On generating network traffic datasets with synthetic attacks for intrusion detection

Cordero, Carlos Garcia; Vasilomanolakis, Emmanouil; Wainakh, Aidmar; Mühlhäuser, Max; Nadjm-Tehrani, Simin

Published in:
ACM Transactions on Privacy and Security

DOI (link to publication from Publisher):
10.1145/3424155

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

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 field of network intrusion detection heavily relies on datasets. Datasets in this field, however, are scarce and difficult to reproduce. To compare, evaluate, and test related work, researchers usually need the same datasets or at least datasets with similar characteristics as the ones used in related work. In this work, we present concepts and the Intrusion Detection Dataset Toolkit (ID2T) to alleviate the problem of reproducing datasets with desired characteristics to enable an accurate replication of scientific results. Intrusion Detection Dataset Toolkit (ID2T) facilitates the creation of labeled datasets by injecting synthetic attacks into background traffic. The injected synthetic attacks created by ID2T blend with the background traffic by mimicking the background traffic's properties.

This article has three core contributions. First, we present a comprehensive survey on intrusion detection datasets. In the survey, we propose a classification to group the negative qualities found in the datasets. Second, the architecture of ID2T is revised, improved, and expanded in comparison to previous work. The architectural changes enable ID2T to inject recent and advanced attacks, such as the EternalBlue exploit or a peer-to-peer botnet. ID2T's functionality provides a set of tests, known as TIDED, that helps 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 replicate scientific results with the help of reproducible datasets. 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

This research work has been funded by (1) the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE, and (2) the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), Grant No. 251805230/GRK 2050. The fifth author's work was supported by the research centre on Resilient Information and Control Systems (www.rics.se) funded by the Swedish Civil Contingencies Agency (MSB).

Authors' addresses: C. G. Cordero, Technische Universität Darmstadt, Telecooperation Group, Darmstadt, Hessen, 64289, Germany; email: carlos.garcia@tk.tu-darmstadt.de; E. Vasilomanolakis, Aalborg University, Electronic Systems, Cyber Security Network, Copenhagen, 2450, Denmark; email: emv@es.aau.dk; A. Wainakh, Technische Universität Darmstadt, Telecooperation Group, Darmstadt, Hessen, 64289, Germany; email: wainakh@tk.tu-darmstadt.de; M. Mühlhäuser, Technische Universität Darmstadt, Informatics Department, Telecooperation Group, Darmstadt, Hessen, 64289, Germany; email: max@informatik.tu-darmstadt.de; S. Nadjm-Tehrani, Department of Computer and Information Systems, Linköping University, Linköping, SE-581 83, Sweden; email: simin.nadjm-tehrani@liu.se.

2471-2566/2020/12-ART8 $15.00
https://doi.org/10.1145/3424155

ACM Reference format:
Carlos Garcia Cordero, Emmanouil Vasilomanolakis, Aidmar Wainakh, Max Mühlhäuser, and Simin Nadjm-Tehrani. 2020. On Generating Network Traffic Datasets with Synthetic Attacks for Intrusion Detection. *ACM Trans. Priv. Secur.* 24, 2, Article 8 (December 2020), 39 pages.
https://doi.org/10.1145/3424155

1 INTRODUCTION

Evaluating the detection capabilities of Network Intrusion Detection Systems (NIDSs) is a crucial task in today's Internet age. NIDSs have become an almost mandatory line of defense against attacks due to our dependency on the Internet and the Internet's threat landscape. The 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 NIDSs have 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[1] to assess their capabilities. Quality datasets enable 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 [25]. Most datasets are distributed as Packet Capture (PCAP) 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 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 Application Programming Interface (API) to easily extract, compare and replicate traffic properties.

ID2T aims at being utilized by *researchers* (e.g., for proposing and evaluating novel detection algorithms), *network security specialists* (e.g., for testing their NIDS in a company-level), *educators* (e.g., to teach students how cyber-attacks are created), *cyber-security enthusiasts* and *security organizations*. The toolkit is being already used by more than 10 different universities, as well as by companies and security agencies.

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

[1]A dataset in the context of NIDSs refers to network traffic captures. This is in contrast to machine learning, where a dataset is a collection of attribute-value pairs.

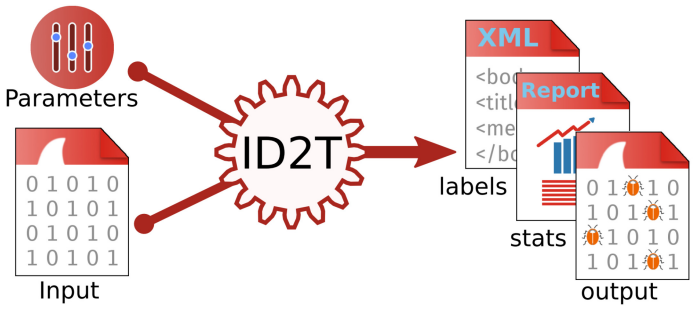


Fig. 1. As inputs, ID2T takes a PCAP file that is used as background traffic and a set of parameters that define attacks to inject and their specifications. As outputs, ID2T creates a new PCAP file with the background traffic injected with synthetic attacks, a collection of statistics related to the background traffic, and a file containing the labels of each injected attack.

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.

1.1 Contributions

This work substantially extends two earlier publications [10, 50]. The first publication presents the idea of ID2T in the form of a poster [10]. 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 [50]. 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[2] software to help NIDS researchers create, distribute and replicate datasets.
- We develop the ID2T module named Testing Intrusion Detection Datasets (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, then 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, then ID2T will only approximate the real effects of the attack. Most denial of service and botnet attacks fall into this category.

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

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, then not all attacks would be labeled. To try and address this problem, ID2T analyses the background traffic provided by the user 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 of 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

We identify eight requirements that relate to the datasets needed in the NIDS field. On the one hand, there are requirements that apply to the datasets themselves. On the other hand, requirements apply to the generation and injection of synthetic attacks. All requirements are derived from related work. Additionally, our defect analysis presented in Section 3.3 adds to the requirements of datasets for NIDSs. The development of ID2T adds to the requirements of generating and injecting synthetic attacks. We classify all requirements into functional and non-functional ones.

2.1 Requirements of Datasets Suitable for the Evaluation of NIDSs

We derive the following requirements from surveys [51], related work [6, 25, 31, 43] and our experience in the field of NIDSs [10, 50, 52]. The following requirements are targeted at making datasets suitable for testing, evaluating and comparing NIDSs. In the following, these requirements are split into functional and non-functional ones.

2.1.1 Functional Requirements. 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. Datasets should include packet payloads so that they are useful to test all types of NIDSs. Many NIDSs rely on deep packet inspection to recognize intrusions. Due to privacy, packet payloads are often unavailable. Maintaining privacy is important but datasets without packet payloads limit the capability of NIDSs to detect attacks at the payload level (e.g., application layer exploits).
(2) Labeled Attacks. Datasets must be labeled such that the individual packets of an attack are associated to a label.
(3) Ground Truth. Dataset labels must be accurate. This implies that there is no benign traffic labeled as an attack and that there are no attacks without an associated label. Without ground truth, we cannot compare different NIDSs with conclusive results.
(4) Renewable. Datasets must be updated to include new attacks and emergent network patterns. With the speed network protocols and patterns change, a dataset can only remain relevant if it is continuously renewed.

(5) Flexible. Datasets must have the ability to test different scenarios with distinct scopes, both in the context of detection and size (e.g., anomaly or signature-based detection in office, or backbone network environments).

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 Replicability. Datasets should be public or, if not possible, easy to replicate. Many datasets, although originally public, are now difficult to obtain due to lack of maintenance from the side of the creator. Releasing a dataset to the public is, therefore, desired.
(2) Interoperability. Datasets should be shared using a common and widespread format. The most popular file format for sharing network packet captures is the PCAP file format.
(3) Quality. Datasets need to actively minimize or avoid creating issues and defects. Quality datasets reduce the bias that may exist when evaluating the capabilities of NIDSs. We distinguish between issues that arise from the generation of real traffic, and from the creation of synthetic traffic. In Section 3.3, we detail and categorize issues found in commonly used datasets.

2.2 Requirements for Injecting Synthetic Attacks

Tools need to take into account the requirements proposed in this section to facilitate the task of injecting synthetic traffic. The following functional requirements are derived from observations of how datasets are compiled (e.g., Reference [43]) and how other tools create synthetic traffic (e.g., Reference [7]). The non-functional requirements are targeted at tools capable of processing arbitrary PCAP files.

2.2.1 Functional Requirements. We derive the following functional requirements from observations of how datasets are manually created. To alleviate the manual task of creating tailor-made labeled datasets with attacks, tools that generate synthetic traffic need to take several functional requirements into account. These next requirements focus on what tools need to customize and create synthetic attacks.

(1) Packet-level Injection. Synthetic attacks need to be modeled at the packet level (instead of flow level, for example), which is the lowest common denominator for most NIDSs.
(2) PCAP Interoperability. Synthetic attacks need to be injected into arbitrary PCAP files. To do this, attacks need to adjust their properties to match the statistical properties of the traffic in the PCAP.
(3) Minimal User Interaction. Synthetic attacks should be created without much user interaction. A user should be expected to only specify an initial set of parameters that are enough to generate attacks. The generation process needs to be deterministic to enable others to replicate a set of attack injections.
(4) Packet-volume or Payload-based Attacks. Synthetic attacks should be created to model either packet or payload-based attacks. Attacks such as exploits rely on the ability to model the payloads of packets while Distributed Denial of Service (DDoS) attacks need to alter the number of created packets.

2.2.2 Non-functional Requirements. Synthetic attack injection tools need the following non-functional requirements to process arbitrary PCAP files. When not being limited to certain PCAP files, the main challenge 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

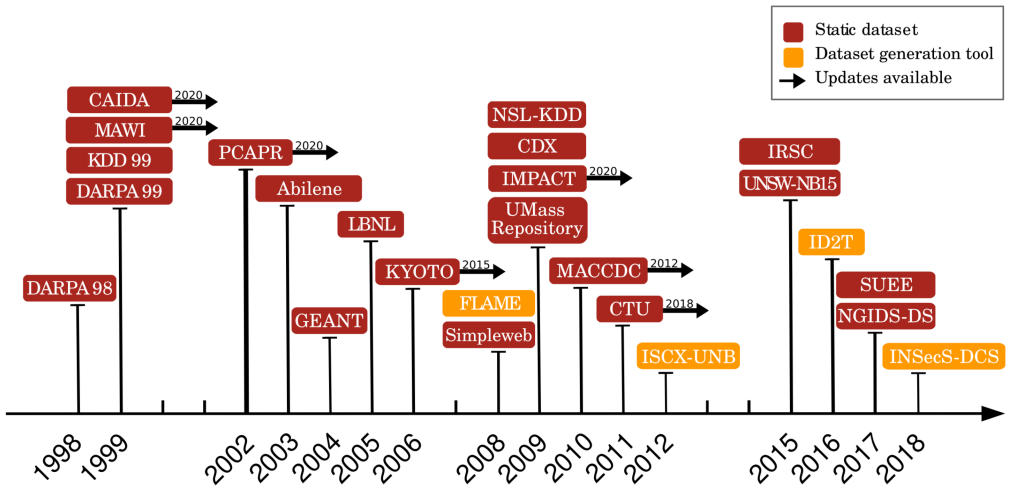


Fig. 2. Timeline of the published year of static datasets and dataset generation tools. We show static datasets in red blocks and dataset generation tools in yellow blocks. An arrow to the right of a block indicates that updates exist for the dataset.

contrast, large enterprise or university networks may generate massive quantities of data with heterogeneous characteristics.

(1) Scalability. Injection tools need to process capture files of large networks. Therefore, the parsing and collection of statistics need to be efficient.
(2) Extensibility. Injection tools need to be easily extended with new attacks to cope with the evolving threat landscape.
(3) Usability. Injection tools should not require specialized hardware to create synthetic traffic. At the same time, users should be involved as little as possible in the injection process.
(4) Open Source and Public Availability. Injection tools, to be useful to the community, need to be public and open source. Many tools and datasets become inaccessible or are lost, because their creators stop maintaining them. For example, the tools proposed by Shiravi et al. [43] and Brauckhoff et al. [7] 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 NIDS to detect anomalies [6], preferably allowing to draw valid conclusions about the appropriateness of the NIDS. Although many of the currently available datasets are valuable to the research community, they unfortunately fail to fulfill all the requirements we propose in Section 2.

Existing datasets can be grouped into the categories of *static* datasets and *dynamically generated* datasets. In Figure 2, we use a timeline to show the publication year of the 18 static datasets and tools we analyze. An arrow to the right of a dataset name indicates that the dataset has received updates after it was first published. The selected datasets have been specifically created for the purpose of NIDS evaluation or can be practically used for that purpose. Note that we consider datasets for other purposes (e.g., Host-based IDS) out of the scope for this article; nevertheless, for such datasets, we refer the curious reader to References [21, 42]. In the following, we provide basic information about some of the most popular static datasets and dataset generation tools. Afterwards, we discuss the deficiencies and artifacts found in datasets.

3.1 Static Datasets

We define static datasets as those generated either from real or synthetic traffic that, once created, are not altered. Many such datasets exist, each with different degrees of popularity and deficiencies. In the following, we briefly describe each static dataset. Subsequently, we present a summary and comparison table of all presented datasets.

DARPA 98 and 99. Created by the Lincoln Laboratory of the MIT to enable an offline intrusion detection evaluation challenge set by DARPA, these datasets are still widely used in spite of their age [1]. The datasets consists of records of simulated traffic (of hundreds of users) between a United States Air Force base and the Internet. It was a first attempt at creating an "objective, repeatable and realistic [dataset]" [13]. The dataset contains different threats that range from network scans to privilege escalation exploits. Five weeks of data are provided: 2 weeks of normal traffic, 1 week of labeled attack traffic, and 2 weeks of unlabeled attack traffic. The datasets have been found to contain many defects that may bias the detection capabilities of anomaly detectors [30]. For example, as a notable defect, all network packets that belong to an attack have a fixed Time To Live (TTL) value.

KDD 99. The KDD 99 dataset was created in 1999 for a competition aimed at developing NIDS [24]. The dataset is a collection of records of network flows extracted from the Defense Advanced Research Projects Agency (DARPA) 98 dataset. Each record contains 31 features. With such a dataset, developers do not need to engineer features and can fully concentrate on developing NIDSs. The KDD 99 dataset is still used today despite it being outdated and having well known issues [12].

MAWI. The MAWI dataset [15] is a collection of PCAP files containing backbone network traffic flowing between Japan and United States. The dataset exists since 1999 and, as of today, is updated almost every day with new PCAP files. Each PCAP contains 15 min of anonymized traffic and, due to the amount of recorded traffic, may be up to 20 GiB in size. Despite the advantage of its recentness, the dataset has other limitations that hinder its usage in the field: packet payloads are not available, IP addresses are inconsistently scrambled among different PCAPs, and ground truth is not available. Anomaly detectors are used to label the PCAPs in an attempt to leverage the inherent problems of labeling real network traffic.

CAIDA. The organization known as the Cooperative Association for Internet Data Analysis (CAIDA) [8] gathers, anonymizes and publishes network traffic captures of different purposes. Six datasets are available for the specific purpose of assisting security researchers, each with specialized attacks or anomalies. However, the datasets are limited in scope and have no labels.

PCAPR. This is a repository [48] of PCAPs created by the company Spirent. The purpose of the repository is to enable users to share PCAP files. There are more than 3,000 PCAP files in the repository that contain DDoS attacks, exploit traffic or regular traffic, among other things. The repository does not restrict the types of PCAPs that can be uploaded and many PCAPs are not useful for the purpose of evaluating NIDSs. Yet, the mere abundance of PCAP files is noteworthy.

Abilene. This dataset, created in 2003 by Lakhina et al. [26], was used to evaluate an anomaly detector based on traffic feature distributions. The dataset contains 3 weeks of sampled traffic flow data captured from the Abilene network. The network is an Internet2 backbone network, connecting universities and research centers in the USA, Europe and Asia. The sampling rate for the flows was 1 out of 100 packets and the destination and source IP addresses are anonymized. Although this dataset had been used frequently by the research community, it has become outdated by modern networking standards. Because the dataset only contains flows, it cannot be used by NIDSs that process PCAPs.

GEANT. The GEANT dataset consists of 3 weeks of sampled flow data. This data was collected by Lakhina et al. [26] in 2004 from GEANT. GEANT is a backbone network that interconnects national research and education networks across Europe. The dataset was collected under an Non-Disclosure Agreement (NDA) and, therefore, it was not published. The traffic of GEANT contains mainly academic traffic and is described by its authors as suitable to validate anomaly detection systems.

LBNL. This dataset, created by the Lawrence Berkeley National Laboratory (LBNL) and Berkeley's International Computer Science Institute (ICSI), contains around 100 h of traffic activities recorded from a large enterprise network. With a size of 11 GiB, the traffic consists of background network traffic with no known attacks. The dataset has been used to analyze large networks [34], but has a limited scope in the context of NIDSs. The dataset is now outdated (having been collected in January 2005) and does not contain labels.

KYOTO. The Kyoto dataset [45], rather than being a collection of network captures, consists of a series of records of features extracted from traffic captured by honeypots. The honeypots are deployed in the University of Kyoto and implement advanced mechanisms of disguise [46]. The dataset has been updated up to December 2015. The records use the same 14 features as those used in the KDD 99 dataset plus 10 additional features related to the output of diverse Intrusion Detection Systems (IDSs). Ground truth is not available and the original packet captures are not provided.

Simpleweb. In an attempt to publish open datasets to evaluate and compare NIDSs, the Simpleweb/University of Twente dataset was created [4]. The dataset consists of eight *traces*, with varying formats, of network traffic captured in a university network. All traces have been anonymized and, if applicable, stripped of their payloads. One trace, from the eight available, consists of features extracted from network traffic that passed through a honeypot [51]. All traffic from this trace is considered (and labeled) as malicious [47]. The other traces do not contain labels.

UMass Repository. The Laboratory for Advanced System Software from the University of Massachusetts has a public repository of, among many things, heterogeneous network traces [28]. The traces are collected from specialized scenarios, reflecting specific network configurations, threats, situations and attacks. The format and labels of the traces vary. Most of them are synthetically generated and the ground truth may or may not be available.

IMPACT. The community known as the Information Marketplace for Policy and Analysis of Cyber-risk & Trust (IMPACT) provides diverse datasets related to cyber-security [22]. The datasets are provided by several partners in different formats including PCAPs files. Recent and old datasets exist with different combinations of labeled attacks or synthetic traffic. Access to the datasets is restricted to researchers from the USA and other authorized countries.

CDX. Created in 2009 from the Cyber Defense Exercises (CDXs) competition [41], the CDX dataset provides labeled traces of network traffic [2]. All traffic originating from participants in charge of compromising other systems is labeled as malicious. All other traffic is considered normal or benign. Although the captured traffic consists of network traces of human activities, it does not conform to the expectations of realistic traffic. In particular, the ratio of malicious and benign traffic is almost the same, and the traffic is limited to only 4 days of captures (the duration of the competition).

NSL-KDD. After a statistical analysis of the KDD 99 dataset, Tavallaee et al. [49] identified issues responsible for biases that would negatively affect the performance of anomaly detection

mechanisms. The NSL-KDD dataset attempts to alleviate these issues by creating a new dataset from specifically chosen records of the KDD 99 dataset.

CTU. This is a collection of datasets published as part of the Stratosphere IPS project [16] at the Czech Technical University between 2011 and 2018. The datasets consist of traffic captures from various networks, and are classified into four categories: malware [17], normal, mixed, and malware on IoT traffic. The datasets are labeled and available in the PCAP file format. However, the malicious traffic in these datasets is limited to malware-generated traffic and contains no other attacks.

MACCDC. This dataset was created in the Mid-Atlantic Collegiate Cyber Defense Competition (MACCDC) [9]. The competition is organized annually by the National CyberWatch Center since 2006. The available dataset [35], however, only consists of three years of the generated traffic (2010, 2011, and 2012) in the PCAP file format. The dataset contains a variety of attacks and simulated traffic of the business activities of IT staff. The traffic is not labeled. Many derivatives of this datasets exist in varying forms, such as in Snort and Bro logs [42].

UNSW-NB15. Motivated by the lack of publicly available and modern datasets that reflect modern network traffic scenarios, Moustafa et al. [33] created the UNSW-NB15 dataset in 2015. The dataset was generated using a mixture of network test beds and synthetic traffic generation tools. Nine different attack families are included, all of which are labeled. The dataset is provided in the PCAP file format as well as a record of network features (in the same style as the KDD 99 dataset).

IRSC. Created in 2015 by the Indian River State College (IRSC), this dataset consists of real controlled and uncontrolled attacks carried out in a production network [53]. The dataset consists of full network traces in the PCAP file format along records of network flows. Labels are provided for all controlled attacks while some uncontrolled attacks are labeled through the usage of the Snort IDS. The dataset does not look to be publicly available as of now.

NGIDS-DS. Haider et al. [20] proposed a metric to evaluate the realism of datasets for IDS. Taking into account the qualities that improve their proposed metric, they developed a synthetic and labeled dataset with a medium-high score (according to their metric). The dataset is provided in the PCAP file format along with labels and the logs of each network host. The dataset emulates five different network scenarios: e-commerce, military, academia, social media, and banks.

SUEE. The SUEE dataset [29], created in 2017, contains traffic captured from and to the web server of the Student Union for Electrical Engineering at Ulm University. The dataset is available in the PCAP file format. The traffic captures only contain packets destined to ports 80 and 443. The captures also mix traffic generated by several attack tools. As such, the dataset is labeled.

3.1.1 Requirement Compliance of Static Datasets. In Table 1, we summarize the compliance of the static datasets we survey against the requirements we propose in Section 2. The columns of the table correspond to each proposed functional and non-functional requirements for static datasets (except the last column). Our last non-functional requirement, that of *quality*, is replaced with a different one as *quality* does not lend itself well to be summarized as a check mark. In Section 3.3, we discuss in detail the quality issues of different datasets. The *genuineness* attribute of a dataset replaces the *quality* requirement. A genuine dataset is one that is generated using traffic of a real network environment. A synthetically created dataset is not considered *genuine*. In the datasets we survey, we found that the *genuineness* of a dataset correlates with its quality: Synthetic datasets contain more issues or defects that genuine ones. Therefore, the *genuineness* attribute works as a heuristic that determines the quality of a dataset.

Table 1. Summary Requirements of 14 Different Static Datasets

Dataset	Payloads	Labels	Ground Truth	Renewable	Flexible	Public	Interoperable	Genuine
DARPA 98/99	✓	✓	✓	✗	✗	✓	✓	✗
KDD 99	✓	✓	✓	✗	✗	✓	✗	✗
MAWI	✗	✓	✗	✓	✗	✓	✓	✓[s]
CAIDA	✓	✗	✗	✓	✓	✓[a]	✓[i]	✗
PCAPR	✓	✓[l]	✗	✓	✓	✓[a]	✓	✓
Abilene	✗	✗	✗	✗	✗	✓	✗	✓
GEANT	✗	✗	✗	✗	✗	✗	✗	✓
LBNL	✓	✗	✗	✗	✗	✗	✗	✓
Kyoto	✗	✓	✗	✓[k]	✗	✓	✗	✓
SimpleWeb	✗	✓[l]	✗	✗	✗	✓	✓[i]	✗
UMass	✓	✓[l]	✗	✗	✓	✓	✓[i]	✓[s]
IMPACT	✗	✓[l]	✗	✓	✓	✓[a]	✓[i]	✓[s]
CDX	✓	✓	✓	✗	✗	✓	✓	✓
NSL-KDD	✓	✓	✓	✗	✗	✓	✗	✗
IRSC	✓	✓[l]	✓	✗	✗	✗	✓	✓
CTU	✓	✓	✗	✓[k]	✓	✓	✓	✓
MACCDC	✓	✗	✗	✓[k]	✗	✓	✓	✓[s]
UNSW-NB15	✓	✓	✓	✗	✗	✓	✓	✓[s]
NGIDS-DS	✓	✓	✓	✗	✓	✓	✓	✗
SUEE	✓	✓	✗	✗	✗	✓	✓	✓[s]

[a] Access is restricted or special permissions are needed.
[i] The PCAP file format is not used all the time.
[k] Renewed for a limited period.
[l] Only partially labeled.
[s] Mixed set of genuine and synthetic data.

No single dataset satisfies all our proposed requirements. This attests to the difficulty of creating useful datasets for this field of research. Up to now and according to our requirements, the NGIDS-DS dataset is the most suitable as it is labeled with ground truth, contains full packet information, is flexible (incorporating multiple network scenarios), and is distributed using the PCAP file format. On the negative side, the dataset is not *renewable* nor *genuine*. The lack of renewability is not currently a problem as the dataset is modern (published in 2017); however, the dataset will eventually become outdated. The genuineness attribute is a more subtle topic. The community has strong negative feelings against synthetic datasets. Nonetheless, it is arguable that, for a system to perform well in real network environments, the system must also perform well using synthetic datasets [1].

3.2 Dataset Generation Tools

Static datasets are difficult to maintain and update. This is evident from the fact that most static datasets, as seen in Table 1, have troubles meeting the *renewable* and *flexible* requirement. More often than not, when researchers need modern datasets targeted at specific scenarios, they build custom tools to generate them. In this section, we present three tools for creating custom datasets

and an overview of our tool (ID2T). Afterwards, we summarize how the tools conform to the requirements we propose for tools that inject synthetic attacks to generate datasets.

FLAME. Bauckhoff et al. [7] have developed a tool for injecting anomalies into network flows. The tool known as Flow-Level Anomaly Modeling Engine (FLAME) injects anomalies using one of three different modes. With an additive mode, flows are injected without modifying the background traffic (e.g., port scans). Using the subtractive mode, flows are removed from background traffic (e.g., ingress shifts). Both modes are combined into an interactive mode to add and remove flows (e.g., to simulate congestion due to a Denial of Service (DoS)). Due to Flow-Level Anomaly Modeling Engine (FLAME) being discontinued, the tool can no longer be easily found in the public domain.

ISCX-UNB. Shiravi et al. [43] point out the difficulty of evaluating, comparing and deploying anomaly-based NIDS due to the lack of suitable datasets. In their work, they propose an approach to synthetically generate network traffic from *profiles*. A profile is an abstract model that describes network traffic. They created two types of models for representing profiles, one for legitimate traffic and another for malicious behavior. Legitimate traffic is modeled by replicating the statistics of real network traffic. Malicious behavior is modeled using a custom description language. A test bed is used to generate network packets from profiles. A dataset example is accessible, however, the tool is not readily available to the public.

ID2T. This is the tool we develop and present in this work. The first prototype of this tool was released in 2015 [10]. An improved version of the tool, and first public release, followed in 2016 [50]. The objective of ID2T is to create attacks that replicate the properties of background traffic. Background traffic is provided by the user. Attacks are manually programmed and use the statistics collection mechanism of ID2T to replicate the network properties of any input PCAP file. Many attacks are readily available that range from DoS and network scans to malware and botnet infections.

INSecS-DCS. Rajasinghe et al. [38] have proposed a framework for extracting packet features from network traffic. The framework captures the traffic from a network directly or takes a PCAP file as input. The output of the tool is feature-value pairs in the CSV format suitable for machine learning algorithms. A user can customize the output features by specifying the tracked protocols (e.g., TCP, UDP), packet attributes (e.g., IP, port), and a time window for statistical properties. This tool does not generate network traffic. Instead, it extracts features. Therefore, the user is responsible for feeding the tool with attacks and labeling the attacks manually.

3.2.1 Requirement Compliance of Synthetic Traffic Generation Tools. There is a distinct lack of tools that can generate synthetic attacks suitable for the evaluation of NIDSs. FLAME and *ISCX-UNB* try to achieve similar goals as ID2T but do not meet the requirements we derived as essential for such tools. FLAME is not *available* online anymore despite it being originally publicly released. This is an issue that would potentially be prevented had the software been published as Free and Open Source Software (FOSS) in an open platform. Although the *flexibility* of the tool is limited, as it does not operate at the *packet level*, research that specializes in processing flows would benefit from such a tool. ID2T is more general and generates network traffic at the packet level. Tools that work on flows can use the output of ID2T to extract flows with labeled attacks.

The creators of ISCX-UNB proposed a tool capable of generating synthetic background and malicious traffic. In contrast, ID2T only generates synthetic malicious traffic. The ISCX-UNB tool has the disadvantage, however, that it requires a test bed to model traffic. The test bed is not easy to setup, and requires specialized software and hardware. Due to the test bed, ISCX-UNB has low

Table 2. Summary Requirements of Three Different Datasets Generation Tools

Dataset	Packet Level	Interoperable	Minimal Interaction	Flexible Modeling	Scalable	Extendable	Usable	Public and FOSS
FLAME	✗	✗	✓	✗	✓	✓	✓	✗[f]
ISCX-UNB	✓	✓	✗[i]	✓	✗[s]	✓	✗	✗[d]
ID2T	✓	✓	✓	✓	✓	✓	✓	✓
INSecS-DCS	✓	✓	✓	✗	✓	✗	✓	✓

[d] Only a sample dataset generated by the tool is provided.
[f] Originally publicly available. Cannot be easily found online anymore.
[i] The tool requires a dedicated test bed.
[s] It is not obvious if more profiles requires a larger test bed with more systems.

usability, does not *scale* well to large networks, and lacks an adequate *interactivity*. The tool is not *publicly available* and the quality of its output cannot be compared to ID2T.

The tool INSecS-DCS tries to achieve the goal of ID2T of creating datasets for NIDSs. The scope of the two tools, however, is substantially different. ID2T injects PCAP files with attacks. The PCAP files are left to be processed by the user or NIDS. INSecS-DCS processes PCAP files by extracting features. We include the tool INSecS-DCS in our survey, however, as we recognize its usefulness on top of ID2T: While ID2T creates PCAP files, INSecS-DCS can extract features from those PCAP files to test Machine Learning (ML) algorithms.

3.3 Classification of Dataset Defects

We aim at creating a dataset generation tool that actively avoids the same mistakes found in related work. In this section, we present a classification of the defects found by others and ourselves in the datasets and tools previously presented in Sections 3.1 and 3.2. Our classification is the outcome of the systematic analysis of related work. It serves as a guide to acknowledge potential defects that synthetic attacks may inadvertently leave behind. The classification also enables us to organize solutions to actively detect and circumvent dataset defects. This section is used in Section 6 as a guide to generate synthetic attacks that actively avoid leaving defects behind by considering this classification.

In this classification, we distinguish between two types of defects found in network traffic datasets. We use the term *artifact*, in accordance to related work [1], to refer to defects in a dataset introduced as a side effect of creating synthetic traffic. The term *deficiency* refers to a defect or problem in a dataset that originate from incorrectly using, capturing or recording real network traces.

Figure 3 presents our proposed classification of defects for network traffic datasets. Defects can be divided into two general classes. The *Invalid Network Traffic* class covers defects related to the incorrect usage of network protocols or specifications (e.g., TCP communication with windows of size zero). The *Inconsistent Network Traffic* class encompasses defects associated with the existence of inconsistent or unexpected network traffic (e.g., overly regular packet interarrival times). Furthermore, in our analysis of related work, we observed that some defects may be classified as a defect only in certain contexts.[3] Therefore, classifying something as a defect also depends on

[3] For example, every attack packet having a TTL value of 126 is a defect only if the background packets does not have this same property.

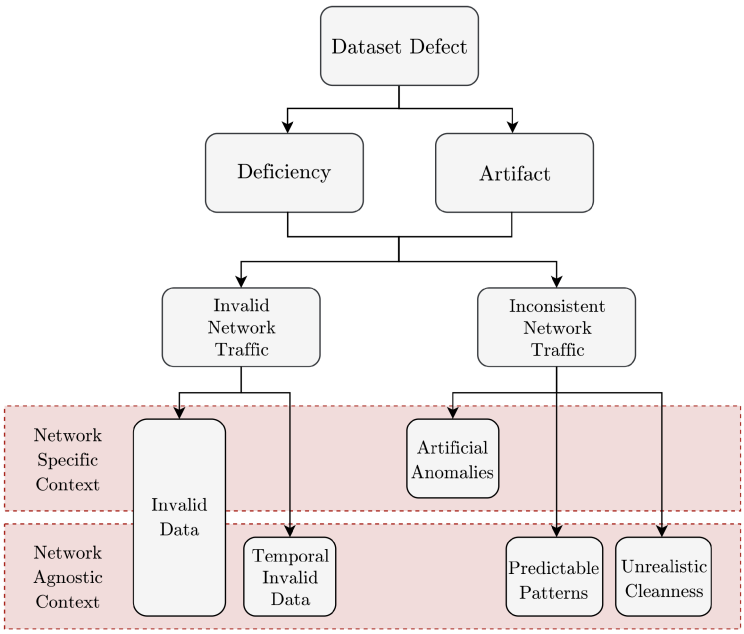


Fig. 3. Classification of dataset defects. Defects may be distinguished as either a deficiency or an artifact. Both defect types may fall into the classes of invalid or inconsistent network traffic. These classes can further be organized into contexts depending on whether a problem is specific or agnostic to a network.

the context. From a *network specific context*, something is a defect only when the characteristics of the network traffic as a whole are taken into account (e.g., the existence of public IPs in a local network). From a *network agnostic context*, something is a defect irrespective of the characteristics of network traffic (e.g., invalid IP addresses). We use five different classes to categorize defects (see Section 3). Each class is described below.

- Invalid Network Traffic
  - Network Specific Context
    - ☐ Invalid Data. Disregarded characteristics or physical limitations of a network in its captured representation.
  - Network Agnostic Context
    - ☐ Invalid Data. Violation or incorrect usage of network protocols.
    - ☐ Temporal Invalid Data. Incorrect usage of protocols due to protocol deprecation or renewal.
- Inconsistent Network Traffic
  - Network Specific Context
    - ☐ Artificial Anomalies. Anomalous data patterns that do not correspond to the overall characteristics of the network traffic.
  - Network Agnostic Context
    - ☐ Predictable Patterns. Network characteristics that repeat with regular periodicity, without much variance, against the expected behavior of a network.
    - ☐ Unrealistic Cleanness. Network characteristics that do not behave as expected, are too regular or do not exhibit the typical untidy behavior [5, 36] of network communication.

In the following, we demonstrate how this classification applies to the known defects found in the DARPA 99 dataset. The TCP version used in the dataset is old [37], making the TCP header field "IPv4 Type of Service (ToS)" invalid according to modern standards [19] *(temporal invalid data*). The dataset uses only nine different TTL values that follow an uncommon network pattern *(predictable pattern)*. Furthermore, Mahoney et al. [31] noticed that the packets of all attacks in the dataset use one of two different TTL values *(artificial anomaly)*. As identified by McHugh [32], although an attempt was made to add some of the untidy behavior typical of TCP communication, in the form of fragmented packets, the dataset does not show the expected characteristics of irregular network behavior [36] *(unrealistic cleanness)*. The data rates of the dataset are invalid if the theoretical network configuration, from where the dataset was generated, is taken into account [32] *(invalid data, from a network specific context)*.

4 THE INTRUSION DETECTION DATASET TOOLKIT (ID2T)

ID2T is a public and open source[4] command-line toolkit that injects synthetic attacks into (network) traffic captures. To achieve this, the toolkit first analyzes and collects statistics from a traffic capture. The statistics are then used by attack scripts to set internal parameters, and to create attack packets that generally replicate the statistical properties of background traffic. Some attack scripts are meant to generate packets that distinctively alter background traffic statistics, and ID2T facilities this. For example, ID2T's DDoS attack script may significantly change the packet size distribution of a network (among other properties) while retaining many other statistical properties (such as the packet's TTL distribution). Finally, ID2T merges the attack packets with the packets of the input. The outputs of ID2T are a PCAP file with injected attacks and a file of labels that clearly indicate when attacks start and end. If needed, then ID2T can watermark or taint packets to associate each packet to an attack.

The main architecture of ID2T was presented in References [10, 50]. In this work, we extend the *Attacks* module to include a wider range of attacks. Generating additional attacks with different characteristics requires computing additional statistical properties of the background traffic; therefore, we extend the *Statistics* module. In addition, we develop a new module named *Testing Intrusion Detection Datasets (TIDED)*, which analyzes the properties of input traffic to identify potential defects and produces a report accordingly. In this section, we provide a detailed description of ID2T and its extended architecture.

4.1 The Architecture of ID2T

In this section, we present the architecture of ID2T. ID2T is a toolkit aimed at injecting synthetic attacks into PCAP files of network traffic. To blend synthetic attacks with arbitrary input PCAP files, ID2T attempts to match the statistical properties of synthetic network traffic with the statistical properties of the input. Others have illustrated the usefulness of replicating network traffic properties to generate synthetic traffic [14, 43]. In addition to replication, ID2T enhances the quality of synthetic traffic by actively avoiding the defects presented in our classification (see Section 3.3). The architecture of ID2T is therefore built to enable the two strategies of replicating statistics and avoiding defects (see Section 6). In addition, the architecture provides the tools needed to alter or create new attack scripts that follow both strategies.

Figure 4 illustrates the architecture of ID2T. ID2T processes two inputs and, using seven modules, yields three different outputs. The *inputs* include a PCAP of network traffic and a set of

[4]See footnote 2.

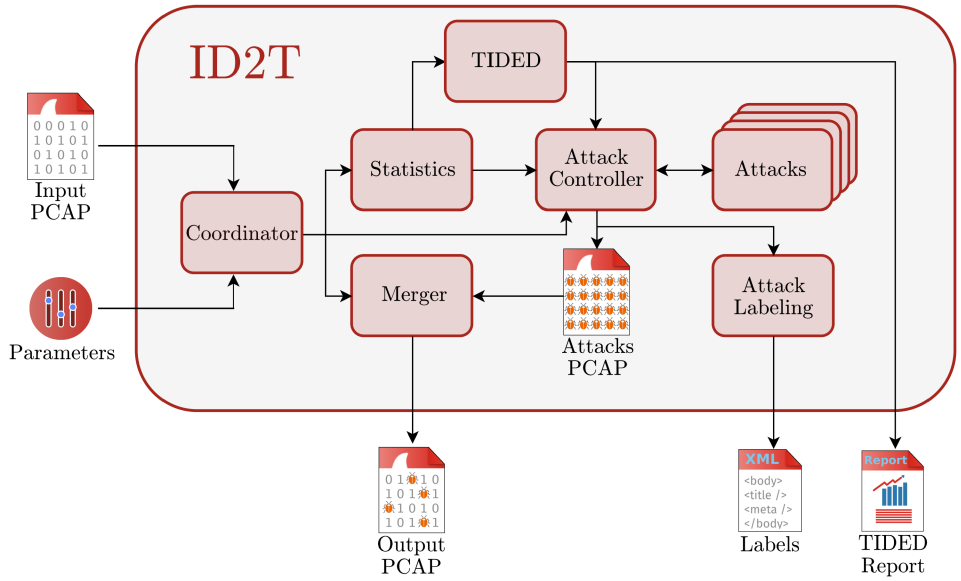


Fig. 4. The architecture of ID2T is composed by seven modules. These modules carry out the tasks needed to create a labeled output PCAP (with its statistics) that mimics the input PCAP in accordance to some user-defined parameters.

parameters. The input PCAP has no restrictions about its size or provenance; however, IPv6 traffic is not considered.[5] The parameters specify one or many attacks to inject and their respective properties.

The *outputs* of ID2T include a copy of the input PCAP merged with packets created by attack scripts, a human-readable XML file with labels and a report of the input statistics. The labels indicate when an attack started and finished along with all user-supplied or automatically calculated properties of the attack. It is also possible to taint all individual packets, e.g., by manually specifying a known MAC address. However, this process is mostly intended for debugging purposes, since packet tainting may introduce artifacts.

The statistics output shows a quantitative characterization of the PCAP input. This characterization includes the values of different properties such as the average packet rate, average packet size, total bandwidth used, IP address entropies, and TTL distribution, among others. When appropriate, these statistics are also reported on an interval rather than an input-wide basis. All the available statistics are detailed in Section 5.

The *modules* of ID2T are responsible for generating the outputs from the inputs. The following section describes each module in detail.

4.2 The Modules of ID2T

ID2T is a multi-modular application built to be easily extendable. Modules are decoupled from each other and perform a specific set of tasks. In this section, we describe each of the seven modules,

[5]Besides its name, IPv6 does not share many things with IPv4. The protocols function in different ways. Therefore, the research presented here does not directly translate to work with the IPv6 protocol.

their tasks and how they interact with each other. Figure 4 illustrates each module as a small block inside the main block of the architecture.

Coordinator Module. This user-facing module is responsible for parsing the command-line arguments issued by the user. The module coordinates with the *Statistics*, *Attack Controller*, and *Merger* modules to perform the tasks of injecting attacks, plotting the statistics of the input or listing the properties of each attack.

Statistics Module. This is a backend module responsible for collecting and computing statistics from an input PCAP file. These statistics are collected for two purposes. On the one hand, to create packets; on the other hand, to test input PCAPs. Testing PCAPs is the responsibility of the TIDED module, which is detailed in Section 5. The statistics related to the creation of packets are divided into four categories: First, host-specific statistics, i.e., collected per host (e.g., number of sent and received packets, maximum and minimum packet rate). Second, protocol-specific statistics, i.e., collected per protocol (e.g., number of packets and bytes for each protocol). Third, conversation statistics, i.e., collected for each network flow (e.g., average packet inter-arrival time per network flow). Fourth, and finally, field-specific statistics, i.e., related to a particular field in the IP, UDP and TCP headers (e.g., used ports, TTL distribution, average Maximum Segment Size (MSS)). The complete list of collected and computed statistics is described in the website of ID2T.[6] The *Controller* module enables users to directly interact with the *Statistics* module though a mode termed *query mode*. In query mode, users can query for predefined information or generate custom information queries. The *Statistics* module also provides an interface to the *Attack Controller* and *TIDED* modules. Through this interface, the modules can query information programmatically to carry out their operations (i.e., create synthetic attacks that replicate the statistics of the input).

TIDED Module. This is a backend module that performs qualitative tests on the input PCAP using data from the *Statistics* module. The objective of the tests is to provide measurements that may expose deficiencies (see Section 3.3) and sources of problems in the input PCAP file. The tests fall in one of the categories of availability, validity, or diversity. A detailed description of *TIDED* is given in Section 5.

Attack Controller Module. This backend module instructs attack scripts, via the *Attacks* module, to create the packets of synthetic attacks. This module validates the attack parameters supplied by the user. It also enables the attack scripts to obtain information from the *Statistics* or *TIDED* modules. After the *Attacks* module generates packets, this module creates a consolidated PCAP file (of only attack packets) that is forwarded to the *Merger* module. Information about the attacks in the consolidated PCAP is forwarded to the *Attack Labeling* module to enable it to create a file with attack labels.

Attacks Module. This module is both a user-facing and a backend module. From the side of the user-facing part, this module provides an API for users to program attack scripts. The API has functions to facilitate the programming of attacks that replicate the properties of an input PCAP file. The programmed attacks become available to the user by registering them with the *Attack Controller* module. From the side of the backend part, this module configures attacks given the set of parameters specified by the user. ID2T comes with 12 programmed attacks, each detailed in Section 6.

Attack Labeling Module. This backend module is responsible for generating an XML file containing the labels of injected attacks. Based on the information received from the *Attack Controller*

[6] https://github.com/tklab-tud/ID2T/wiki/Extended-ID2T-Statistics-DB:-Tables.

module, labels for each attack are generated including: attack name, number of injected packets, time of injection (i.e., timestamp of the first and last attack packet), and user-specified parameters (e.g., IP addresses and ports of attackers and/or victims). Individual packets are currently not labeled. However, the architecture of ID2T does not limit programming such a feature. The file is meant to be readable by machines and humans (see Listing 1).

```
<labels version_parser="0.3">
  [...]
  <name>DDoSAttack</name>
  <injected_packets>10640</injected_packets>
  <timestamp_start>
    <timestamp>1554613500.00104</timestamp>
    <timestamp_hr>2019-04-07 05:05:00.001040</timestamp_hr>
  </timestamp_start>
  <timestamp_end>
    <timestamp>1554613501.0076916</timestamp>
    <timestamp_hr>2019-04-07 05:05:01.007692</timestamp_hr>
  </timestamp_end>
  <parameters>
    <ip.src user_specified="True">129.128.127.126</ip.src>
    <ip.dst user_specified="True">133.7.133.7</ip.dst>
    <attackers.count user_specified="True">1</attackers.count>
    <packets.per-second user_specified="True">5320.0</packets.per-second>
    [...]
  </parameters>
</labels>
```

Listing 1. Excerpt of an ID2T label file.

Merger Module. This backend module merges together the attacks supplied by the *Attack Controller* module and the input PCAP to create an output PCAP file.

4.3 ID2T's Performance

This section presents how ID2T performs with respect to processing network captures and injecting them with synthetic attacks. Our performance results show that the architecture of ID2T does not contain bottlenecks that negatively affect its performance and achieve their intended objectives. In the following, we illustrate three performance indicators of ID2T: its statistics calculation performance, its synthetic packet generation performance, and its ability to inject synthetic attacks that replicate background traffic properties.

4.3.1 Statistics Calculation Performance. One of our proposed requirements (see Section 2) is to handle arbitrary files. Hence, it is important to examine how the architecture of ID2T handles large files. The only bottleneck to handle large files is located in the *Statistics* module and the time the module takes to analyze packets. Figure 5 shows the time required to analyze and extract statistics from a network traffic capture coming from the Measurement and Analysis on the WIDE Internet (MAWI) dataset with different number of packets. To test the performance of the statistics module to analyze different packet numbers, we split the capture that corresponds to the date 2015/06/02 of the MAWI dataset[7] into files that contain different packet numbers. The split files range from six to 212 million packets (of 0.5 to 18 GiB). All split files are heavily loaded with traffic, i.e., the original file has 212 million files that were captured in the span of 15 min.

For each file, we examine two different sets of measurements. The first set of measurements, known as *basic statistics*, are related to the collection of simple statistics (e.g., mean number of packets or standard deviation per IP). The second set of measurements, known as *advanced statistics*, take into account heavy statistics that can be computed of network traffic. These statistics

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

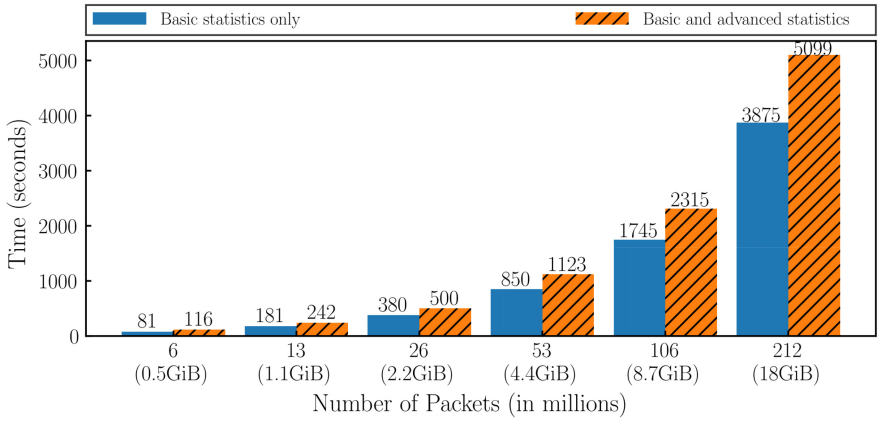


Fig. 5. Analysis of the performance of the statistics module of ID2T to compute statistics. The plot shows with blue bars the speed (in seconds) for computing basic statistics. In orange, the plot shows the time (in seconds) taken to compute, both, basic and advanced statistics.

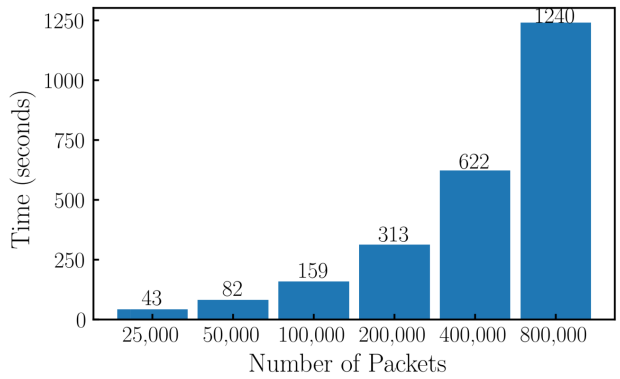


Fig. 6. Analysis of the time (in seconds) that ID2T needs to create different number of synthetic packets. In this example, we show the performance of creating the synthetic packets of DDoS attack.

include, but are not limited to, calculating the entropies of IP addresses, examining the correctness of TCP checksums, or checking the availability of packet payloads. Many of the advanced statistics come from the *TIDED* module, and are further explained in Section 5. At a glance, Figure 5 suggests that PCAP files of 18 GiB in size can be analyzed by ID2T in between 3,875 and 5,099 s, depending on which statistics are used. To create most attacks, it is sufficient to only compute basic statistics. An attack that requires advanced statistics (e.g., the peer-to-peer bot creation attack) tells the user if the advanced statistics need to be calculated if they are not available.

4.3.2 Attack Packets Generation Performance. ID2T creates attack packets that mimic the properties of background traffic according to the times shown in Figure 6. The process of creating packets is considerably heavy and is carried out in two main steps. In the first step, ID2T computes diverse statistics for a given background traffic capture. In the second step, ID2T creates packets that mimic many of the statistics computed during the first step (when appropriate). In the previous section, we showed the performance of collecting statistics. In this section, we show the performance of creating packets alone. For the tests shown in Figure 6, we have chosen to generate packets using the properties of a heavily loaded background traffic capture (with 212 million

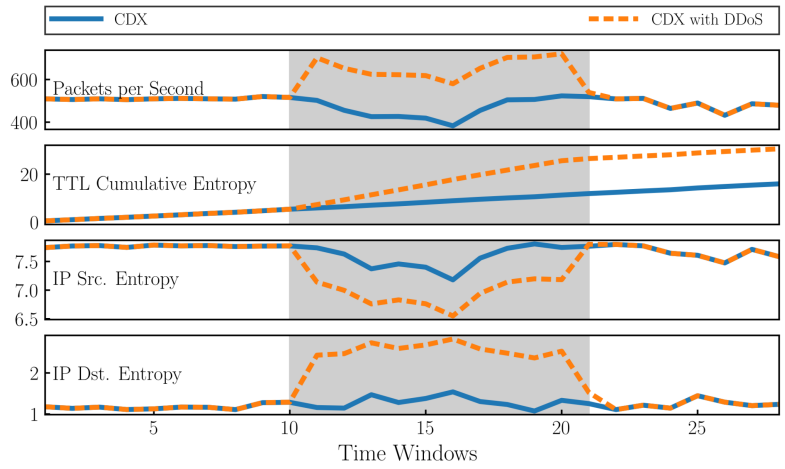


Fig. 7. Statistics of 3 min of the CDX dataset split into 30 time windows. The statistics are shown for a traffic capture of the dataset without and with a DDoS attack injected by ID2T between the time windows 10 and 21.

packets in a span of 15 min). Despite the large number of packets, for which their properties must be replicated, ID2T needs 1,240 s (20 min) to create 800,000 packets. The packet generation performance is in fact linear. Close to 30% of the time spent in the packet creation process comes from saving packet batches on disc to avoid using all available RAM memory. Consequently, faster hard drives (e.g., such as SSD drives), should speed up the packet creation process.

The packet creation performance of ID2T, i.e., creating 800,000 packets in 20 min, is not impressive. However, ID2T does not aim at creating packets in real time. Instead, ID2T tries to create reusable datasets that contain as few artifacts as possible. The process of creating a single packet is slow as many statistics need to be consulted (which sometimes cannot be cached), distributions approximated, and values interpolated. Note that only few attack scripts (e.g., the DDoS attack script) need to generate more than tens of thousands of synthetic packets. Most other attack scripts need to create less than one thousand packets to replicate their attacks.

4.3.3 Replication of Background Properties. ID2T provides tools that attack scripts can use if they wish to replicate background traffic properties in synthetic traffic without creating artifacts. A DDoS attack script, for example, may want to replicate the maximum bandwidth or TTL entropy of some background traffic, but not the packets per second rate. We illustrate the ability of ID2T to inject attacks that do not leave artifacts by comparing how the same attack looks in two different datasets.

Figures 7 and 8 put in contrast the traffic properties of a dataset in its original form to itself with an injected attack. In both figures, the attack is the same[8] DDoS and is injected in the shaded interval. The DDoS consists of ten attackers SYN flooding a single victim at a minimum rate of 100 packets per second for 60 s. Figures 7 and 8 apply, respectively, to the CDX dataset and the DARPA dataset. Note that both figures show that the *packets per second* property generally respects the shape of the background traffic without following it exactly. The IP source entropy decreases in the CDX datasets, while in the DARPA dataset it increases. This is because the time windows of the CDX dataset have hundreds of packets more than those of the DARPA dataset.

[8]We define two injected attacks to be the same if the exact same parameters are used to create both attacks.

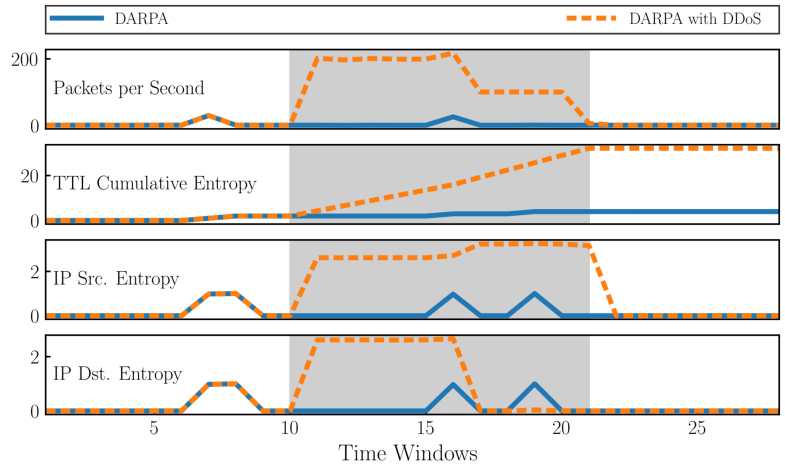


Fig. 8. Statistics of 3 min of the DARPA dataset split into 30 time windows. The statistics are shown for a traffic capture of the dataset without and with a DDoS attack injected by ID2T between the time windows 10 and 21.

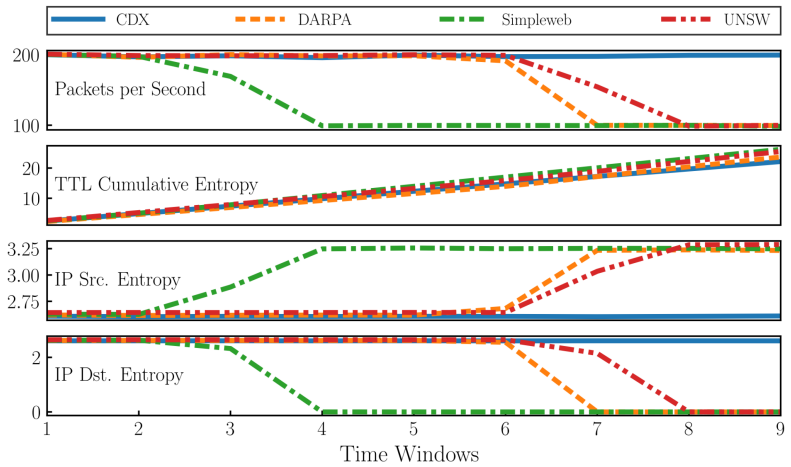


Fig. 9. Comparing the properties of the packets created by the same attack using different datasets as background.

To avoid leaving artifacts behind, the attack scripts of ID2T create packets stochastically taking into account the statistical properties of background traffic. Figure 9 shows how ID2T creates attack packets with different statistical properties for the same attack[9] when ID2T uses different datasets as background traffic. The figure shows the properties of *only* attack packets without the background traffic. The shown properties apply to four different datasets. Despite injecting the same DDoS, note how there are no patterns that repeat (i.e., no obvious artifact is left behind).

5 TESTING INTRUSION DETECTION DATASETS (TIDED)

Quality datasets are essential to the development, comparison and evaluation of NIDSs. The overall quality of a dataset not only depends on the quality of the injected attacks but also on the quality

[9]See footnote 8.

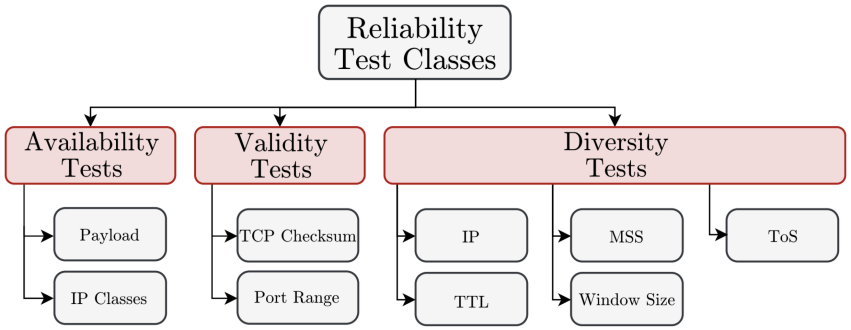


Fig. 10. Classification of the reliability tests implemented in TIDED.

of the background traffic. The TIDED module of ID2T performs tests on the background traffic to identify potential artifacts or deficiencies (see the classification in Section 3.3).

We assign the term *reliability tests* to the tests implemented in the TIDED module. The tests can be used to spot deficiencies in the background traffic supplied as an input to ID2T. We borrow the term *reliability tests* from the field of computer systems. From the perspective of this field, reliability refers to the probability that a system satisfactorily performs the tasks for which it is designed. Analogously, we deem NIDS databases reliable if they satisfactorily perform their intended task (i.e., provide valid grounds for the evaluation of NIDSs). Thereafter, we use the term *reliability tests* to denote tests that may uncover aspects that hamper the reliability of an NIDS dataset.

The purpose of TIDED is twofold. On the one hand, it uses reliability tests to validate the conformance of the network traffic to some expected properties (e.g., the distribution of IP addresses conforming to supposedly backbone network traffic). On the other hand, TIDED makes the results of the reliability tests available to the attack scripts (through the *Attack Controller* module) to enable a better replication of the properties of background traffic. The TIDED module also confers ID2T the characteristic of a network dataset analysis tool.

We derive nine reliability tests from our survey and analysis of related work (see Section 3) that correlate with the type and purpose of a network (e.g., for office work, for home use, or for backbone data transfer). The tests are designed to detect the artifacts or deficiencies, which we have classified in Section 3.3. The tests are classified into different classes. In the following, we present the different classes and the related tests. Afterwards, we describe the metrics used by the reliability tests and illustrate examples of the usefulness of the tests.

5.1 Classification of Reliability Tests

We use three different classes to categorize the reliability tests of TIDED. Figure 10 shows a diagram of the nine implemented tests. The tests use the statistics collected by ID2T from a given input PCAP file and do not need to directly analyze the file. If needed, then ID2T provides an API to create new tests. The currently implemented tests are detailed next.

- Availability Tests. This is a class of tests that encompasses all tests that verify the existence of a property or attribute in network traffic captures.
  - Payload Availability. This test verifies if the packets of a network traffic capture contain payloads or not. If payloads are available, then the test tracks the entropy of the payloads to deduce and report on the amount of encrypted payloads. An attack script may use this test to, among other things, simulate encrypted payloads.

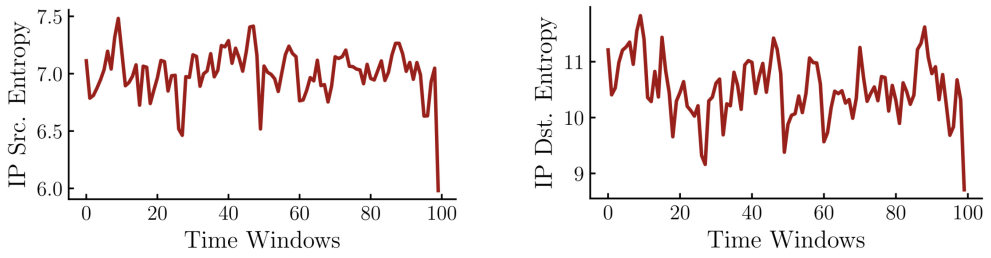
- IP Classes Availability. This test identifies which classes of IP addresses a network capture contains. The test specifies how public and private addresses mix and by which ratio.

- Validity Tests. This is a test class that corresponds to the tests that identify consistency issues within network traffic captures.
  - TCP Checksum Validity. This test identifies the amount of TCP checksum errors in a network traffic capture. The test reports the ratio of checksum errors.
  - Port Range Validity. This test keeps track and reports the distribution of all ports used as either the source or destination of a TCP connection within a network traffic capture. Additionally, the test reports the distribution of ports within the three port ranges defined by the Internet Assigned Numbers Authority (IANA) in accordance to RFC 1700 [39] and RFC 6335 [11]. The test also keeps track of packets that target port zero, which is not defined as a valid port by the IANA but is often seen being used in the wild.
- Diversity Tests. This is a class that relates to the tests that determine the correct or incorrect behavior of TCP header values in network traffic captures.
  - IP Distribution. This test identifies if the distribution of source and destination IP addresses correspond to expected patterns of normality within a network traffic capture. This is an important test, because IP address distribution correlates with the network type (e.g., home, office or backbone network).
  - TTL Distribution. This test measures the distribution of TTL values in TCP header fields and looks for unexpected deviations in a network traffic capture. This value correlates with the behavior of operating systems and hardware devices (i.e., different systems and devices modify the TTL values differently).
  - MSS Distribution. This test determines how MSS varies within network traffic captures. The MSS varies depending on the type of hosts and their activities in a network. Hosts increase the MSS to maximize their throughput and decrease it to minimize IP fragmentation. This value correlates with the general design and capabilities of a network. The MSS indirectly signals the purpose of a network (e.g., content distribution, cloud services, or office work).
  - Window Size Distribution. This test analyses how different the window sizes of packets are within a network traffic capture. The window size correlates with the bandwidth of a network. Therefore, by estimating the bandwidth, the test determines if the different values of the window sizes match the expected distribution of different window size values.
  - ToS Distribution. This test tracks the ToS TCP header value and detects whether the ToS follows modern protocol specifications within a network traffic capture. The meaning of these headers has changed multiple times in the past, i.e., in RFC 791, 1122, 1349, 1455, 2474, 2780, and 3168 [23]. In some specialized environments, the ToS is used to determine the priority of packets. In most environments, the ToS is ignored.

5.2 Reliability Test Metrics

The reliability tests use diverse metrics to characterize network traffic and identify potential sources of defects. The results of the metrics are available both to an analyst to expose potential problems, and to an attack script to better replicate background traffic properties in synthetic packets.

All metrics can be applied once to a network traffic capture as a whole or many times at different time intervals of the capture. Metrics applied to a whole capture yield *a single scalar value.* A capture may also be split according to a *time interval* to generate many consecutive, non-overlapping, *time windows.* The metrics can be applied to all *time windows* to obtain *a vector of scalar values*, one



(a) Entropies of source IPs.

(b) Entropies of destination IPs.

Fig. 11. Measurements of IP entropies using 100 time windows on the 2018/04/01 of the MAWI dataset.

for each time window. With such a vector, we can plot a sequence of values to see the behavior of a metric across time (windows). We refer to the vector of scalars of a metric applied to consecutive time windows as the *metric distribution*. To simplify the description of the metrics, the term *time window* should be understood also as a reference to a traffic capture as a whole (i.e., a time window that spans all traffic). ID2T creates a file containing all metrics, as they apply to a capture as a whole or at different time intervals. If requested, then ID2T also creates plots of the metric distributions.

In what follows, we present six metrics and examples of how we use these to detect irregular network characteristics.

(Shannon) Entropy. The entropy metric is used to characterize the uncertainty of a network feature within a time window. Entropy $H(X)$ is defined as

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

where $X = \{x_1, x_2, \ldots, x_n\}$, x_i are the values of any one feature, and $P(X)$ is the probability mass function of X (i.e., normalized counts of values). The lowest possible entropy is zero and the largest is $\log_2 n$. A low entropy indicates that the values of X repeat often. Conversely, a high entropy signals that the values of X do not repeat much.

The entropies of IP addresses provide useful information that allows us to identify noteworthy events that occur in network traffic. Figure 11 shows the entropies of the source and destination IP addresses of traffic in a day (2018/04/01) of the MAWI dataset (see Section 3.1 for a description of the MAWI dataset). Entropies are calculated by dividing the traffic in 100 time windows and calculating the entropies of the IPs within each time window.

Normalized (Shannon) Entropy. This metric corresponds to the entropy normalized in the range $[0, 1]$. The normalized entropy $H_n(X)$ is defined as

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

and is equivalent to dividing $H(X)$ by the its maximum value $\log_2 n$, where $X = \{x_1, x_2, \ldots, x_n\}$. This metric enables us to directly compare the values of different datasets. We can use the normalized entropy to compare the features of different datasets.

The properties of the static datasets presented in the related work section (see Section 3.1) cannot be directly compared. This is because the datasets span different amounts of time (e.g., a day in MAWI spans 15 min while a day in DARPA spans 24 h) and contain different packet quantities.

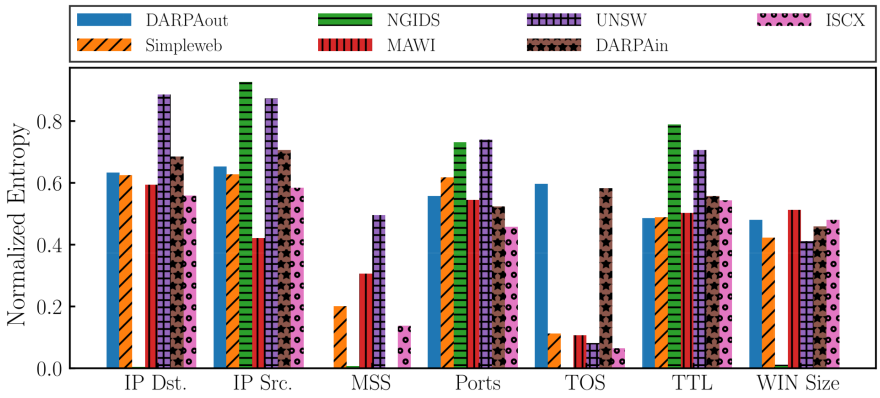


Fig. 12. Comparing the normalized entropies of different datasets.

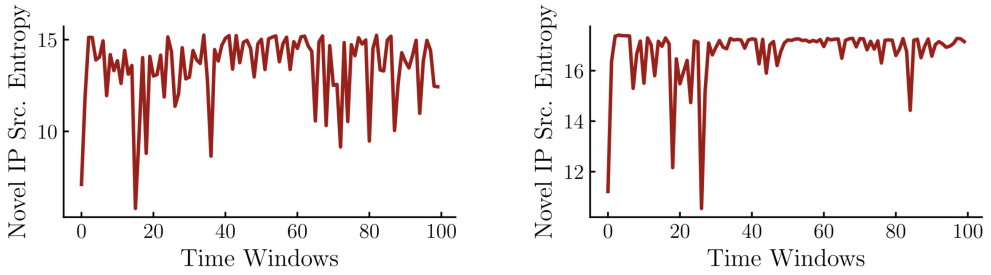
A comparison is meaningful, however, if we use normalized entropies: All properties are characterized with entropies, which we can compare when normalized. The histograms in Figure 12 compare seven properties of seven datasets using the normalized entropy metric.

The y-axis of Figure 12 corresponds to the average normalized entropy of time windows of 14 min of a dataset. We choose time windows of 14 min as all compared datasets span at least 14 min. *DARPAin* refers to one week of the internal traffic of the DARPA dataset. *DARPAout* refers to one week of the external traffic.

Some conclusions can be drawn from the histograms. The DARPA dataset is known to have deficiencies in how traffic uses the MSS, ToS and TTL fields [31]. DARPA has almost zero entropy in all these fields. The fields also differ from other realistic datasets such as MAWI or Simpleweb. The NGIDS dataset shows that the normalized entropy of destination addresses is close to zero, signaling that the same IP addresses are almost always targeted. There are also no variations of the MSS, ToS, and TTL fields. The normalized entropy of the ports shows that the NGIDS dataset has traffic that almost always goes to different IP addresses, something that is not typical of real users. The histograms also shows (from the high source and destination IP address normalized entropies) how the UNSW dataset captures traffic of hosts that are seen few times only.

Novelty Count. This metric counts, for a given random variable, how often new values have been observed during a time window that have never been observed in previous time windows. When the metric is applied to the first time window (or a time window that spans an entire capture), the metric effectively counts each observed value. For example, when applied to a network capture, if we compute the novelty count of destination IP addresses in the first time window, then we count how many packets are associated with each destination IP address. If the metric is applied to time windows other than the first one, then we count how many packets are associated with destination IP addresses that have not been observed in the previous time windows. The novelty count metric is able to inform us about the type and size of a network: In small home networks, the novelty counts of many variables (i.e., source IP addresses or TTL values) are frequently low. The contrary is true for backbone networks, where the novelty count of many variables is typically high.

Novelty Count Entropy. This metric computes the entropy of the novelty counts of a random variable within a time window. For example, if in a time window nine packets are equally distributed among three never-before-seen source IP addresses (the random variable), the novelty count entropy of the source IP at the given time window would only consider those three source IP addresses. Because the packets are equally distributed, the entropy would be maximal (i.e., $\log_2(3)$).



(a) Entropies of novel source IPs.

(b) Entropies of novel destination IPs.

Fig. 13. Measurements of novelty distribution entropies using 100 time windows on the 2018/04/01 of the MAWI dataset.

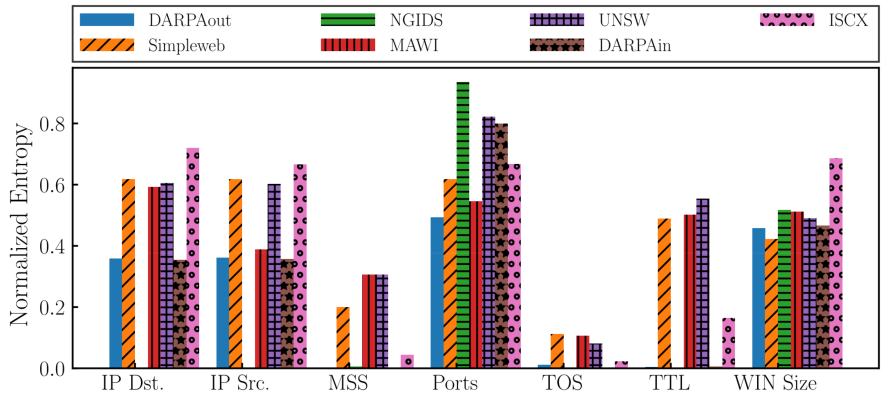


Fig. 14. Comparing the normalized novelty distribution of different datasets.

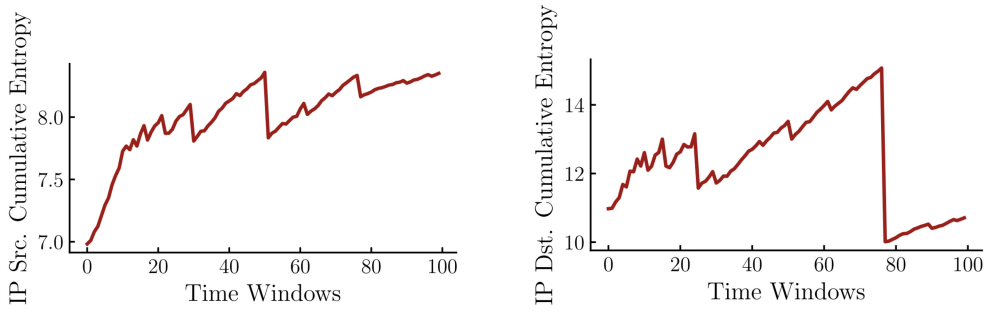
Entropy is typically used to estimate the uncertainty of a random variable. However, entropy is also a good tool to characterize the shape of a distribution.

Figure 13 shows the distribution of novelty IP addresses in a day of the MAWI dataset. From Figure 13(b), we can appreciate that during time window 27 (spanning 9 s), given all newly seen IPs, only few new IPs receive the majority of the traffic. This behavior may signal traffic shifts, routing reconfigurations, the start of new alpha flows [27], network errors or other similar traffic behaviors.

Normalized Novelty Count Entropy. This metric normalizes the novelty count metric in the range of $[0, 1]$ This metric enables us to compare directly the novelty count entropy of different network traffic captures. An example of how this metric can be used in practice follows.

The histograms in Figure 14 compare the normalized novelty distribution of seven fields of seven different datasets. Some known as a well a new insights can be deducted from the figure. Once again, we can conclude from the low novelty values that the DARPA dataset has unusual MSS, ToS, and TTL values. The NGIDS dataset sees all the hosts in the dataset in few time windows, as shown from the zero normalized entropy values of its source and destination IP addresses.

Cumulative Entropy. This metric measures how the entropy of a random variable accumulates over time windows. The cumulative entropy of a random variable in the first time window is the same as the variable's entropy in that same time window. The cumulative entropy of a random variable in other time windows corresponds to the sum of the variable's entropies in the given



(a) Cumulative entropies of src. IPs.

(b) Cumulative entropies of dst. IPs.

Fig. 15. Measurements of cumulative entropies using 100 time windows on the 2018/04/01 of the MAWI dataset.

time window plus all previous time windows. In the following, we show a sample scenario where this metric gives insights into the behavior of network traffic.

Consider the cumulative entropy shown in Figure 15. The figure highlights specific points where entropy drastically shifts; signaling swift behavioral changes in relation to past observations. In time window 78 of Figure 15(b), it is clear that the distribution of destination IP addresses drastically changes. This behavior, although not common, is expected to occur within backbone networks. Had we observed this phenomenon in a home network instead, we would conclude that the unexpected change of traffic probably comes from a deficiency in the dataset.

6 SYNTHETIC ID2T ATTACKS

The main goals of ID2T is to inject synthetic attacks. ID2T comes with attack scripts that generate the packets to simulate diverse attacks. All attack scripts leverage the properties of an input PCAP to disguise the synthetic nature of the packets they create. All attack scripts are programmed to be flexible and, thus, require different user-supplied parameters to craft network packets. If users do not specify all parameters, then the attack scripts select default values for the missing ones. Default values try to match the characteristics of the input. For example, if a victim's IP address parameter is not specified for an attack script that targets a website, the script must find a suitable default. In such a scenario, the script finds a random IP address within the input PCAP that receives traffic on port 80 to use as default.

Many tasks performed by attack scripts, such as finding suitable default parameters, are abstracted away into libraries. These libraries implement functionalities that attack scripts commonly use to ease the creation of new attacks that meet our requirements (see Section 2.2). ID2T has a modular design that enables the easy addition of new attack scripts. An attack script needs to follow some basic conventions[10] to be seen by the *Attack Controller* of ID2T (refer to ID2T's architecture in Section 4.1). The *Attack Controller* gives attack scripts the ability to interact with all other ID2T modules.

We classify the 12 injectable attacks currently provided by ID2T into four classes. Figure 16 shows these injectable attacks and the classes. The four classes are used to refer to the advantages and limitations of groups of attacks (e.g., all implemented exploit attacks). Our classification does not imply that new attacks must strictly belong to one of these classes. Users may use the existing

[10] All conventions, along with the libraries and API, are detailed in the documentation of the tool in https://github.com/tklab-tud/ID2T.

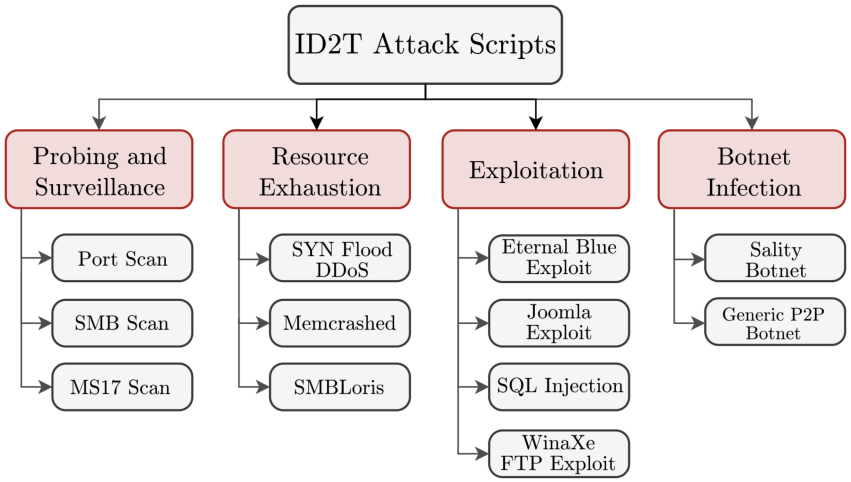


Fig. 16. Classification of the synthetic attacks ID2T can inject. ID2T can inject 12 different attacks. We classify the attacks into four different classes.

libraries of ID2T to create any type of attack, present or not. A user can also extend the libraries to add new functionality onto ID2T. Contrary to creating attack scripts, which do not require direct modifications to ID2T, modifying the libraries implies modifying the source code of ID2T. In what follows, we present the four classes of attacks along with their individual attacks, the potential artifacts that may affect the classes (according to our classification in Section 3.3), and how we avoid artifacts.

6.1 Probe and Surveillance

The attacks crafted by the attack scripts of this class, more than being explicit attacks, are probing or surveillance techniques typically used to gather information before conducting an attack. These attacks use different techniques to disclose information about the provided services or exposed vulnerabilities of network hosts. ID2T includes three different types of network probes.

6.1.1 Potential Artifacts of Probes and Surveillance Attacks. The synthetic attacks crafted by probe and surveillance attack scripts may potentially contain several artifacts. In the *invalid data* category, from a *network specific context*, these attacks could incorrectly generate an amount of packets that disregard the physical limitations (e.g., bandwidth) of the input PCAP. Determining the physical limitations of PCAP files is challenging as the PCAP file format does not explicitly carry such information. The probing and surveillance attacks that we develop estimate the bandwidth and network routing capabilities (i.e., latency and speed) to avoid creating more packets than the network can theoretically carry. Our estimates are based on the heuristic that network bandwidth is a multiple of 10 Mb/s, and that the average packet interarrival time directly correlates with routing capabilities. Whenever our probing and surveillance attacks inject packets, they take into account the estimated physical limitations of the network and the number of packets already present: For a given time window, the attacks only inject a number of packets that does not exceed the difference between the packets present in the time window and our estimated maximum number of packets that the time window may contain.

From a *network agnostic context*, the injection of port scans and similar attacks can easily create artifacts that fall in the *predictable pattern* category. Our probing attacks consider the distribution

of interarrival times and the current background traffic to estimate where and how many packets may be injected. This actively avoids injecting packets that fall within predictable time slots.

6.1.2 The Synthetic Probe and Surveillance Attack Scripts.

Port Scan Attack Script. This attack script emulates port scans that uncover open services in network hosts. The attack script implements a variation of this attacks that executes a vertical TCP SYN port scan: One IP address is scanned fully before moving to the next IP. The attack script disguises the synthetic nature of the packets it creates by imitating the behavior of the popular Nmap[11] port scanner. The disguise process uses an adaptive packet rate generation strategy that targets the most common 1,000 ports (by default).

SMB Scan Script. This attack script generates attacks that search a network for hosts that offer SMB or Samba shared resources. The simulated attack attempts to establish a TCP connection with port 445 of a victim to list available shared resources. This attack script simulates the network communication that takes place when hosts respond, both, positively or negatively to SMB queries. By specifying parameters, a user can control which resources are returned as available on successful queries.

MS17 Scan Script. This is a specialized attack script that simulates attacks that probe hosts running the Windows operating system to determine if they have the *MS17–010* patch.[12] This patch fixes several vulnerabilities that leave a host open to remote exploitation. The attack script disguises the attacks it generates by imitating the behavior of the open source Metasploit framework.[13]

6.1.3 Probe and Surveillance Limitations. When a host answers back to a probing attack, the response creates many packets that may be affected by the background network traffic, attack itself or the network configuration. The network packets generated by ID2T assume that the responses follow the standard TCP procedure (using SYN and ACK control packets) without packet drops or retransmission errors.

6.2 Resource Exhaustion Attack Scripts

These attack scripts aim at conducting attacks that consume the resources of a host to deny legitimate users from using one or more services of the host. ID2T has with three resource exhaustion attack scripts.

6.2.1 Potential Artifacts of Resource Exhaustion Attack Scripts. The injection of synthetic exhaustion attacks relies on crafting large amounts of packets. This crafting process is prone to leave artifacts behind. From a *network specific context*, these attacks may introduce *invalid data* and *artificial anomalies* if care is not taken. These attacks avoid creating *invalid data* artifacts using the same technique that probes and surveillance attacks use: We design the attacks to first estimate the bandwidth and routing capabilities of an input PCAP file. Afterwards, the attack scripts create a number of packets that do not exceed the estimated physical limitations of the input (see also Section 6.1.1).

Synthetic resource exhaustion attacks may incorrectly inject *artificial anomalies* if the attacks craft packets that disregard the network specific characteristics of the input PCAP file. Our resource exhaustion attacks recognize and deal with anomalies that relate to IP addresses and to

[11] https://nmap.org/.
[12] https://docs.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010.
[13] https://www.metasploit.com/.

packet frames.[14] Regarding the IP addresses, we implement our attacks to take into account the classes (i.e., A, B C, D, or E) and types (i.e., public or private) of the IP addresses present in the input. The IP of injected packets mimic these two IP characteristics. Regarding the packet frames, our attacks consider all time-related metrics of a packet (i.e., arrival time, time delta from previous packet, and time of capture), packet length, payload entropy, and protocol parameters (e.g., TCP flags and ports) to create packets that blend with the background traffic (when appropriate). For all considered frame properties, we fit a Gaussian probability distribution to the property and sample from the distribution to obtain values that match the background traffic.

From a *network agnostic context*, without adequate prevention mechanisms, the packets these attacks inject can introduce artifacts of the *predictable pattern* class. The resource exhaustion attacks of ID2T set the TCP/IP protocol properties of a packet (i.e., TTL, window size, MSS, DSCP, and IP Flags) by sampling the distribution of existing properties in the background traffic. These attacks also mimic packet congestion and packet drops.

6.2.2 The Synthetic Resource Exhaustion Attack Scripts. SYN Flood DDoS Attack Script. This attack script creates attacks that imitates multiple malicious network clients trying to consume the network capabilities of a host. In the generated attacks, malicious clients try to establish as many TCP connections as possible. In the attack generation process, the attack script creates the packets that would originate from attackers as well as from the expected responses of victims. The attack script uses the network activities of a victim to tune its behavior. The tuning process has two steps: First, the script determines which ports are open. Second, the script calculates a packet rate that is expected to overwhelm the incoming traffic of a victim. This attack script disguises the attacks it generates by imitating the characteristics of DDoSs generated with Metasploit.

Memcrashed Attack Script. This attack script creates attacks that use an amplification DoS technique that exploits a vulnerability in Memcached[15] servers to send unsolicited traffic to a victim. In the Memcached attack, an attacker sends forged UDP requests to a vulnerable Memcached server specifying the victim's IP address as the source address. The victim then receives unsolicited packets in magnitudes larger than the number of packets sent by an attacker. The attack script implementing this attack only simulates the traffic generated by the attacker. The traffic that the victim would receive is omitted. A SYN Flood DDoS attack can instead be used to simulate the traffic received by the victim.

SMBLoris Attack Script. This attack script creates synthetic DoS attacks that exploit a vulnerability[16] in the SMB protocol to exhaust the resources of a system. With the SMBLoris exploit, an attacker forces large memory allocation on a system, which renders the system inoperable.

6.2.3 Resource Exhaustion Limitations. Resource exhaustion attacks change network flow patterns. The most prominent changes include, but are not only limited to, increased response times, network congestion, increased network latency and high packet-loss rates. The attack scripts of ID2T do not alter the background traffic. Instead, attack scripts add additional traffic to existing traffic. ID2T therefore cannot replicate the side-effects of resource exhaustion attacks. Most NIDSs, however, rely on detecting attacks from suspicious traffic rather than from the observable side-effects of unsuspecting traffic.

[14]The PCAP file format stores information that relates to a network packet in a data structure known as a *frame*. A frame contains packet information that goes from the Data Frame layer up to the Application layer of the OSI reference model.
[15]Common vulnerability and exposure entry CVE-2018-1000115.
[16]A common vulnerability and exposure entry is not available.

6.3 Exploitation Attack Scripts

These are attacks that target specific defects and vulnerabilities in software with the goal of executing unwanted code in a target system. ID2T provides five different attack scripts of this exploitation class. The attack scripts focus on replicating the real payload contents and characteristics of an exploit. In this type of attack, the contents of the payload are as important as the properties of the TCP/IP headers as most NIDSs detect these attacks through deep packet inspection.

6.3.1 Potential Artifacts of Exploitation Attack Scripts. To create synthetic exploitation attacks, we supply the exploitation attack scripts of ID2T with the packets generated by real exploitation attempts. The scripts use the packets as templates and modify them to match the network properties of the input PCAP and the user parameters. Modifying existing packets to make them look as if originating from a different network may leave diverse artifacts behind.

From a *network agnostic context*, we take care that the attack scripts do not alter packets such as to create artifacts of the *invalid data* class. When we modify real packets, we ensure that the resulting packets conform to the underlying protocols and network specification of the input. The exploitation attack scripts of ID2T, when needed, recalculate the TCP, IP and Ethernet packet checksums. For each packet of a template, the attack scripts also modify and set adequate TCP/IP flags; and modify the classes and types of IP addresses to be consistent.

The process of generating synthetic exploits from packet templates can create artifacts of the *predictable patterns* class. We take two measures to avoid these types of artifacts. First, the exploitation attack scripts of ID2T alter the order of the established connections in the template (without affecting the order of the packets) and change the interarrival times between the packets. Second, the scripts employ different examples of successful and unsuccessful exploitation attempts, which are used depending on the conditions of the background traffic. These two measures ensure that different injections of the same attack (in the same or different PCAP) do not produce the same network trace.

6.3.2 The Synthetic Exploitation Attack Scripts.

Eternal Blue Exploit Attack Script. This attack script creates attacks that exploit a buffer overflow vulnerability in the SMB protocol implementation of the Windows operating system.[17] The imitated attack consists in crafting an especially designed SMB message, which an attacker can use to take control of a host. This attack script of replicates the exploit as if launched with the popular Metasploit framework. The created attack relies in the creation of many TCP connections, all of which the script replicates and adapts according to user-supplied parameters.

Joomla Exploit Attack Script. This attack script forges attacks that execute arbitrary code in servers hosting the Joomla[18] content management system. The attack being imitated consists in exploiting a vulnerability[19] that enables remote attackers to create user accounts with full privileges. This attack script uses packets created by the Metasploit framework as a template. The attack script manipulates the TCP/IP and HTTP headers of the template packets to replicate properties of background traffic and adjust to user-supplied parameters.

SQL Injection Attack Script. This attack script forges the packets that an SQL injection attack would create against a vulnerable[20] ATutor[21] learning management system. The replicated attack

[17] Common vulnerability and exposure entry CVE-2017-0144.
[18] https://www.joomla.org/.
[19] Common vulnerability and exposure entry CVE-2016-8870.
[20] Common vulnerability and exposure entry CVE-2016-2555.
[21] https://atutor.github.io/.

creates special SQL commands that are sent over the network to a system hosting the ATutor platform. These sent commands bypass the authentication of the system and give administrator privileges to an attacker. This attack script replicates the behavior of sending malicious SQL commands over a network. To disguise the synthetic nature of the created attacks, the attack script uses the packets created by Metasploit when it is used to conduct a real SQL injection attack against ATutor. The attack script uses the parameters supplied by the user and the input background traffic to blend and disguise, respectively, the synthetically generated packets with the background traffic.

WinaXe FTP Exploit Attack Script. This attack script creates the packets of an attack that target the WinaXe FTP client. The attack script imitates an exploit that gives full unauthorized control of the client's system to a third party. The exploit uses a buffer overflow vulnerability[22] to corrupt the memory of a WinaXe FTP client and give control to whoever controls the malicious FTP server. The attack script gives users the possibility of specifying the payload that the attack contains.

6.3.3 Exploitation Limitations. Exploitation attacks are characterized by network traffic that transports malicious code in the payload of packets. ID2T replicates this type of attacks accurately by using the packets generated by real-world tools (e.g., Metasploit) as templates. However, ID2T replicates a chosen behavior of an attack instance. For example, in an Eternal Blue exploit, the established TCP connections vary depending on whether the attack is successful or not. A user needs to specify, as a parameter, whether the attack is successful or not. Based on this parameter, ID2T chooses which connections to emulate. Hence, even when the number of connections is fixed, ID2T reorders, delays or speeds up the connections. This way, no two exploit attacks leave the exact same packet sequence. This way, no two exploit attacks leave the exact same packet sequence.

6.4 Botnet Infection Attack Scripts

These attack scripts create synthetic traffic related to botnet infections. A botnet is a collection of compromised systems that respond to the commands of an unauthorized actor. Botnets are mostly used to conduct malicious activities that require the coordination of many systems. Typical examples of these malicious activities range from email spam and advertisement fraud to DDoS attacks. ID2T includes attack scripts that inject traces of a real or hypothetical botnet.

6.4.1 Potential Artifacts of Botnet Infection Attack Scripts. Botnet infection attacks have characteristics and patterns similar to resource exhaustion and exploitation attacks. Consequently, generating synthetic botnet infection attacks may create the same artifacts as those potentially generated by resource exhaustion and exploitation attack scripts. These attack scripts avoid artifacts of the *invalid data* (from the *network specific context*), *artificial anomalies*, and *artificial patterns* classes with the same techniques used by resource exhaustion attack scripts (see Section 6.2.1 for more details). In short, botnet infection attack scripts borrow the mechanisms to generate large amounts of packets from the resource exhaustion attack scripts. The scripts also adopt a template-based approach to inject botnet traffic following the mechanisms of exploitation attack scripts.

Botnet communications rely on establishing many network connections, usually using UDP as the underlying protocol. To avoid generating artifacts of the *unrealistic cleanness* type, these attack scripts simulate communication problems: packet loss, duplication, and retransmission. The scripts estimate the bandwidth and network load (see Section 6.1.1 for details) to determine when packets are retransmitted, duplicated, dropped, or lost.

[22] A common vulnerability and exposure entry is not available.

6.4.2 The Synthetic Botnet Infection Attack Scripts.

Sality Botnet Infection Attack Script. This attack script creates packets as if originating from the Sality malware. The real malware infects executable files in the Windows operating system with different purposes. One of its main purposes is to give third parties unauthorized control of a systems. Systems infected with Sality communicate over a Peer to Peer (P2P) network to enable the coordination and execution of remote commands. This attack script creates the network packets that the Sality variant *Win32-Sality.AM* generates when it communicates with peers. The payloads of the packets contain a malicious Dynamic Link Library (DLL) that can be found in Sality infections in the wild. The network traffic patterns and infected DLL come from VirusTotal.[23] This attack script takes care of modifying the network communication traces of Sality to match that of the background input data.

Generic P2P Botnet Infection Attack Script. This attack script generates network packets as if originating from a user-defined P2P botnet. This script simulates botnets that have multiple properties and characteristics. A user specifies the botnet communication patterns in a comma-separated file. The communication patterns specify the type and timing of the messages bots send to each other. The script transforms the communication patterns into network traffic that matches the user-supplied background traffic.

6.4.3 Botnet Infection Limitations. The Sality botnet variant replicated by ID2T creates network traffic that targets a set of external peers. The IP addresses of these peers are fixed and do not vary between injections. However, NIDSs identify Sality based on the network traffic footprint, which ID2T replicates, instead of the IP addresses used.

7 EXEMPLARY EVALUATION BY USE CASES

Synthetic traffic can be useful in many different contexts. One such context, and the one on which this section focuses, is that of reproducing scientific results. We believe that the field of NIDSs is currently experiencing a reproducibility crisis. The crisis is mostly due to results being highly coupled to private or difficult to replicate network datasets. Many in the research community believe that synthetic or partially synthetic datasets are the best means to bring the field forward toward reproducibility [40, 44]. Therefore, we show in this section how ID2T can be used to reproduce the detection capabilities of NIDSs.

Two use cases are the subject of this section. The first use case employs ID2T to create datasets that can be used to evaluate the results of an anomaly-based IDS. The second use case validates the ability of ID2T to inject attacks with well known footprints that Signature-based Network Intrusion Detection Systemss (SNIDSs) can successfully detect. In the two use cases, we use ID2T (to generate synthetic attacks) and publicly available PCAP files (as background traffic).

7.1 Reproducing Anomaly-based Evaluation Results

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 [15] 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

[23] https://www.virustotal.com/.

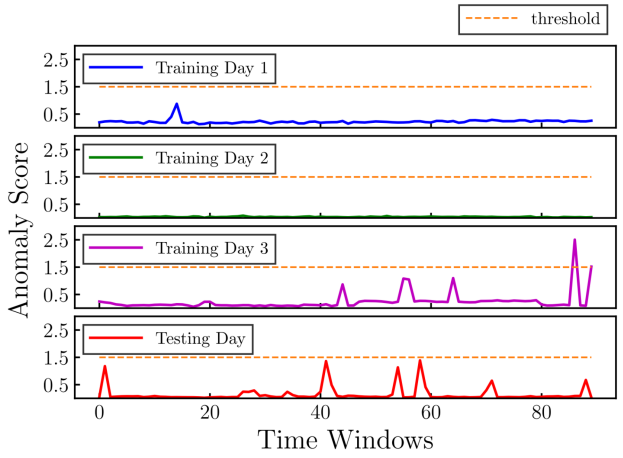


Fig. 17. Anomaly scores given to 4 different days of the MAWILab dataset by an RNN. The first 3 days are used to train the model. The fourth day is used to test the trained model.

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.

In this section, we demonstrate the ability to replicate the anomaly detection capabilities of shallow autoencoders, known as Replicator Neural Networks (RNNs) [18], with easy to reproduce datasets. 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 a previously trained normality model. It is assumed that traffic that does not conform to a normality model will yield higher anomaly scores than normal traffic. When the anomaly score passes a threshold determined by the model, an anomaly is raised (signaling a potential attack).

The experimental setup for this use case is as follows. We use 4 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 leave distinguishable fingerprints that an anomaly detector should identify. Each day of the MAWI dataset consists of 15 min of traffic. We process the 4 days using a three step process. First, we split each day into non-overlapping time windows of 10 s intervals. Second, we convert all traffic within a time window into network flows and sanitize them by removing flows marked as anomalous in 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 Reference [18] for more details). The set of four entropies belonging to the time windows of the first 3 days are used to train an RNN. The last day is used to evaluate the performance of an RNN.

Figure 17 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 min $\times$ 60 s = 90 time windows). The y-axis shows the anomaly score of the entropies calculated at each time window. The RNN uses the first 3 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.

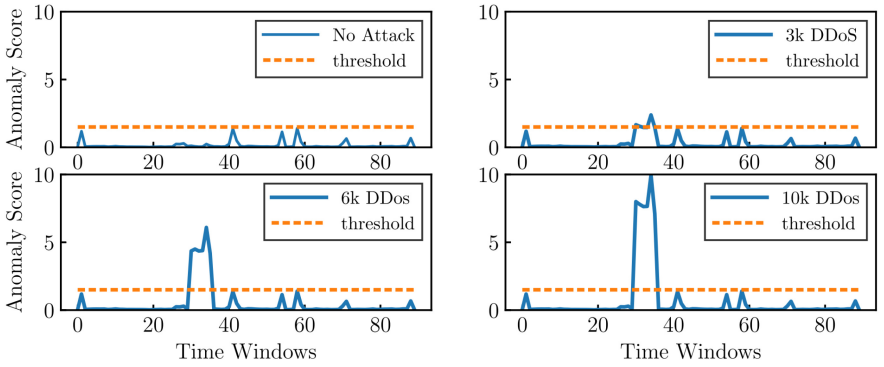


Fig. 18. Detecting different DDoS attacks injected with ID2T using anomaly detection.

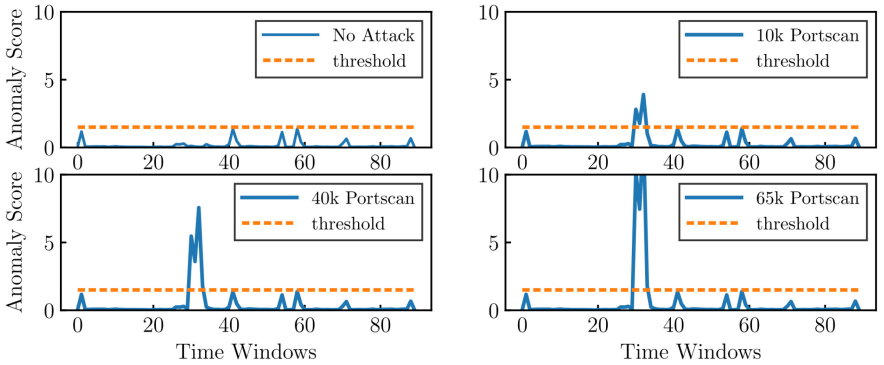


Fig. 19. Anomaly scores of different port scans of varying intensities.

7.1.1 Detecting Injected DDoS Attacks. We proceed to inject DDoS attacks of different intensities into the testing day and use a RNN to detect the attacks. Figure 18 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%, respectively.

From Figure 18, 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 2 years ago in relation to this work).

7.1.2 Detecting Injected Port Scans. Using ID2T, we test the capability of RNNs to detect port scans in large networks. Figure 19 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

Table 3. Testing Two Different SNIDSs Against Four Different Attacks Injected by ID2T

	Bro	Suricata
Port Scan	✓[a]	✗
EternalBlue Exploit	✓	✓[b]
WinaXe FTP Exploit	✗	✗
Sality Botnet	✓	✓[c]

A check mark (✓) indicates that the attack was detected; otherwise, a cross (✗) is used.
[a]Identified as: Scan::Port_Scan 188.165.214.141 scanned at least 15 unique ports of host 188.165.214.253 in 0 m, 4 s.
[b]Exploit identified as: ETERNALBLUE MS17-010 Echo Response and ET-PRO TROJAN (possible Metasploit payload).
[c]Sality identified as: ETPRO TROJAN Win32/Sality.AM and ET MALWARE Sality Virus User Agent Detected (KUKU)

30 s (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.

7.2 Validating Signature-based Configurations

In this exemplary evaluation by a use case, we use ID2T to create PCAPs with injected attacks and determine if the SNIDSs Bro [36] and Suricata[24] detect the injections. We choose attacks that leave well-known signatures that should be identified in a correctly configured NIDS. Each attack is injected into the same background PCAP (without mixing the attacks). The background PCAP consists of office network traffic collected during one minute. It contains 30 MiB of data and 55,000 packets.

The experimental setup of this use case is as follows. Using ID2T, we inject four different attacks into a PCAP file of one minute of (university) office network traffic. The network traffic was close to 30 MiB of data and had 55,000 packets. Because we inject small specialized attacks that generate no more than 500 packets, our attacks comprise no more than one percent of the total traffic.

Table 3 shows the detection results of Bro and Suricata when the signatures of the recognized ETOpen Ruleset[25] are in use. We mark successful detections with an arrow (✓). A cross (✗) indicates that the corresponding malicious activity was not detected. We emphasize that we are not testing the detection accuracy of NIDSs, but rather the ability of ID2T to create synthetic attacks that do not trigger false positives.

The detection results of the tested NIDSs shown in Table 3 are as follows. The synthetic port scan was detected by Bro given that we included a rule that finds port scans. Suricata, however, was not configured with such a rule (or signatures) and did not find the port scan. Both Bro and Suricata detected our synthetic *EternalBlue* exploit with the signatures that they both have by default. Additionally, Suricata detected that the payload being used originates from Metasploit. This is a positive result as ID2T mimics what Metasploit does to create the "EternalBlue" exploit.

[24]https://www.openinfosecfoundation.org.
[25]https://rules.emergingthreats.net.

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 expected with no configured signatures, neither Bro nor Suricate could detect the malicious payload. Finally, both Bro and Suricata correctly identify a synthetic Sality bot infection. ID2T mimics how Sality sends HTTP messages with the user agent field set to "KUKU." Suricata was able to identify Sality because of this fact.

7.3 Discussion of the Use Cases

The two exemplary evaluations by use cases showcase the ability of ID2T to reproduce and replicate scientific results, and to test the detection capabilities of NIDSs. In the first exemplary evaluation, we reproduce the network intrusion detection capabilities of RNNs using network traffic captures of the MAWI dataset that were captured almost two years apart. This means that although the traffic comes from the same dataset, one can argue that because of how backbone networks have evolved the last two years [3], the traffic captures are significantly different. With ID2T, we show how RNNs can be easily re-evaluated using an easy to replicate dataset. The first exemplary evaluation also demonstrated that we do not need someone to publish labeled datasets with ground truth (which are expensive to create). Instead, we just need arbitrary network traffic captures (which are easy to obtain).

We argue that in the last two years the characteristics of network traffic in backbone networks have changed considerably [3]. ID2T is able to help conduct or reproduce evaluations without having to create and publish a 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 exemplary evaluation, we use ID2T to test two popular SNIDS. ID2T 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 injected attacks that successfully replicated their signatures. ID2T can be used, therefore, to verify the correct operation of already deployed systems.

8 CONCLUSION AND FUTURE WORK

We developed ID2T with the intention to address the long standing issue of not having reliable[26] 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 to create custom reproducible datasets that can fulfill the five functional requirements we identified for datasets to be generally useful in the evaluation of NIDSs (see Section 2.1). The users of ID2T can easily create datasets that have full payloads (i.e., *the payload availability requirement*) given that the input traffic fed to ID2T belongs to the user. A user only needs to share attack scripts. ID2T automatically creates labeled datasets (i.e., *the labeled attacks requirement*) with ground truth (i.e., *the ground truth requirement*) if the user supplies ID2T with benign traffic only. The attacks of ID2T can be injected into past, present and future traffic

[26] We use the term *reliable dataset* to refer to a dataset that fulfills all requirements identified in Section 2.1.

captures (i.e., *the renewable requirement*). Finally, ID2T can parse, analyze and inject traffic captures with attacks regardless of the provenance of the capture (i.e., *the flexible requirement*)

ID2T actively avoids creating artifacts in the attacks it injects. The injected attacks replicate the characteristics of the background traffic to erase any trace that points toward 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 exemplary evaluation section (Section 7), 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 three 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 retransmissions 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. Finally, we intend to investigate how ID2T can automatically support the notions of spatial correlation (collecting traffic at different anchor points of a network) and collaborative intrusion detection [51]. To achieve this automatic support, the toolkit should provide users with adjustable options to support multiple (and different) spacial anchors.

REFERENCES

[1] Sebastian Abt and Harald Baier. 2013. Are we missing labels? A study of the availability of ground-truth in network security research. In *Proceedings of the Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS'14)*.

[2] United States Military Academy. 2009. CDX 2009 Network. Retrieved from https://www.westpoint.edu/centers-and-research/cyber-research-center/data-sets.

[3] Akamai. 2018. The state of the internet / security report. Retrieved from https://www.akamai.com/uk/en/multimedia/documents/case-study/spring-2018-state-of-the-internet-security-report.pdf.

[4] 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.

[5] Steven M. Bellovin. 1992. Packets found on an internet 1 introduction 2 address space oddities. *Comput. Commun.* 23, 3 (1992), 1–8.

[6] Monowar H. Bhuyan, Dhruba K. Bhattacharyya, and Jugal K. Kalita. 2015. Towards generating real-life datasets for network intrusion detection. *Int. J. Netw. Secur.* 17, 6 (2015), 683–701.

[7] Daniela Brauckhoff, Arno Wagner, and May Martin. 2008. FLAME: A flow-level anomaly modeling engine. In *Proceedings of the Conference on Cyber Security (CSET'08)*.

[8] CAIDA. 2017. CAIDA Data—Overview of Datasets, Monitors, and Reports. Retrieved from http://www.caida.org/data/overview/.

[9] National CyberWatch Center. 2017. Mid-Atlantic Collegiate Cyber Defense Competition. Retrieved from https://maccdc.org/.

[10] 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 *Proceedings of the Conference on Communications and Network Security (CNS'15)*. IEEE, 739–740.

[11] 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. Retrieved from http://buildbot.tools.ietf.org/html/rfc6335.

[12] Gideon Creech and Jiankun Hu. 2013. Generation of a new IDS test dataset: Time to Retire the KDD Collection. In *Proceedings of the Wireless Communications and Networking Conference (WCNC'13)*. IEEE, 4487–4492.

[13] 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. MIT Lincoln Lab.

[14] Peter B. Danzig and Sugih Jamin. 1991. tcplib: A library of internetwork traffic characteristics. *Library* 48 (1991), 1–8.

[15] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proceedings of the Conference on Emerging Networking EXperiments and Technologies (CoNEXT'10)*. ACM, 1–12.

[16] Sebastian Garcia. 2011. Stratosphere Research Laboratory. Retrieved from https://www.stratosphereips.org/.

[17] Sebastian Garcia, Martin Grill, Jan Stiborek, and Alejandro Zunino. 2014. An empirical comparison of botnet detection methods. *Comput. Secur.* 45 (2014), 100–123.

[18] Carlos Garcia Cordero, Sascha Hauke, Max Mühlhäuser, and Mathias Fischer. 2016. Analyzing flow-based anomaly intrusion detection using replicator neural networks. In *Proceedings of the 14th Annual Conference on Privacy, Security and Trust (PST'16)*. 317–324. DOI: https://doi.org/10.1109/PST.2016.7906980

[19] Dan Grossman. 2002. *New Terminology and Clarifications for Diffserv*. RFC 3260. Retrieved from http://buildbot.tools.ietf.org/html/rfc3260.

[20] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie. 2017. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *J. Netw. Comput. Appl.* 87 (2017), 185–192.

[21] Santiago Hernández. 2018. Awesome-Cybersecurity-Datasets. Retrieved from https://github.com/shramos/Awesome-Cybersecurity-Datasets.

[22] IMPACT. 2017. Information Marketplace. Retrieved from https://www.impactcybertrust.org.

[23] Kadangode K. Ramakrishnan, Sally Floyd, and D. Black. 2001. *The Addition of Explicit Congestion Notification (ECN'01) to IP*. Technical Report.

[24] KDD Cup 99. 1999. Knowledge Discovery and Data Mining Tools Competition. Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[25] Robert Koch, Mario Golling, and Gabi Dreo Rodosek. 2014. Towards comparability of intrusion detection systems: New data sets. In *Proceedings of the TERENA Networking Conference*. 7.

[26] Anukool Lakhina, Mark Crovella, and Christophe Diot. 2005. Mining anomalies using traffic feature distributions. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'05)*. ACM Press, 217–228.

[27] Imed Lassoued. 2011. *Adaptive Monitoring and Management of Internet Traffic*. PhD Thesis. Université de Nice.

[28] Marc Liberatore and Prashant Shenoy. 2013. Umass trace repository. Retrieved from http://traces.cs.umass.edu.

[29] Thomas Lukaseder. 2017. 2017-SUEE-data-set. Retrieved from https://github.com/vs-uulm/2017-SUEE-data-set.

[30] 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.

[31] Matthew V. Mahoney and Philip K. Chan. 2003. An analysis of the 1999 DARPA/lincoln laboratory evaluation data for network anomaly detection. In *Proceedings of the International Symposium on Recent Advances in Intrusion Detection*. 220–237. DOI: https://doi.org/10.1007/b13476

[32] John McHugh. 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Info. Syst. Secur.* 3, 4 (2000), 262–294. DOI: https://doi.org/10.1145/382912.382923

[33] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proceedings of the Military Communications and Information Systems Conference (MilCIS'15)*. IEEE, 1–6.

[34] Boris Nechaev, Mark Allman, Vern Paxson, and Andrei V. Gurtov. 2010. A preliminary analysis of TCP performance in an enterprise network. *INM/WREN* 10 (2010).

[35] NETRESEC. 2010. Capture files from Mid-Atlantic CCDC. Retrieved from https://www.netresec.com/?page=MACCDC.

[36] Vern Paxson. 1999. Bro: A system for detecting network intruders in real-time. *Comput. Netw.* 31, 23–24 (1999), 2435–2463. DOI: https://doi.org/10.1016/S1389-1286(99)00112-7

[37] Jon Postel et al. 1981. *Internet Protocol*. RFC 791. Retrieved from http://buildbot.tools.ietf.org/html/rfc791.

[38] Nadun Rajasinghe, Jagath Samarabandu, and Xianbin Wang. 2018. INSecS-DCS: A highly customizable network intrusion dataset creation framework. In *Proceedings of the IEEE Canadian Conference on Electrical & Computer Engineering (CCECE'18)*. IEEE, 1–4.

[39] Joyce Reynolds and Jon Postel. 1994. *Assigned Numbers*. Technical Report.

[40] Haakon Ringberg, Matthew Roughan, and Jennifer Rexford. 2008. The need for simulation in evaluating anomaly detectors. *SIGCOMM Comput. Commun. Rev.* 38, 1 (Jan. 2008), 55–59. DOI: https://doi.org/10.1145/1341431.1341443

[41] 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 *Proceedings of the USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET'09)*.

[42] Mike Sconzo. 2015. Samples of Security Related Data. Retrieved from https://www.secrepo.com/.

[43] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* 31, 3 (2012), 357–374.

[44] John Sonchack, Adam J. Aviv, and Jonathan M. Smith. 2013. Bridging the data gap: Data related challenges in evaluating large scale collaborative security systems. In *Proceedings of the 6th Workshop on Cyber Security Experimentation and Test*.

[45] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. 2006. Description of Kyoto University benchmark data. *Academic Center for Computing and Media Studies (ACCMS), Kyoto University*.

[46] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. 2008. Cooperation of intelligent honeypots to detect unknown malicious codes. In *Proceedings of the WOMBAT Workshop on Information Security Threats Data Collection and Sharing (WISTDCS'08)*. IEEE, 31–39.

[47] Anna Sperotto, Ramin Sadre, Frank Van Vliet, and Aiko Pras. 2009. A labeled data set for flow-based intrusion detection. In *Proceedings of the International Workshop on IP Operations and Management*. Springer, 39–50.

[48] SPIRENT. 2002. pcapr: PCAP files repository. Retrieved from https://www.pcapr.net/.

[49] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *Proceedings of the Symposium on Computational Intelligence for Security and Defense Applications (CISDA'09)*. IEEE, 1–6. DOI: https://doi.org/10.1109/CISDA.2009.5356528

[50] Emmanouil Vasilomanolakis, Carlos Garcia Cordero, Nikolay Milanov, and Max Mühlhäuser. 2016. Towards the creation of synthetic, yet realistic, intrusion detection datasets. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'16)*. IEEE, 1209–1214.

[51] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 2015. Taxonomy and survey of collaborative intrusion detection. *Comput. Surveys* 47, 4 (2015), 33.

[52] Emmanouil Vasilomanolakis, Matthias Krügl, Carlos Garcia Cordero, Max Mühlhäuser, and Mathias Fischer. 2015. SkipMon: A locality-aware collaborative intrusion detection system. In *Proceedings of the IEEE 34th International Performance on Computing and Communications Conference (IPCCC'15)*. IEEE, 1–8.

[53] Richard Zuech, Taghi M. Khoshgoftaar, Naeem Seliya, Maryam M. Najafabadi, and Clifford Kemp. 2015. A new intrusion detection benchmarking system. In *Proceedings of the FLAIRS Conference*. 252–256.

Received June 2018; revised March 2020; accepted September 2020