets have been found to contain inadvertent *artifacts* or deficiencies. The well known DARPA 1999 dataset [9], for example, uses a limited set of TTL values in all TCP headers. Unknowingly, attacks use TTL values that background traffic does not use, making attacks easy to identify. The FLAME [4] tool injects synthetic attacks at the flow level. Each synthetic flow is injected with a predictable delay, making the synthetic flows easy to uncover. In Section 3.3, a classification of *artifacts* is presented with more examples like these.

TIDED tests the reliability of any network capture file (in the PCAP format). This module focuses on finding abnormal statistical properties that might point to *artifacts* or defects that contradict the dataset's requirements. A set of statistical tests are run on top of a dataset. The results of these tests have a twofold use. On the one hand, they are used to measure and validate certain metrics belonging to the network's background traffic. On the other hand, the reliability tests are put at the disposal of ID2T's attack controller (see Figure 4). This enables the attacks to refine the parameters used by attacks to better replicate the network's background traffic. With this module, ID2T becomes not only an attack injection tool, but also a network dataset analysis tool.

5.1 Test Categories

The statistical tests performed by TIDED can be categorized as shown in Figure 7. *Availability Tests* refer to those that verify the availability of packets' payloads. *Validity Tests* look for either TCP checksum problems, or invalid and uncommon ports. The port validity tests report the usage of standard but unassigned ports (in accordance to IANA (Internet Assigned Numbers Authority)) and how many times port 0 is used. *Diversity Tests* use a set of metrics (see Section 5.2) to present information related to the packets' header fields. With this set of metrics, we quantitatively qualify an analyzed network. This enables us to identify potential problems when certain qualities do not meet expectations. For example, if the analyzed traffic comes from a supposedly backbone network and the entropy of source IP addresses is below a threshold, there is evidence to suspect that the traffic is not from the claimed source. In the following, the tests are discussed in more detail.

5.1.1 Availability Tests.

Payload availability. The ratio of packets with data payload against those without. A dataset without data payloads cannot be used by all NIDSs. Therefore, this information is needed to determine if a dataset is suitable only in restricted evaluation scenarios.

5.1.2 Validity Tests.

TCP Checksums. The ratio of wrong TCP checksums against correct ones. This test helps to detect synthetic datasets that have not correctly generated the network packets. It can also identify networks with faulty hardware, tools with inadequate anonymizing mechanisms or packet capturing

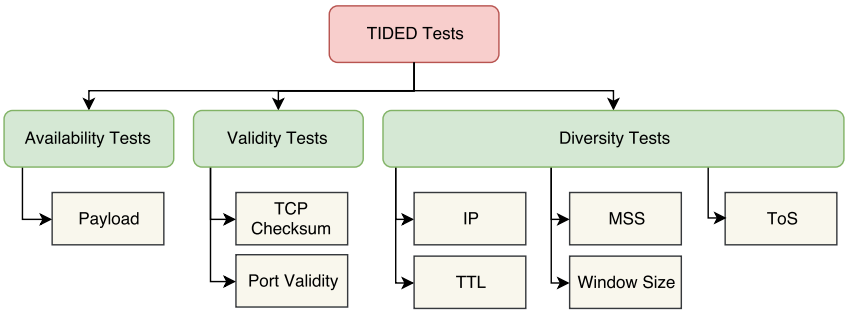


Figure 7. The categories into which the statistical tests performed by the TIDED module are divided.

tools with deficiencies. In [23], for example, attacks could be identified based on incorrect packet checksums. We note that the complete absence of incorrect checksums is potentially an artifact or defect. Real traffic usually contains small quantities of network packets with incorrect checksums.

Port Validity. The counts of port numbers used in each of the three port ranges defined by IANA in RFC 1340 [17] and RFC 6335 [8]. As a special case, the test reports the number of times packets targeted port zero. In the popular Berkeley sockets API, port zero indicates that a random port should be utilized. In wrong implementations or incorrectly generated synthetic attacks, zero is used as a port. Real networks seldom observe packets directed towards port zero.

5.1.3 Diversity Tests. The following tests use various metrics, which will be explained on the next section.

IP Diversity. The goal here is to quantitatively characterize the diversity of source and destination IP addresses . The IP address diversity correlates with the network type (home, office, backbone, industrial control, etc.). A combination with the knowledge of the network type enables an analyst to identify potential issues.

TTL Diversity. This aims to characterize the diversity of the TTL values of all network packets. Different sources alter the TTL value of the packets, for example, different operating systems use different initial TTL values and networking devices (e.g., routers, switches, etc.) alter these. The diversity of this field correlates with the diversity of the packets' sources and networking devices.

MSS (Maximum Segment Size) Diversity. The distribution of all observed MSS values within a PCAP file. Although MSS values tend to remain constant in most packets, certain hosts choose to increase it to maximize their throughput. Conversely, a host may choose to lower the MSS to reduce IP fragmentation. The distribution of the MSS may correlate with a network's design (e.g., data or resource sharing, providing services, cloud computing, etc.).

Window Size Diversity. This shows the changing behavior of the window sizes of network packets. When troubleshooting network problems, the window size is often a key indicator to examine. Additionally, networks attached to different systems (e.g., workstations, clusters, grids, etc.), benefit from tuning the window sizes [11].

ToS Diversity. This relates to the evolution of the ToS TCP values. The meaning of the ToS header field, originally defined in RFC791, has gone through five different definitions until its current definition, as defined in RFC2474. Therefore, the diversity of this field lets an analyst identify the year and version of the used TCP standard.

5.2 Diversity Test Metrics

With the help of a few metrics, an analyst may quantify and characterize key aspects of a network PCAP file. ID2T is also able to use these metrics to create attacks that better replicate the input traffic. To express diversity, we use several metrics and graphs.

5.2.1 (Shannon) Entropy distribution. We use entropy to characterize the uncertainty of observing a particular feature within a time window. Entropy is calculated as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2 P(x_i), \tag{1}$$

where $X = \{x_1, x_2, \ldots, x_n\}$ and $P(X)$ is the probability mass function of X. The minimum entropy value is zero and the maximum is $\log_2 n$, where n is the total number of elements $\|X\|$. Low entropy signals that some feature values rarely change within a time window; that is, the values of the feature are predictable. High entropy indicates that different values of the same feature are repeatedly seen; the predictability of such a feature is low. In Figure 8, we exemplify this metric by showing the entropy of the source IP addresses of all packets seen in a PCAP file of the MAWI dataset (see Section 3.1.3 for a description of the MAWI dataset). The file is analyzed in 100 intervals. It can be observed that in the first quarter of the capture, the traffic is highly irregular. Afterwards, the traffic settles and the observed source IP addresses are less randomly encountered.

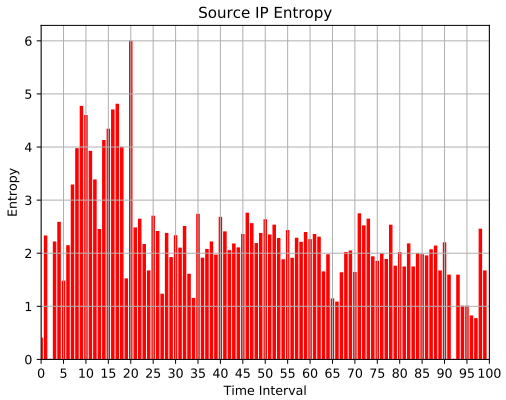


Figure 8. Example entropy graph of all the observed source IPs in a network PCAP file.

5.2.2 Normalized (Shannon) entropy. The normalized entropy $H_n(X)$ is defined as the entropy $H(X)$ divided by the maximum entropy value $\log_2 n$ for n samples:

$$H_n(X) = -\sum_{i=1}^{n} \frac{P(x_i) \cdot \log_2 P(x_i)}{\log_2 n}. \tag{2}$$

Normalized entropy lies in the range $[0, 1]$. Normalized entropy is used when comparing two random variables that have different number of samples n. Figure 9 compares six different properties belonging to eight different datasets. The DARPA1 and DARPA2 bars correspond to two different PCAP files found in the DARPA dataset. Similarly, the MAWI1 and MAWI2 bars corresponds to PCAP file from different days of the MAWI dataset.

A few conclusions can be made from comparing the normalized entropy: The DARPA datasets have considerably high ToS entropy which corresponds to the fact that DARPA was created when the ToS field was defined to have a different meaning than more modern datasets. The DARPA dataset has deficiencies in how window sizes are chosen. When contrasting the normalized entropy of the window sizes, this deficiency of the DARPA dataset becomes obvious (the window sizes barely change due to how synthetic traffic was generated [22]). The NGIDS-DS synthetic dataset also has some potential issues when looking at the MSS, ToS and window size. Furthermore, from the normalized entropy of the destination IP addresses, it can be seen that almost always the same hosts are contacted; signaling a potential lack of network diversity. From the same perspective of network diversity, the UNSW-NB almost always captures traffic of hosts that are only seen once. This is not an issue on itself. Depending on the purpose of the dataset, however, a high diversity might reduce its usability (e.g., learning patterns of normality for detecting anomalous host communications).

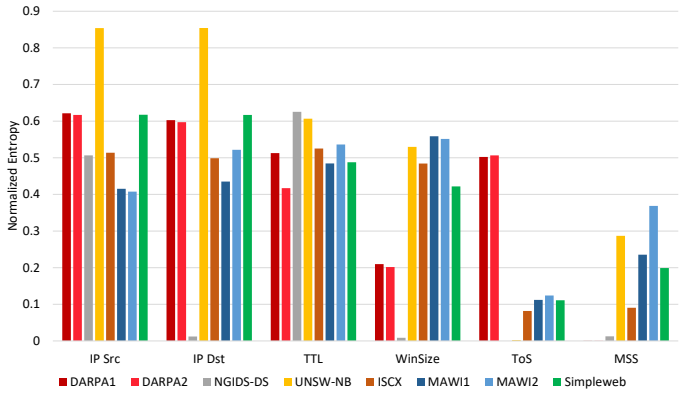


Figure 9. Comparison of normalized entropies between different datasets.

5.2.3 Novelty distribution. Distribution of newly observed values at different time windows. This metric partitions the dataset into time windows (x-axis). For each time window, the number of values (y-axis) that have never seen in the previous time windows are displayed. Figures 10 and 11 show examples of this metric for different network features.

In Figure 10, the distribution of novel IP addresses of a day of the DARPA dataset are shown. After time window 60, where each time window spans around 15 minutes, no new IP addresses are seen again.

In Figure 11a and 11b, the novelty distribution of window sizes of two datasets can be compared. In Figure 11a, we see that new window sizes in one day of the MAWI dataset are always recorded (in diminishing counts). This is consistent with the fact that MAWI has data of a backbone network. In contrast, Figure 11b shows that only in few time windows new values of window sizes are observed. This points to a potential dataset defect: this is not expected given that the UMSW-NB dataset is supposed to contain real network traffic with injected attacks for the evaluation of NIDS.

5.2.4 (Normalized) Entropy of novelty distribution. The (normalized) entropy pertaining to a specific network feature as found in the entire dataset. Entropy is not only used for determining the predictability of a random variable, it is also used to characterize the shape of a distribution. This metric provides a quantitative measurement that provides a graphical representation as well as the means to compare different datasets.

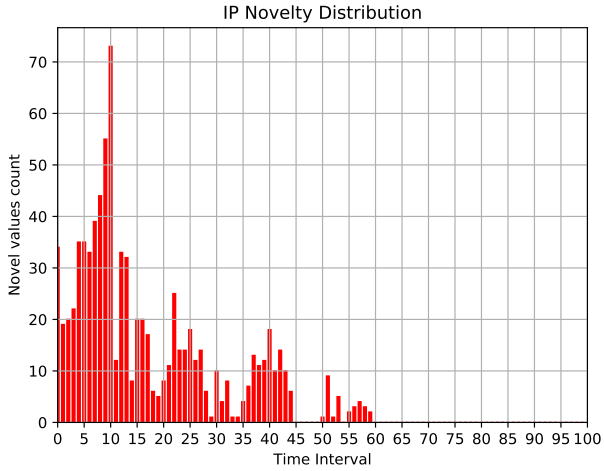
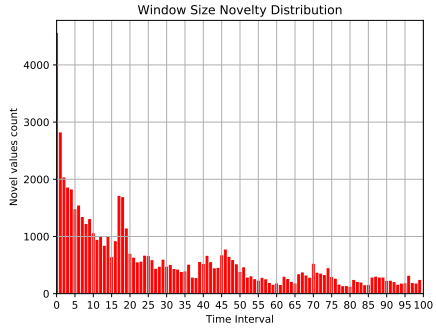
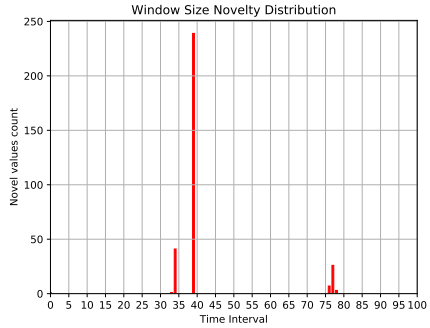


Figure 10. Novelty distribution of IPs of one day of the DARPA dataset. Halfway though the day, no records of new IP addresses are observed.



(a) Novelty distribution of window sizes of a day of the MAWI dataset.



(b) Novelty distribution of window sizes of a day of the UNSW dataset.

Figure 11. Example plots of the novelty distribution metric.

Figure 12 shows a comparison of the normalized entropy of different network features as found in different datasets. From the figure, it is possible to see that the ToS, TTL and MSS header fields of the DARPA dataset (DARPA1 and DARPA2) have seen a low number of novel values due to their low entropy. The same is true for the UNSW-NB dataset, with the addition that the window sizes and IP addresses have this same property. Due to the nature of the dataset, this is a potential issue. The MAWI (from two distinct days, MAWI1 and MAWI2) and Simpleweb datasets can also be assessed from the plot. Namely, these datasets record traffic from large networks and, therefore, never cease to observe new IP addresses (along with different window sizes, presumably due to congestion prevention mechanisms).

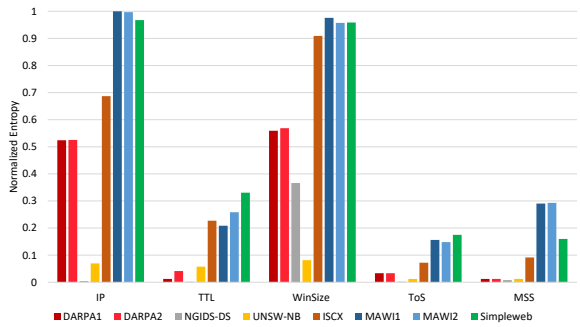


Figure 12. Comparison of the normalized entropies of the distribution of novel values between different datasets.

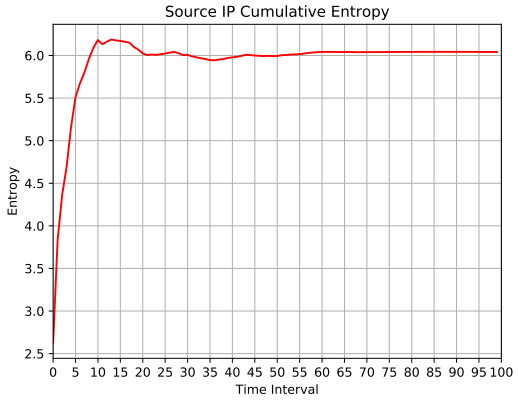


Figure 13. Cumulative entropy of a day in the DARPA dataset for the IP sources.

5.2.5 Cumulative (Shannon) entropy distribution. The cumulative entropy of a dataset at different time windows. At each time window, all previous time windows are used to calculate a feature's entropy. This metric enables an analyst to identify unexpected entropy gains (or losses).

Figure 13 shows the source IP cumulative entropy of one day of the DARPA dataset. This plot is an alternative view of the entropy distribution plot (cf. Figure 8). In this example, we observe how no new source IP addresses are observed in one of the days of the DARPA dataset after time interval 60.

6 SYNTHETIC ID2T ATTACKS

ID2T puts several synthetic attacks at the disposal of the NIDS community. Figure 14, presents a classification of the attacks that are currently supported by ID2T. Each attack attempts to create synthetic traffic that replicates, whenever possible and desirable, the conditions of some background network traffic. For example, ID2T replicates the distributions of various header fields' values, such as the TTL, MSS, and window size. Moreover, the rate of the injected packets is calculated in a way that considers the changes of the background traffic intensity, i.e., high traffic intensity leads to extra latency, thus, the packets are injected at a lower rate. Users can provide ID2T with a set

of parameters to adjust the generated attack. These parameters vary from one attack to another. If the user does not provide the required parameters, ID2T uses the statistical properties of the background traffic to select proper values automatically. In addition, ID2T considers the behavior of real-world tools, e.g., Nmap and Metasploit, that are commonly used to perform such attacks. In the following subsections, we will describe the categories of attacks in Figure 14 in more detail.

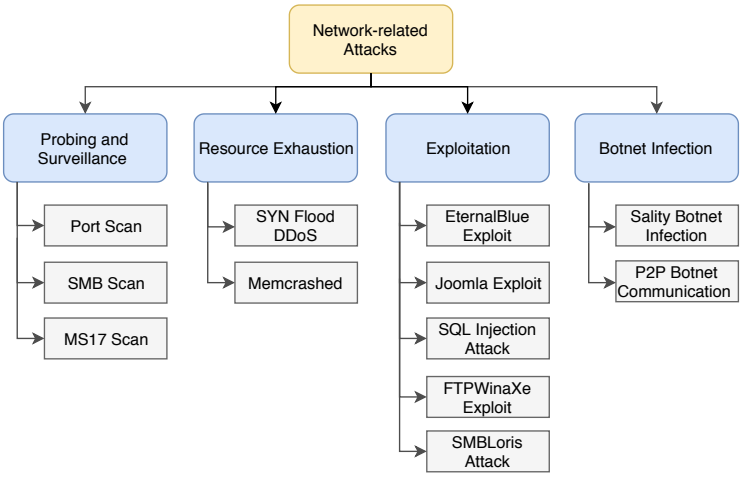


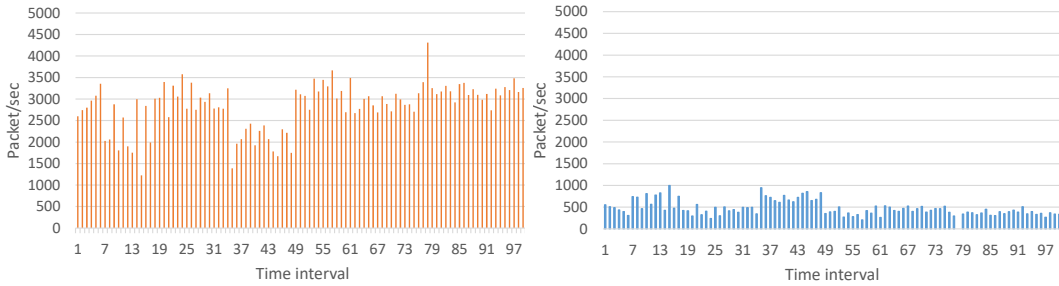
Figure 14. Classification and attacks provided by ID2T.

6.1 Probe and Surveillance

This class contains scan techniques which aim at collecting information about networks or hosts, usually, with the intention of preparing for further attacks. In ID2T, three scans are provided as follows.

6.1.1 Port Scan. A port scan is a reconnaissance technique used to discover vulnerabilities in network hosts by sending port probes. In ID2T, a vertical *TCP SYN* port scan is implemented. This scan targets various ports on a single host by sending *SYN* packets. ID2T generates three types of packets, namely *SYN*, *SYN+ACK*, and *RST*. Furthermore, ID2T imitates the behavior of an Nmap (default) scan with regard to the targeted ports; that is, the Nmap-service table is used to select the most common 1000 open TCP ports. Nmap utilizes an adaptive packet rate technique, in which the packet rate corresponds to the changes of the background traffic intensity. ID2T replicates this behavior. In particular, this was implemented by using the complementary packet rate of the background traffic after normalizing it to a user-selected value. Figure 15a, shows the intervals of the packet rate of an arbitrary PCAP file used as background traffic, while Figure 15b, depicts the complementary packet rate normalized to 1000 packets per second, which is used for the injected packets rate.

6.1.2 SMB Scan. This attack scans a network for SMB (Server Message Block) servers. In particular, the attacker attempts to establish TCP connections with the victim(s) on the port 445. If the connection is established successfully, an SMB negotiation starts and more packets are exchanged, where the attacker can learn more about the service provided by the victim (e.g., the version of SMB). ID2T simulates this scan by generating and injecting the packets of TCP connections and SMB negotiations.



(a) Background traffic packet rate.

(b) Injected attack with a user-selected normalized packet rate.

Figure 15. Packet rate of an arbitrary PCAP file and an injected attack.

6.1.3 MS17 Scan. This specialized scan examines whether a victim has the MS17-010 patch. This patch resolves several vulnerabilities in the implementation of SMBv1 in MS Windows OS. One of these vulnerabilities was used by the EternalBlue exploit. In Metasploit, the module *smb_ms17_010* [4] performs this scan. ID2T utilizes the packets of the *smb_ms17_010* module as templates. It then injects these packets into the input PCAP file after manipulating them to replicate the properties of the background traffic.

6.1.4 Probing limitations. During probe activities, attackers receive responses from the victims and these responses are used to derive information about the victim. ID2T provides a limited version of these responses. For example, in a real-world port scan, the response packets can be *SYN+ACK*, *RST*, or *ICMP unreachable error* packets. In ID2T, we consider one response packet type, which is the *SYN+ACK* packet. However, detecting probe scans depends mainly on the traffic generated by the attackers. Therefore, the packets injected by ID2T are sufficient to evaluate IDSs against these scans.

6.2 Resource Exhaustion

The attacks in this class aim to use up the resources of networks or hosts, to deny legitimate users access a particular service. ID2T provides two attacks under this class, namely, SYN Flood DDoS and Memcrashed.

6.2.1 SYN Flood DDoS Attack. In this attack, a multitude of machines establish a massive amount of TCP connections with a victim to use up its resources. ID2T creates this attack by generating two types of packets: *SYN* packets, which are sent by the attackers, and *SYN+ACK* packets, which are sent by the victim. Known aspects from the victim previous activities in the background traffic are taken into account; first, to determine which (open) ports to attack, and second, to estimate the packet rate that is sufficient to bring the host down. Moreover, ID2T imitates some of the properties of Metasploit DoS attack's packets; for instance, the random window size.

6.2.2 Memcrashed Attack. This is an amplification attack that exploits a vulnerability[5] in Memcached[6] servers, where an attacker sends forged UDP requests with a spoofed targeted source IP to

[4] https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/smb/smb_ms17_010.rb
[5] CVE-2018-1000115
[6] Distributed memory caching system used to speed up dynamic web applications.

servers. The servers send back responses to the targeted source IP, overwhelming its resources. Currently, ID2T injects only the first part of this attack, i.e. the packets sent from the attacker to the Memcached servers.

6.2.3 Resource exhaustion limitations. In real-world networks, resource exhaustion attacks leave usually remarkable impacts on the targeted networks and hosts, such as increasing the response time of the victim and creating network congestions, thus, increasing the network latency and causing packet loss. Replicating such impacts requires modification in the background traffic, e.g., by deleting packets, which is not considered currently in ID2T. However, the resource exhaustion attacks are usually recognizable by NIDSs based on the suspicious traffic generated by attackers, rather than the implications on the normal traffic. Thus, ID2T datasets can be used effectively to evaluate NIDSs against these attacks.

6.3 Exploitation

These attacks target an existing bug or vulnerability in a system with the intention of gaining control, privilege escalation, or denying a service. Five different exploits are available in ID2T.

6.3.1 EternalBlue Exploit. This exploit targets a buffer overflow vulnerability[7] in the SMBv1 in MS Windows OS. The Metasploit module *eternalblue_doublepulsar*[8] performs this attack. ID2T injects the packets generated by this module after manipulating the header fields, while maintaining the payload since it contains the malicious code. During this attack, several TCP connections are established. ID2T takes into account preserving the conditions of these connections with regard to the order, overlap, and dependency.

6.3.2 FTPWinaXe Exploit. In this attack, a malicious FTP server sends packets with overly long payloads to a WinaXe 7.7 [9] FTP client, exploiting a buffer overflow vulnerability. A user can provide the payloads as input, otherwise ID2T generates random ones.

6.3.3 Joomla Privilege Escalation Exploit. This exploit uses a vulnerability[10] found in Joomla[11] versions 3.4.4 through 3.6.3. The vulnerability allows attackers to create an arbitrary account with administrative privileges. ID2T uses template packets obtained from the Metasploit *joomla_registration_privesc* module[12]. In this attack, ID2T manipulates the packets' header fields and the HTTP headers of the payload.

6.3.4 SMBLoris Attack. This attack exploits a vulnerability in the SMB protocol that allows an attacker to make large memory allocations without being authenticated. ID2T creates this attack by targeting the SMB port of the victim with NBT (NetBIOS over TCP) packets that have the maximum value in the length field.

6.3.5 SQL Injection Attack. This attack targets a vulnerability[13] found in ATutor 2.2.1 [14] applications. The vulnerability allows attackers to inject SQL statements, bypass authentication, and gain administrator privileges. Metasploit *atutor_sqli* module performs this attack. ID2T injects the

[7] CVE-2017-0144
[8] https://github.com/ElevenPaths/Eternalblue-Doublepulsar-Metasploit
[9] An X Windows environment, enables different OSs and their applications to be connected through SSH, TCP/IP, NFS, FTP, TFTP and Telnet.
[10] CVE-2016-8870
[11] Content management system for web applications.
[12] https://www.rapid7.com/db/modules/auxiliary/admin/http/joomla_registration_privesc
[13] CVE-2016-2555
[14] Content management system for education purposes.

packets of this module after modifying them to adapt with the background traffic characteristics and the user parameters.

6.3.6 Exploitation limitations. This class of attacks are mainly distinguished by the packet payload, where the exploit is located. ID2T effectively mimics such attacks by copying real malicious payloads. However, ID2T is limited to producing specific versions of these attacks. For example, in real-world EternalBlue, the number of connections can vary based on the victim's resources, while in ID2T, a fixed number of connections is generated.

6.4 Botnet Infection

A botnet is a set of network-connected compromised machines that work in a coordinated fashion for malicious purposes, such as email spam delivery and performing DDoS attacks. ID2T provides the ability to inject two types of botnet traffic: a variant of Sality botnet and user-defined botnet communication patterns.

6.4.1 Sality Botnet. Sality is a classification of malicious programs that infect executable files in MS Windows OS. Over time, Sality programs were developed to contain a variety of abilities, such as exfiltrating sensitive data. The Sality variant that is supported in ID2T is known as *Win32-Sality.AM*. It loads a malicious DLL file in the memory of the infected host. ID2T injects the traffic of this botnet, which was obtained from *VirusTotal.com*, after modifying the packet headers and the packet rate.

6.4.2 P2P Botnet Communication. ID2T is also able to generate P2P botnet communication traffic. A user needs to provide ID2T with a specification of the attack, namely, a CSV (Comma Separated Values) file, where the interactions and type of messages exchanged between bots are specified. The *Attacks* module in ID2T handles this file and generates the botnet traffic based on its content. ID2T can either add this traffic to existing hosts in the input PCAP or generate new hosts acting as the bots.

6.4.3 Botnet infection limitations. In the ID2T version of the Sality botnet, a fixed set of malicious IP addresses is currently used. In real-world, these addresses can be more diverse and can be changed dynamically. However, NIDSs can detect the Sality botnet traffic not just based on the malicious IP addresses, but also based on footprints in the HTTP headers, which are replicated by ID2T.

7 USE CASES: APPLICATIONS OF ID2T

In this chapter, we demonstrate how ID2T can be applied to determine the detection capabilities of different IDSs using replicable datasets. We illustrate this with two use cases. In the first use case, we inject attacks into a publicly available PCAP file and use an anomaly detection IDS to demonstrate the detection capabilities of the system. In the second use case, we use publicly available signature-based IDSs to demonstrate that the synthetic attacks injected by ID2T are indeed triggering expected signatures. Therefore, assessing whether the signature-based IDS is working as intended. These use cases show how ID2T can be used to compare or test existing systems with datasets that can be replicated.

7.1 Assessing Anomaly Detection Capabilities

As a first use case, we use ID2T to demonstrate the detection capabilities of an anomaly-based NIDS in a backbone network scenario. For such demonstration, we need a labeled PCAP of backbone traffic that contains the attack we wish to detect. The MAWI dataset [12] provides good candidate PCAPs that include labels. Nonetheless, the provided labels do not reflect the ground

truth concerning the network traffic stored in the PCAPs. The labels mark anomalous traffic in accordance to three different anomaly detectors. To determine the type of attacks, a heuristic is used. Because no human is involved in the labeling process, the accuracy of the labels cannot be guaranteed. Furthermore, due to the amount of network traffic, it is also unfeasible to assess if there are other unlabeled attacks. With ID2T, we can inject precise attacks into sanitized MAWI PCAPs files (e.g. removing traffic labeled as anomalous) to determine if an IDS can at least detect the injected attacks. This approach enables us to test the detection accuracy (and precision) of an IDS against a specific set of chosen attacks.

We demonstrate the detection capabilities of an anomaly detection methodology that uses shallow autoencoders, also known as RNNs (Replicator Neural Networks) [6]. The demonstration, however, is applicable to any other anomaly-based NIDS. RNNs learn how the entropies of traffic features behave within a time window. When testing for anomalies within a time window, the RNN assigns an anomaly score to the network traffic based on what it learned. It is assumed that traffic that does not conform to the learned behavior will yield higher anomaly scores in contrast to benign traffic. When the anomaly score passes a threshold determined by the model, an anomaly is raised (signaling a potential attack).

We use four days (from 2018/04/01 to 2018/04/04) of backbone network traffic of the MAWI dataset to train, inject and detect port scans and DDoS attacks using an RNN. We choose these two attacks as these are expected to leave distinguishable network fingerprints that an anomaly detector should identify. Each day of the MAWI dataset consists of 15 minutes of traffic. We processed each day using a three step process. First, we split each day in time windows of 10 seconds. Second, we convert all traffic within a time window into network flows. The resulting flows are sanitized by removing those marked as anomalous by the labels provided by the MAWI dataset. Finally, we calculate the entropy of source and destination IPs and ports (for a total of four entropies) of all flows (see [6] for more details). The set of entropies belonging to the time windows of the first three days are used to train an RNN. The last day is used for testing purposes.

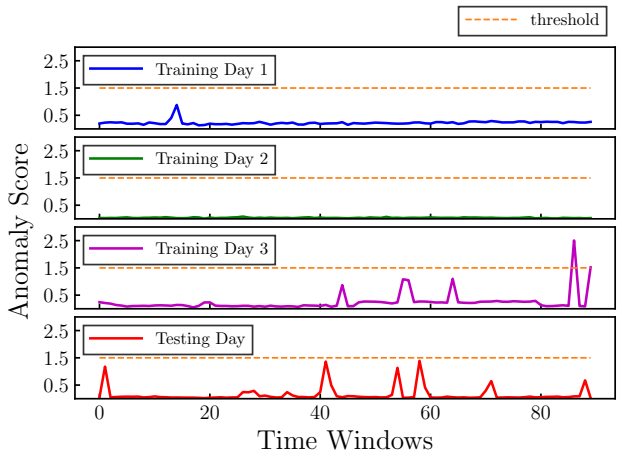


Figure 16. Anomaly scores given to four different days of the MAWILab dataset by an RNN. The first three days are used to train the model. The fourth day is used to test the trained model.

Figure 16 shows the anomaly scores of the training and testing days as given by the trained RNN. The x-axis shows the different time windows into which a single day is split (15 minutes ×

60 seconds = 90 time windows). The y-axis shows the anomaly score of the entropies calculated at each time window. The RNN uses the first three days for training and is, therefore, expected to yield low anomaly scores for these training days. This is generally the case except for two instances within the third day (at time window 85 and 90). After manually examining those time windows, we identified that both spikes in the anomaly score correspond to an irregular amount of observed source ports (the entropy of the source ports more than doubled). By removing the outliers, the mechanism established an anomaly score threshold of 1.5. That is, any time window with an anomaly score above the dashed line is considered to have experienced anomalous traffic.

7.1.1 Detecting DDoS Attacks. We proceed to inject DDoS attacks (see Section 6.2.1) of different intensities into the testing day and use a RNN to detect the attacks. Figure 17 shows the resulting anomaly scores of the testing day with three different DDoS intensities. All attacks are injected into a clean testing day (attacks are not stacked) at time window 30. All attacks last a total of one minute (until time window 36). The upper left plot, shows the anomaly scores of the testing day with no injected attacks for reference. The upper right plot shows a 3,000 packets per second DDoS attack. The bottom left plot shows a 6,000 packets per second DDoS attack. The plot to the bottom right shows a high intensity DDoS attack of 10,000 packets per second. The PCAP file of the testing day contains 78 million packets. The percentage of packets added to the PCAP is, from the low to high intensity attacks, 0.23, 0.46 and 0.77 percent, respectively.

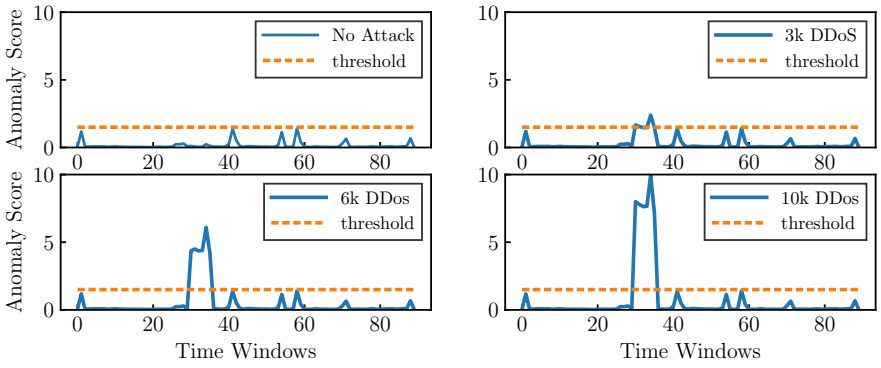


Figure 17. Detecting different DDoS attacks injected with ID2T using anomaly detection.

From Figure 17, we observe that the injected DDoS attacks are detected. The anomaly scores of the time windows where the attacks lie (between 30 and 36) are above the predefined threshold. With the help of ID2T, we are able to demonstrate the DDoS detection capabilities of RNNs. This use case scenario shows how ID2T can be used to reproduce and replicate the conclusions found in other publications without necessarily having to use the same dataset. In this case, we use the same type of background traffic and attack. The original RNN publication, however, uses different background traffic (days of the MAWI dataset from almost two year ago in relation to this work).

7.1.2 Detecting Port Scans. Using ID2T, we test the capability of RNNs to detect port scans in large networks. Figure 18 shows plots of the anomaly scores of three different datasets injected with different port scan intensities. All injections take place at time window 30 and last 30 seconds (3 time windows). On the top left of the figure, as a reference, we show the anomaly scores of the target dataset without injected attacks. On the top right, we show the anomaly scores of the dataset

after injecting a port scan targeting 10,000 random ports of five different host. On the bottom left, we show the anomaly scores when the injected port scan targeted five hosts and randomly scanned 40,000 ports. Finally, on the bottom right, we show the anomaly scores when the port scan targeted five hosts; scanning 10,000 random ports. The RNNs are able to detect all port scans as can be seen by the anomaly scores that go beyond the marked threshold.

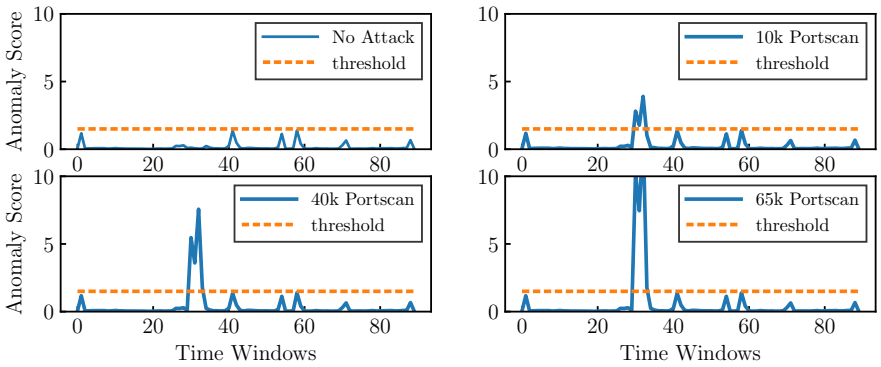


Figure 18. Anomaly scores of different port scans of varying intensities.

7.2 Testing Configurations of Signature-based NIDSs

In the second use case, we test the configuration of two signature-based NIDSs using ID2T. By injecting PCAP files with specifically chosen exploits or malware, we test an arbitrary configuration of Bro[26] and Suricata[15].

	Bro	Suricata
Port Scan	✓[1]	x
EternalBlue Exploit	✓	✓[2]
FTP WinaXe Exploit	x	x
Sality Botnet Infection	✓	✓[3]

[1] Identified as: Scan::Port_Scan 188.165.214.141 scanned at least 15 unique ports of host 188.165.214.253 in 0m4s.
[2] Exploit identified as: ETERNALBLUE MS17-010 Echo Reponse and ETPRO TROJAN (possible Metasploit payload).
[3] Sality identified as: ETPRO TROJAN Win32/Sality.AM and ET MALWARE Sality Virus User Agent Detected (KUKU)

Table 3. Testing different IDSs againt four different attacks injected by ID2T.

Using ID2T, we inject four different attacks into a PCAP file of office network traffic collected during a one minute interval. One minute of captured network traffic was close to 30 MiB of data and 55,000 packets. Because we inject small specialized attacks (that generate no more than 500 packets), this gives us enough room to hide our attacks. Table 3 shows the detection results of Bro and Suricata when only basic signatures are installed. We mark successful detection with an arrow (✓). A cross (x) indicates that the corresponding malicious activity was not detected. We emphasize that these are not the results of testing NIDSs themselves, but rather an arbitrary set of signatures we installed. The port scan was detected by Bro. One of Bro's anomaly rules

[15]https://www.openinfosecfoundation.org

found that the IP that we specified as an attacker was conducting port scans. Suricata, on the other hand, was not configured with signatures capable of detecting port scans. Both NIDSs are able to detect the popular "EnternalBlue" exploit. Suricata detected, additionally, that the payload being used potentially originated from Metasploit. ID2T mimics what Metasploit does to create the "EternalBlue" exploit. The FTP WinaXe Exploit is a difficult attack to detect. A signature for it cannot be easily created as the exploit relies on overflowing a particular FTP command that only affects some vulnerable versions of the WinaXe program. Hence, as to be expected no NIDS detected the malicious payload. Finally, both NIDSs correctly identify a Sality bot infection. Suricata is also able to detect that the user agent of some HTTP traffic is made by Sality. ID2T mimics how Sality sends HTTP messages with the user agent field set to "KUKU".

7.3 Discussion of the Use Cases

The previous two use case scenarios help illustrate how ID2T can be used to either reproduce or replicate results as well as to test the detection capabilities of an arbitrary setup of NIDSs. In the first use case, we reproduce the network intrusion detection capabilities of RNNs using different days of the MAWI dataset (in contrast to the days used in the original publication). Although the PCAP for training the RNN in the original publication and our use case come from MAWI, there are almost two years of difference between them. We argue that in the last two years the characteristics of network traffic in backbone networks have changed considerably [1]. In this context, ID2T was able to help us reproduce past results that also apply to modern network traffic without creating an updated dataset from scratch. The process of creating new synthetic attacks that ID2T can inject is considerably faster than publishing new datasets every time a novel attack or network configuration needs to be tested.

In the second use case scenario, we use ID2T to test two popular NIDS. ID2T is a tool that enabled the creation of network traffic that contained certain attacks that are easy to detect using the right signatures. We showed how a properly configured installation of the Bro and Suricata NIDSs are able to detect expected attacks. ID2T can be used, therefore, to verify the correct operation of already deployed systems.

8 CONCLUSION AND FUTURE WORK

We developed ID2T to try to address the long standing issue of not having reliable and reproducible datasets in the NIDSs community. To evaluate systems, researchers in this community have a tendency of using old, incomplete or private datasets that hamper their results. For example, it is a common occurrence to find new publications that use outdated datasets (such as the DARPA 1999 dataset). A system that demonstrates its capabilities on 19 year old network traffic cannot be trusted to work on modern traffic. The problem of acquiring reliable datasets lies in the difficulty of creating datasets that are useful and, at the same time, fulfill the conditions or constraints of the dataset's creators. Privacy, for example, is a major concern that ends up impacting a dataset because of the way data is anonymized or concealed.

With ID2T, we provide a tool that enables the creation of custom datasets while solving the problems of reproducibility and privacy preservation. Instead of providing a single static dataset, ID2T gives researchers the tools to inject synthetic attacks into background traffic they provide. The injected attacks replicate the characteristics of the background traffic to erase any trace that points towards the usage of ID2T and the existence of synthetic traffic. With such a tool, we envision researchers demonstrating the usage of their systems by specifying the type of background traffic used and the parameters used to create an attack. For example, as illustrated in our use-case section (see Section 7.1), we use backbone network traffic to detect DDoS attacks and SMB probing scans.

One key functionality of ID2T is that of replicating the characteristics of the provided background traffic in the synthetic attacks it generates. Therefore, the properties of the background traffic impact the injected attacks. We developed the TIDED module of ID2T (see Section 5) for the purpose of helping researchers to determine if the characteristics of the provided background traffic conform to their expectations. With such tests, for example, it is possible to determine if traffic that claims to come from a backbone network reflects the properties of such a large network. With TIDED, it is also possible to detect potential sources of defects that a dataset may have.

In future work, we wish to expand the capabilities of ID2T in two directions. First, we plan to enlarge the arsenal of injectable attacks to include recent popular attacks. Second, we would like to address the main limitation of ID2T. As it stands, ID2T has the limitation of not having a feedback loop between the generated attacks and the background traffic. In real scenarios, an attack affects the characteristics of the network onto which it was launched. A DDoS attack, for example, may cause a high number of TCP re-transmissions and lower average packet window sizes. The attacks generated by ID2T are limited to only replicating what is observed in the background traffic. It is also desirable, for the purpose of creating more realistic traffic, to model the interaction that an attack has on the background data. With such an interaction, attacks would also modify the background traffic to either add, modify or delete packets.

REFERENCES

[1] Akamai. 2018. The state of the internet / security report. https://www.akamai.com/uk/en/multimedia/documents/case-study/spring-2018-state-of-the-internet-security-report.pdf Accessed: 2018-06-13.

[2] Rafael Ramos Regis Barbosa, Ramin Sadre, Aiko Pras, and Remco Meent. 2010. *Simpleweb/university of twente traffic traces data repository.* Technical Report. Centre for Telematics and Information Technology, University of Twente.

[3] Monowar H. Bhuyan, Dhruba K. Bhattacharyya, and Jugal K. Kalita. 2015. Towards generating real-life datasets for network intrusion detection. *International Journal of Network Security* 17, 6 (2015), 683–701.

[4] Daniela Brauckhoff, Arno Wagner, and May Martin. 2008. FLAME: a Flow-Level Anomaly Modeling Engine. In *The conference on Cyber security (CSET).*

[5] CAIDA. 2017. CAIDA Data - Overview of Datasets, Monitors, and Reports. http://www.caida.org/data/overview/. [Online; accessed 22.09.2017].

[6] C. García Cordero, S. Hauke, M. Mühlhäuser, and M. Fischer. 2016. Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks. In *2016 14th Annual Conference on Privacy, Security and Trust (PST).* 317–324. https://doi.org/10.1109/PST.2016.7906980

[7] Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Nikolay Milanov, Christian Koch, David Hausheer, and Max Mühlhäuser. 2015. ID2T: a DIY dataset creation toolkit for Intrusion Detection Systems. In *Conference on Communications and Network Security (CNS).* IEEE, 739–740.

[8] Michelle Cotton, Lars Eggert, Joe Touch, Magnus Westerlund, and Stuart Cheshire. 2011. *Internet assigned numbers authority (IANA) procedures for the management of the service name and transport protocol port number registry.* RFC 6335. http://buildbot.tools.ietf.org/html/rfc6335

[9] Robert K Cunningham, Richard P Lippmann, David J Fried, Simson L Garfinkel, Isaac Graf, Kris R Kendall, Seth E Webster, Dan Wyschogrod, and Marc A Zissman. 1999. *Evaluating intrusion detection systems without attacking your friends: The 1998 DARPA intrusion detection evaluation.* Technical Report. MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB.

[10] Knowledge Discovery and Data Mining Tools Competition. 1999. KDD Cup 99 Task. https://kdd.ics.uci.edu/databases/kddcup99/task.html. [Online; accessed 11-July-2017].

[11] Wu-chun Feng, Justin Gus Hurwitz, Harvey Newman, Sylvain Ravot, Roger Les Cottrell, Olivier Martin, Fabrizio Coccetti, Cheng Jin, Xiaoliang David Wei, and Steven Low. 2003. Optimizing 10-Gigabit Ethernet for networks of workstations, clusters, and grids: A case study. In *Supercomputing, 2003 ACM/IEEE Conference.* IEEE.

[12] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Conference on emerging Networking EXperiments and Technologies (CoNEXT).* ACM, 1–12.

[13] Dan Grossman. 2002. *New terminology and clarifications for diffserv.* RFC 3260. http://buildbot.tools.ietf.org/html/rfc3260

[14] W Haider, J Hu, J Slay, BP Turnbull, and Y Xie. 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *Journal of Network and Computer Applications* 87 (2017), 185–192.
[15] Hannes Holm, Martin Karresand, Arne Vidström, and Erik Westring. 2015. A survey of industrial control system testbeds. In *Secure IT Systems*. Springer, 11–26.
[16] IMPACT. 2017. Data & Tools. https://www.impactcybertrust.org. [Online; accessed 22.09.2017].
[17] M St Johns. 1993. *Identification protocol*. RFC 1340. http://buildbot.tools.ietf.org/html/rfc1413
[18] Robert Koch, Mario Golling, and Gabi Dreo Rodosek. 2014. Towards Comparability of Intrusion Detection Systems: New Data Sets. In *TERENA Networking Conference*. 7.
[19] LBNL. 2005. LBNL/ICSI Enterprise Tracing Project. http://www.icir.org/enterprise-tracing/Overview.html. [Online; accessed 22.09.2017].
[20] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. 2000. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks* 34, 4 (2000), 579–595.
[21] Matthew V Mahoney. 2003. Network traffic anomaly detection based on packet bytes. In *Proceedings of the 2003 ACM symposium on Applied computing*. ACM, 346–350.
[22] Matthew V Mahoney and Philip K Chan. 2003. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. *International Symposium on Recent Advances in Intrusion Detection* 2820 (2003), 220–237. https://doi.org/10.1007/b13476
[23] Matthew V. MV Mahoney and PK Philip K. Chan. 2001. *PHAD: Packet Header Anomaly Detection for identifying hostile network traffic*. Technical Report 1998. Florida Institute of Technology. 1–17 pages.
[24] Nour Moustafa and Jill Slay. 2015. 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*. IEEE, 1–6.
[25] Sybil P. Parker. 2003. *McGraw-Hill Dictionary of Scientific and Technical Terms*. McGraw-Hill Education.
[26] Vern Paxson. 1999. Bro: a system for detecting network intruders in real-time. *Computer Networks* 31, 23-24 (dec 1999), 2435–2463.
[27] Jon Postel et al. 1981. *Internet protocol*. RFC 791. http://buildbot.tools.ietf.org/html/rfc791
[28] Benjamin Sangster, Thomas Cook, Robert Fanelli, Erik Dean, William J Adams, Chris Morrell, and Gregory Conti. 2009. Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In *USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET)*.
[29] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security* 31, 3 (2012), 357–374.
[30] CSL Sony and Kenjiro Cho. 2000. Traffic data repository at the WIDE project. In *Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*. 263–270.
[31] Gayathri Sugumar and Aditya Mathur. 2017. Testing the Effectiveness of Attack Detection Mechanisms in Industrial Control Systems. In *Software Quality, Reliability and Security Companion (QRS-C), 2017 IEEE International Conference on*. IEEE, 138–145.
[32] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Symposium on Computational Intelligence for Security and Defense Applications, CISDA*. IEEE, 1–6. https://doi.org/10.1109/CISDA.2009.5356528
[33] UMass. 2009. UMass Trace Repository. http://traces.cs.umass.edu/. [Online; accessed 22.09.2017].
[34] Emmanouil Vasilomanolakis. 2016. *On Collaborative Intrusion Detection*. Ph.D. Dissertation. Technische Universität Darmstadt, Darmstadt.
[35] Emmanouil Vasilomanolakis, Carlos Garcia Cordero, Nikolay Milanov, and Max Mühlhäuser. 2016. Towards the creation of synthetic, yet realistic, intrusion detection datasets. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*. IEEE, 1209–1214.
[36] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 2015. Taxonomy and Survey of Collaborative Intrusion Detection. *Comput. Surveys* 47, 4 (2015), 33.
[37] Emmanouil Vasilomanolakis, Matthias Krügl, Carlos Garcia Cordero, Max Mühlhäuser, and Mathias Fischer. 2015. SkipMon: A Locality-Aware Collaborative Intrusion Detection System. In *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*. IEEE, 1–8.
[38] Richard Zuech, Taghi M Khoshgoftaar, Naeem Seliya, Maryam M Najafabadi, and Clifford Kemp. 2015. A New Intrusion Detection Benchmarking System.. In *FLAIRS Conference*. 252–256.

Received June 2018