Towards the creation of synthetic, yet realistic, intrusion detection datasets

Emmanouil Vasilomanolakis*, Carlos Garcia Cordero*,
Nikolay Milanov*, Max Mühlhäuser*
* CASED / Telecooperation Lab,
Technische Universität Darmstadt
{vasilomano, garcia, max}@tk.tu-darmstadt.de, {nikolay.milanov}@stud.tu-darmstadt.de

***Abstract*—Intrusion Detection Systems (IDSs) are an important defense tool against the sophisticated and ever-growing network attacks. With this in mind, the research community has been immersed in the field of IDSs over the past years more than before. Still, assessing and comparing performance between different systems and algorithms remains one of the biggest challenges in this research area. IDSs need to be evaluated and compared against high quality datasets; nevertheless, the existing ones have become outdated or lack many essential requirements. We present the Intrusion Detection Dataset Toolkit (ID2T), an approach for creating out-of-the-box labeled datasets that contain user defined attacks. In this paper, we discuss the essential requirements needed to create synthetic, yet realistic, datasets with user defined attacks. We also present typical problems found in synthetic datasets and propose a software architecture for building tools that can cope with the most typical problems. A publicly available prototype, is implemented and evaluated. The evaluation comprises a performance analysis and a quality assessment of the generated datasets. We show that our tool can handle large amounts of network traffic and that it can generate synthetic datasets without the problems or shortcomings we identified in other datasets.**

I. Introduction

The continuous increase of cyber-attacks has made IDSs a mandatory line of defense for protecting critical networks [18]. Research in all areas of intrusion detection has flourished over the years [4]. However, many researchers struggle to find valid and commonly accepted datasets for evaluating their proposals [8]. This difficulty has not been tackled yet. There is no commonly accepted dataset that can be used for the purpose of evaluating intrusion detection algorithms or systems. In many cases, researchers still rely on outdated datasets to compare their results. The DARPA 1999 dataset [9] is such an example. This dataset is not only outdated but it also contains many inherent problems [10]. Moreover, many systems are evaluated with non-publicly available datasets; making the reproducibility of the results impossible.

In this paper, we propose ID2T, a toolkit capable of producing labeled datasets intended for the evaluation of IDSs. This paper builds on top of the preliminary work presented in [5] and provides a deeper analysis of the proposed architecture along with evaluations based on a prototype. ID2T takes as input network packet captures (of arbitrary sizes) and generates a labeled dataset. Our target is to build tools to assist researchers in the process of generating datasets so as to compare different IDSs in a reproducible manner. In addition, ID2T also aims at creating flexible datasets tailored to different scenarios, e.g., corporate or backbone networks. We are publishing the toolkit and its prototype publicly [13] so that the research community can benefit from it. The performance of the prototype is evaluated by showing how it can handle large amounts of network traffic. Furthermore, we examine specific properties of generated datasets to assess whether they contain undesired artifacts or anomalies.

The remainder of this paper is structured as follows. In Section II we propose a number of requirements for building high quality datasets and dataset generation tools. We discuss in Section III the related work in the area of intrusion detection dataset generation. Section IV provides an extensive discussion of our system, giving insights into the architecture and the attack generation process. Section V presents evaluation results for the ID2T prototype in relation to its performance as well as its ability to generate quality datasets. We further discuss the outcome of the evaluation along with existing limitations and further steps in Section VI. Lastly, Section VII concludes this paper and provides insights into our intended future work.

II. Requirements

In this section, we propose functional and non-functional requirements for tools that generate synthetic network datasets as well as for synthetic datasets that aim at becoming useful for the evaluation of IDSs. It is not expected that all tools and datasets comply with these requirements. Furthermore. many requirements can also be partially fulfilled. The more requirements fulfilled the more general the tools and datasets become. We also touch on the meaning and importance of *quality*, a non-functional requirement, in the context of automatically creating synthetic datasets.

A. Functional Requirements

Functional requirements relate to certain practical properties of a dataset generation tool, its output dataset or both.

- Payload Availability: The packets' payload needs to be available. This information is required by intrusion detection algorithms that wish to detect intrusions that are

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

only present in the payload, e.g., exploits, SQL injections, phishing attacks, etc. Note that in many cases the payload is removed from datasets due to privacy reasons.
- Attack Diversity: The dataset or tool must contain a broad range of attacks that span from traditional network attacks, e.g., port-scans, to novel malicious activity, e.g., a sophisticated malware.
- Labeled Data: The generated dataset needs to contain labels identifying malicious traffic.
- Ground Truth: Besides labels, the dataset should be able to guarantee the absence of attacks or anomalies in the labeled as non-malicious data. Usually this can only be achieved via a synthetic dataset generation.

B. Non-Functional Requirements

Non-functional requirements refer to the properties of the generated dataset, the toolkit that influences the quality of the produced dataset, or both.

- Availability and Reproducibility: The generated datasets must be publicly available and, hence, allow the reproducibility of experiments.
- Scalability: The toolkit for generating datasets should be able to handle as input and output network files of arbitrary size.
- Interoperability and Flexibility: The toolkit needs to provide the user with a method, e.g., templates or an API, for creating new attacks or modifying existing ones.
- Quality: For creating synthetic datasets, the attack generation process is required to actively avoid introducing undesired patterns, or *artifacts*, outside of the scope of the desired attacks. We expand on this requirement in what follows.

C. Dataset Quality

Many intrusion detection datasets have had problems with the injection of inadvertent anomalies which are easy to recognize by pattern recognition techniques. We have identified several of these which should be addressed whenever unrelated network packet capture files are merged as one or when a packet capture file is altered. The following list of defects, or artifacts, is non-exhaustive and only reflects the main sources of problems we found in our experiments and related work.

- TTL Value Distribution: Due to the different connectivity characteristics of individual networks, the distribution of TTL values varies. The distribution of TTL values must be replicated to avoid creating easy avenues of detection for learning algorithms.
- Packet Capture Time Record: Depending on many factors, such as the bandwidth and the number of connected hosts, packets captured in a network are recorded with different time characteristics. Bursty or constant packet rates should be imitated as well as the distribution of packet inter-arrival times.
- Packet Checksum: It is not sufficient to only modify desired values in network packet capture files. It is crucial to recompute checksums whenever appropriate.
- IP Address Distribution: Depending on the network, IP addresses are usually concentrated between specific address ranges. Injected or modified packets should use IP addresses from the same regions of concentration.
- Network Link Reliability: Due to the nature of networks, it is common to loose packets due to a saturated link or exhausted resources. It is important to identify these problems and replicate them when appropriate.

We further discuss the aforementioned requirements in the evaluation section of the ID2T prototype (see Section V) and in the discussion section (see Section VI).

III. Related Work

In this section, we discuss the related work by first examining existing datasets and discussing other available tools for creating datasets.

A. Datasets

A number of synthetic and non-synthetic datasets have been published over the years with the purpose of being utilized for the evaluation of intrusion detection algorithms and systems [9], [7], [15], [8]. There are two major problems with the majority of existing datasets. First, many of them have been created over a decade ago and, thus, do not exhibit realistic network traffic nor contain up to date cyber-attacks. Second, as a result of their synthetic creation, many of these datasets include undesired artifacts that can significantly reduce their usability. The DARPA 1999 dataset [9] is an illustrative example of both the aforementioned problems. Being generated more than 15 years ago, it contains network traffic and attacks that are antiquated. Moreover, a lot of criticism has been made with regard to undesired artifacts during the generation of the attacks [12]. For instance, Mahoney and Chan [10] discovered inconsistencies in the Time To Live (TTL) values of malicious and non-malicious traffic.

Other examples of datasets include [1], [6], [17] and [7]. None of these fulfill all the desired requirements (cf. Section II). For instance, the MAWI dataset [7] consists of a very large number of modern network captures but there is neither a ground truth nor packet payloads.

B. Dynamic Creation of Datasets

Beyond single, static datasets, research has been conducted in the area of dynamically generated datasets, e.g., [16], [3]. Shiravi et al. [16] proposed a systematic approach for generating datasets by making use of profiles. Even though this work seems promising, the results are only available on-demand. Moreover, the authors do not distribute their toolkit but rather an output dataset.

The Flow-Level Anomaly Modeling Engine (FLAME) [3] tool is the work that is the closest to ours. It works by taking as an input serial streams of flows and injecting hand-crafted network anomalies. As the name implies, this tool manipulates network flows. While this is useful in many circumstances, it also comes with a number of restrictions. First, datasets generated by FLAME cannot be utilized for evaluating intrusion detection algorithms in an agnostic manner; rather, they are limited to algorithms that make use of network flows. Lastly, FLAME, due to only working with network flows, is limited to the injection of attacks with flow footprints.

IV. ID2T

In this section, we first discuss the architecture of our toolkit and, subsequently, give insights into the attack generation modules of ID2T. In addition, we provide a brief description of the Graphical User Interface (GUI) of our prototype.

A. Architecture

Figure 1 provides an overview of the system's architecture. With respect to the core part of the toolkit, there are four internal modules that react to user input for creating labeled datasets: the *statistics*, the *packet splitter*, the *attack controller* and the *merger*.

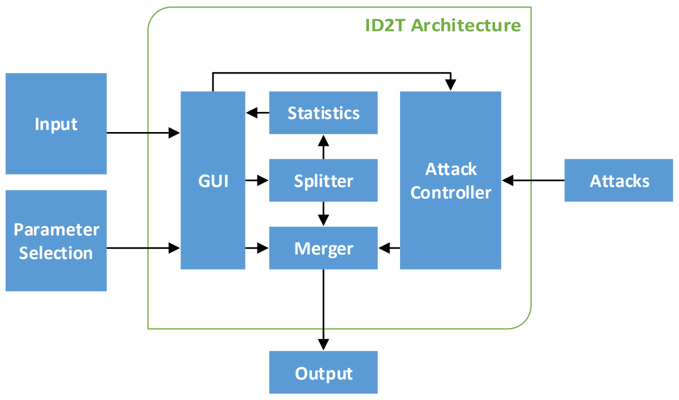


Fig. 1. ID2T Architecture

The *statistics* module is responsible for calculating statistics required by the attack controller module so as to replicate the quantitative and qualitative characteristics of the input. It is also connected with the GUI, providing a statistical overview of the input.

The *packet splitter* module is responsible for processing large network files by splitting them into smaller chunks, thus, allowing an efficient processing of network files. The splitting process is achieved either by utilizing fixed time intervals or fixed packet numbers. This module is envisioned to support different network file formats, e.g., pcap and pnpcap. It currently supports only the pcap file format.

Attacks are modeled utilizing a framework provided by the *attack controller* module. As this is one of the core parts of ID2T, we comprehensively discuss it in the following subsection.

Lastly, the *packet merging* module utilizes the network packets created by the attack controller and merges them with the input file. This module also provides the user with the flexibility of selecting between various generated attacks. After the attacks of interest are specified, the merger combines them with the input. During the merging process, all the packets are chronologically sorted and stored as a single file. Additionally, labels are exported.

B. Attack Generation

ID2T is able to generate attacks by utilizing two different techniques, namely *script-based attack generation* and *pcap modification*. The first technique aims at generating attacks on-the-fly with python scripts. The second one modifies user provided pcap files. This is particularly useful for replicating existing malware or known exploits. We proceed to discuss each technique in more detail.

1) Script-based Attack Generation:

The main idea behind script-based generation is that many cyber-attacks can be modeled as a large number of packets with similar parameters that utilize the same properties. For instance, in the case of a Distributed Denial of Service (DDoS) attack, large number of packets share almost the same parameters; only specific fields change. It is easy to replicate random attackers by selecting random IPs as attackers, all targeting one predefined IP address. A multitude of attacks can be modeled in such a way and are easily implemented in ID2T via scripts.

Implementing new attacks is a straightforward process; the only precondition is to follow the class-template style of ID2T and the toolkit will include the new attack automatically. However, attacks that include several protocols and a large amount of parameters – where these cannot be easily modeled – should, instead, use the pcap modification technique as explained in the following.

2) PCAP Modification:

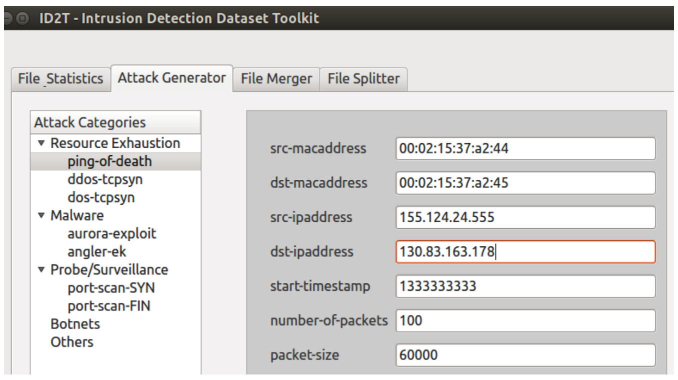
In contrast to the script-based attack generation, pcap modification offers a technique suitable when modeling more complex attacks. For example, this may refer to attacks that make use of multiple protocols, include specific payloads or both. The first requirement for this injection technique is the availability of a pcap file that contains traffic belonging to an attack. The user can either acquire such files from publicly available databases (e.g., from VirusTotal) or create his own manually. When such a pcap file is available, the user can provide replacement parameters (e.g., the preferred malicious IP addresses) that will be used in newly injected attacks. Subsequently, ID2T makes use of certain functions, provided by the Scapy [2] network packet manipulation tool, to adjust parameters with respect to the user input. The outcome of the overall procedure is a new pcap file that includes the malicious traffic with specific fields replaced by the values provided by the user. The merger module is responsible for injecting the newly created file into a user defined input.

C. ID2T Prototype

The prototype for ID2T has been implemented in Python. The GUI, as depicted in Figure 2, is responsible for visualizing all the necessary elements of ID2T.

In the current prototype version, the procedure for generating a dataset is a three step process. First, the user specifies the input pcap file that is used as the basis (normal traffic) of the dataset generation. The statistics module parses the data, generates information with respect to the input and computes various statistics (e.g., the TTL distribution) which are utilized in subsequent steps. Second, the user can decide, through the attack generator, which attacks to inject into the original pcap file. When this is decided and the respective parameters are given, the toolkit can generate a pcap file with the chosen background traffic and the selected attacks. Note that the system is making use of information gathered from the first step to create realistic attacks. For instance, the TTL distribution of the network is taken into account so as to replicate it in the attack packets. The prototype uses two strategies to label data. It can either create a text file indicating where each attack has been inserted (with its time and duration) or by tainting the MAC addresses of all attack packets. MAC address tainting is the process of setting specified bits of the MAC addresses to known values. It is later possible to decide if a packet belongs to an attack by comparing the bits of the MAC addresses to the taint. Finally, ID2T will produce, as a final output, a labeled dataset in the form of a pcap network file.

With regard to the attack generation, the prototype already implements some common network attacks. In more details, resource exhaustion attacks such as the *Ping of Death (PoD)*, a TCP-SYN Denial of Service (DoS) and a TCP-SYN DDoS have been implemented with the script-based attack generation technique. Similarly, a number of port-scan attacks are also implemented. Additionally, the traffic generated by the Angler malware and the Aurora exploit are available.



src-macaddress 00:02:15:37:a2:44
dst-macaddress 00:02:15:37:a2:45
src-ipaddress 155.124.24.555
dst-ipaddress 130.83.163.178
start-timestamp 1333333333
number-of-packets 100
packet-size 60000

Fig. 2. GUI view of the ID2T prototype

V. Evaluation

We evaluate the ID2T prototype in a twofold manner. We begin examining the performance of the toolkit in terms of its ability to handle network files of arbitrary size. Afterwards, we evaluate the quality of the generated datasets by searching for common mistakes and criticisms of other synthetically created datasets, as mentioned in the related work. We further map the requirements stated in Section II to the attributes of the generated datasets.

A. Performance Evaluation

For an intrusion detection dataset toolkit to be practically usable, it is important to efficiently handle large input files. Figure 3, depicts the time required for the statistics module to perform its functions with large network files. For this we utilized real-word traffic files from the MAWI dataset [7]. As one can notice, the toolkit is able to handle large datasets in a reasonable window of time.

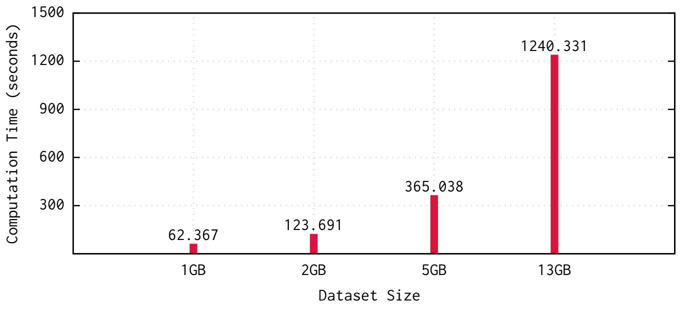


Fig. 3. Performance of the statistics module for different dataset sizes

Likewise, Figure 4 shows the time required for generating three different types of attacks with respect to the number of packets injected. In particular, we examine the generation time for the TCP-SYN DoS and DDoS attack as well as the classic PoD attack. The results suggest that the prototype is able to effectively perform the injection even when a large number of packets are injected.

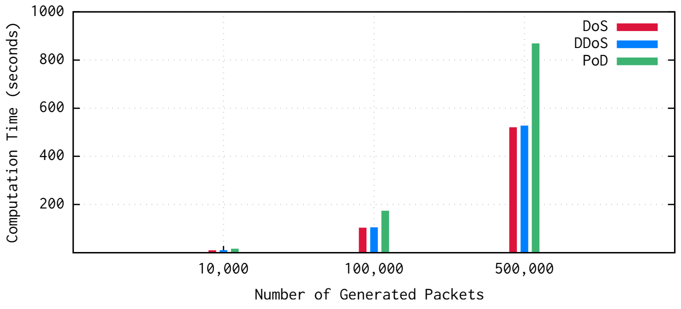


Fig. 4. Attack generation time with respect to the number of generated packets

Table I shows four different experiments, each of them conducted with different input datasets (as normal traffic) and injected attacks. For the first three, a 2GB slice of the MAWI dataset [7] was used, while the last (fourth) assessment is based on 13.8GB from the same dataset. In the first experiment, the 2GB background data is merged with $500,000$ TCP SYN packets and $1,024$ Port Scan packets. The second assessment is similar to the first one but includes an additional $2,724$ packets generated by the Angler malware. The last 2GB merging evaluation uses the four captures similar to the previous two scenarios but adds $6,704$ packets generated by the Aurora exploit. Finally, for the 13.8GB input dataset, all four previous attacks are merged and utilized. In all cases, the timestamps of the attacks have been modified so that they fit within the time of the input dataset. As a result of the merging process, we observe a non-linear increase in the computation time. The required time is, nonetheless, reasonable enough so as to be able to use large network packet capture files.

Input Dataset Size	TCP SYN (packets)	Port Scan (packets)	Exploit 1 (packets)	Exploit 2 (packets)	Total Time (seconds)
2GB	500,000	1,024	-	-	19.884
2GB	500,000	1,024	2,724	-	20.156
2GB	500,000	1,024	2,724	6,704	20.638
13.8GB	500,000	1,024	2,724	6,704	199.128

TABLE I
OVERALL MERGING PERFORMANCE OF ID2T FOR VARIOUS DATASETS AND ATTACKS

B. Avoiding Artifacts

It is highly important to take specific quality requirements into account when synthetic attacks are injected into real packet captures. In the following, we examine quality requirements and discuss how ID2T avoids adding undesired artifacts.

1) TTL Distribution: As discussed in Section III, it is important to take into consideration the packets' TTL distribution of a given network file before injecting attacks into it. We compare the packets' TTL distribution before and after attacks are injected. Figure 5 depicts a TTL distribution comparison of one arbitrary network packet capture file from the MAWI dataset against the same file after having being injected with multiple attacks. As one can observe, the statistic module correctly derives the TTL distribution of the input file and, thus, the toolkit was able to generate a dataset that closely resembles the original distribution.

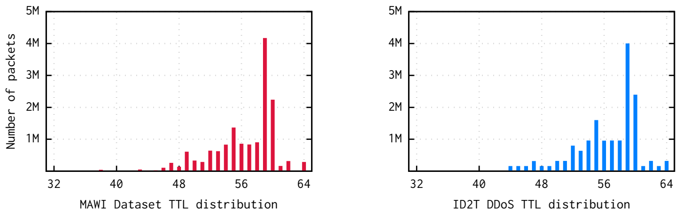


Fig. 5. TTL distribution comparison of the Mawi dataset and ID2T

2) Consecutive Packets Time: In a real world network, packets take different paths depending on the routing information or the load of routing devices. Similarly, some packets might be dropped or lost so that they never reach their final destination.

We examine such network communication characteristics and model the inter-arrival packet time. Ideally, the time between consecutive packets (Δ time) should follow a burst behavior in which large number of packets are sent and received in a short time interval. There are usually, however, distinct time clusters corresponding to delayed packets or network congestion protocols doing what they can to alleviate congestion, among other communication issues.

Figure 6 depicts the case of the Δ time for 1000 consecutive packets of a TCP-SYN DoS attack generated by ID2T. A packet rate of 10 packets per second is used for injection. This implies that the average time in between 10 packet timestamps should be approximately one second. To model realistic looking inter-arrival packet intervals, ID2T utilizes various randomization functions which make use of time distributions extracted by the statistics module. Similarly, Figure 7

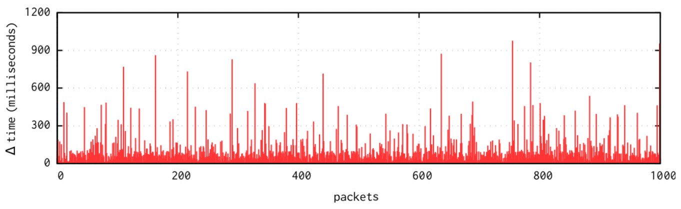


Fig. 6. Modeling time between two consecutive packets with a 10 p/s rate

presents the same Δ time for a DoS attack but with an increase packet rate of 100 packets per second. In both experiments, the results suggest that the packet time distribution emulates the characteristics of real network data as the distribution follows a burst behavior.

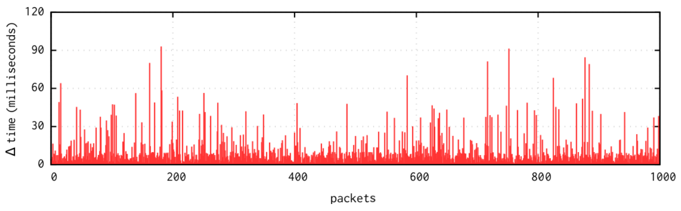


Fig. 7. Modeling time between two consecutive packets with a 100 p/s rate

3) Packet Head Checksum Calculation: The packet integrity of many protocols, e.g., IP and TCP, is guarded by the usage of checksums. Checksum calculations are important in the context of a synthetic dataset generator as inconsistencies can introduce significant defects. For instance, such a case was discovered by [11] with regard to the DARPA dataset. As a result of this defect, anomaly detectors were able to differentiate between normal and malicious traffic by identifying checksum inconsistencies.

To overcome such problems, checksum values are instantly computed when attacks are injected into a network file. In other situations, such as pcap capture modification, where packet header information has to be forged according to user-defined parameters, another approach is taken. As the checksum is related to the packet's header data, any alternation of it will lead to changes in the respective checksum value. Therefore, ID2T recalculates the checksums of every packet that has been modified. We manually injected packets containing incorrect IP and TCP checksums into pcap files with external tools. Afterwards we processed these files with ID2T and proceeded to further inject them with synthetic attacks. The resulting output was thoroughly checked with Wireshark [14]. No incorrect checksums for either the IP or the TCP protocols where found.

4) IP selection and distribution: During the injection of attacks, the selection of source IP addresses, their distribution and randomization, are of high importance in order to create realistic datasets. ID2T handles this by first modeling the user input (i.e., a supplied pcap file) and then suggesting source IP addresses as potential attack targets. In addition, the toolkit offers the ability to add different weights of occurrences for certain IP addresses. For instance, this might be utilized to model attackers with different computational resources. Lastly, the toolkit excludes certain IP addresses from the selection process (e.g., an IP that starts with 192.168 must not be used as the source IP of a DDoS attack).

VI. Discussion

The performance and qualitative evaluation for the prototype of ID2T is promising. The prototype is able to handle large network files in a reasonable time. In addition, the examination of various parameters suggests that the generated datasets contain realistic properties. There are a multitude of challenges that remain, however, in order to eliminate as many artifacts as possible from synthetically injected attacks.

First, we argue that there is a need for novel metrics to measure the quality of synthetically generated datasets. For instance, such a metric could be formally defined as a function that includes a weighted composition of all possible parameters that may introduce undesired artifacts. Second, a more in-depth investigation into the *quality* requirements, beyond the ones already identified in this paper or in related work, is needed in order to establish how resulting datasets are affected.

Many functionalities of the ID2T prototype currently include steps that users have to manually perform (for instance, using the results of the statistics analysis). We plan to better interconnect and automate the process of selecting parameters from the statistics collected during the static analysis of the user packet capture files. Additionally, we intend to further develop the attack controller module; focusing on producing a more flexible method of creating custom and novel attacks.

VII. Conclusion

Due to the increasing sophistication of cyber-attacks, if countermeasures are to be developed, research in the area of IDSs is required. The availability of adequate datasets, nonetheless, has always been a major weakness in this field. The scientific community requires modern, publicly available, datasets to be able to assess the novelty of old and new IDSs. We present ID2T, a prototype toolkit for generating datasets intended for the evaluation of intrusion detection algorithms and systems. This publicly available toolkit can inject synthetic, yet realistic, attacks into large network packet capture files. Our toolkit outputs a labeled packet capture file that avoids introducing artifacts or defects beyond the desired attacks. The evaluation shows that ID2T can handle large network files and that numerous artifacts that were identified in related work have been avoided.

As future work we aim at further enriching ID2T with more properties and features, e.g., more sophisticated attacks. At the same time, we are still evaluating other sources of undesired artifacts that are typically present in intrusion detection datasets and how to avoid them. Furthermore, the limitations and concerns identified in Section VI are being tackled and the improvements to the toolkit will be made available as development continues. Lastly, additional support for IPv6 is envisioned.

References

[1] Caida: The cooperative association for internet data analysis. http://www.caida.org, 2015.

[2] Philippe Biondi. Network packet manipulation with scapy, 2007.

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

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A Survey. *ACM Computing Surveys*, 41(3):1–58, jul 2009.

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

[6] Gideon Creech and Jiankun Hu. Generation of a new IDS Test Dataset : Time to Retire the KDD Collection. In *Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013.

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

[8] Robert Koch, Mario Golling, and Gabi Dreo Rodosek. Towards Comparability of Intrusion Detection Systems: New Data Sets. In *TERENA Networking Conference*, page 7, 2014.

[9] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[10] Matthew V Mahoney and Philip K Chan. An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection. *International Symposium on Recent Advances in Intrusion Detection*, 2820:220–237, 2003.

[11] Matthew V. MV Mahoney and PK Philip K. Chan. PHAD: Packet Header Anomaly Detection for identifying hostile network traffic. Technical Report 1998, Florida Institute of Technology, 2001.

[12] J McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM transactions on Information and system Security*, 3(4):262–294, 2000.

[13] Milanov Nikolay, Vasilomanolakis Emmanouil, and Garcia Cordero Carlos. The id2t: Intrusion detection dataset toolkit. https://www.tk.informatik.tu-darmstadt.de/de/research/secure-smart-infrastructures/id2t/, 2015.

[14] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

[15] Benjamin Sangster, Thomas Cook, Robert Fanelli, Erik Dean, William J Adams, Chris Morrell, and Gregory Conti. Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In *USENIX Security's Workshop on Cyber Security Experimentation and Test (CSET)*, 2009.

[16] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali a. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3):357–374, 2012.

[17] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Symposium on Computational Intelligence for Security and Defense Applications, CISDA*, number Cisda, pages 1–6. IEEE, 2009.

[18] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys*, 47(4):33, 2015.