USB-IDS-TC: A Flow-Based Intrusion Detection Dataset of DoS Attacks in Different Network Scenarios

Marta Catillo[a], Antonio Pecchia[b] and Umberto Villano[c]
Università degli Studi del Sannio, Benevento, Italy
{*marta.catillo, antonio.pecchia, villano*}*@unisannio.it*

Keywords: Dataset, Intrusion Detection, Denial of Service, Network Flows, Traffic Control, Network Emulation.

Abstract: Network intrusion detection systems (NIDS) play a key role for cybersecurity. Most of the times, NIDS are built on machine learning/deep learning (ML/DL) models that are trained and tested on public intrusion detection datasets. This paper presents the novel USB-IDS-TC dataset, conceived to explore the dependence of ML/DL-based NIDS on the network used to collect the training traffic data. In this new publicly-available dataset, DoS attacks have been conducted in different network scenarios, in the belief that the network has a non-negligible effect on the detection capability of the NIDS as indicated by our initial analysis. Differently from existing datasets that collect the data in a single scenario, USB-IDS-TC allows studying the dependence of the attacks, traffic features and ML/DL models on the network, in order to strive for generalizable and widely-applicable NIDS.

1 INTRODUCTION

In the desperate and presumably endless struggle against network hackers and misusers, network intrusion detection systems (NIDS) currently play a key role. The detection of potentially dangerous network activity is canonically carried out by means of policy rules and signatures based on known attacks. Unfortunately, this requires frequent signature updates and is mostly ineffective against never-seen-before attacks (*0-day attacks*). This is the primary reason for the blossoming of an ever-increasing body of research on machine learning/deep learning (ML/DL) detectors, which aim to infer the class of network traffic or to detect anomalies by comparing the network traffic to a legitimate baseline. Whenever a significant difference is detected, an alert is raised. The hope of the scientific community working on this topic is that anomaly-based detectors will be able to detect also 0-day attacks, as they should deviate from normal network traffic.

Most of the times, ML/DL NIDS are trained and tested on public intrusion detection datasets. As a matter of fact, public datasets, such as KDD-CUP'99 (Özgür and Erdem, 2016), UNSW-NB15 (Moustafa and Slay, 2015), NDSec-1 2016 (Beer and Buehler, 2017), CICIDS2017 (Sharafaldin et al., 2018), and many others (Ring et al., 2019) have become the *de facto* standard benchmarks for evaluating novel NIDS techniques. The wide availability of intrusion datasets, together with the rapid advancement of deep learning frameworks has led to the emersion of numerous attack detection methods in the literature. Notably, some of these detectors achieve highly promising detection rates, approaching 100% on public datasets. It has been argued elsewhere that many studies leveraging public datasets for NIDS research tend to "blindly" trust the data without considering the representativeness of the network traffic and its potential cybersecurity implications, such as the actual impact on service continuity and performance of the targeted applications (Catillo et al., 2021b). For this reason, in the past our research group, based at the University of Sannio in Benevento (USB), Italy, has released a dataset (USB-IDS-1[1]) where the network data collected was complemented with (i) *performance measurements* of the victim under attack to make it clear if the attack was actually successful in disrupting the victim service and (ii) the actual *configuration* of the victim server (capacity, multithreading capability and potential defense mechanism enabled, if any).

[a] https://orcid.org/0000-0002-5025-7969
[b] https://orcid.org/0000-0003-2869-8423
[c] https://orcid.org/0000-0001-5382-4650

[1] https://idsdata.ding.unisannio.it/usbids1.html

Catillo, M., Pecchia, A. and Villano, U.
USB-IDS-TC: A Flow-Based Intrusion Detection Dataset of DoS Attacks in Different Network Scenarios.
DOI: 10.5220/0013248600003899
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 1*, pages 302-309
ISBN: 978-989-758-735-1; ISSN: 2184-4356
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

Following up our past work on these topics, this paper presents the novel USB-IDS-TC dataset, which addresses a different – and strongly overlooked by the literature – issue: *the dependence of the ML/DL-based NIDS on the network scenario used to collect the training traffic data*. USB-IDS-TC is motivated by the fact that the majority of the ML/DL NIDS do not examine individual network packets, but rely on bidirectional traffic flows. Flows are obtained by hardware (e.g., routers) or by software from the packets exchanged in the two directions pertaining to the same connection. Each flow consists of a record of features suited for ML/DL purposes. The extracted features include *packet and payload length*, *number of packets*, *transmitted bytes* and *mean length of packets*, along with statistical measurements of the **timing** of the communication. ML/DL NIDS exploit heavily time-related features, such as the interarrival times of forward and backward packets making up the flow. For example, the ubiquitous flow extractor CICFlowMeter, originally named ISCXFlowMeter (Draper-Gil. et al., 2016), in its most recent version generates bidirectional flow records made of 93 features: it must be noted that around 30% of features are timing-related. In consequence, it is almost natural to wonder if different network scenarios with different bandwidth and latency – which surely impact the timing-related features – would affect the detection capability of ML/DL NIDS.

In response to the challenge presented, USB-IDS-TC provides network flow data – both normal traffic and Denial of Service (DoS) attacks – obtained in different network scenarios. The dataset is publicly available on our web site[2]. To the best of our knowledge, it is the first time that the flow data relative to the same normal traffic and DoS attacks are collected in different network scenarios. Our hope is that the dataset could be beneficial to the NIDS community, making clear the possible hidden effect of network characteristics. USB-IDS-TC allows studying the dependence of the attacks, traffic features and ML/DL models on the network scenario, in order to strive for generalizable and widely-applicable NIDS.

The paper is organized as follows. Section 2 discusses related work in the area and the original contribution of USB-IDS-TC. Section 3 describes the experimental testbed and its emulation capabilities, the attacks performed and the collection procedure. Section 4 discusses the network scenarios and the dataset organization. The key insights learned from our data and possible future research directions are presented in Section 5. In Section 6 we draw our conclusions.

[2]https://idsdata.ding.unisannio.it/usbidstc.html

2 RELATED WORK

Public intrusion datasets have boosted the academic research on NIDS. Typically, these datasets are accessible in a raw format, such as PCAP packet data files, or in a more "refined" format, such as network flows organized in comma-separated values (CSV) files. These CSV files are ideally suited for ML applications, which is the reason for the extensive use of public datasets. A number of these datasets have achieved notable popularity within the literature.

For example, KDD-CUP'99[3] can be considered the "pioneer" for ML-based intrusion detection. It was collected in 1999 and consists of two weeks of instances free from attacks and five weeks of instances containing attacks. It is important to note that, despite its continued popularity (Kushwaha et al., 2017) and status as a foundational contribution to the field of intrusion detection, KDD-CUP'99 has several documented drawbacks (McHugh, 2000). Additionally, after approximately two decades, it is no longer an accurate representation of present-day network traffic. This also applies to the more recent NSL-KDD dataset (Tavallaee et al., 2009), a version of KDD-CUP'99 with reduced size and duplicate entries removed. In recent years, there has been a growing trend towards critical analysis of security datasets. For example, (Silva et al., 2020) identified statistical flaws within the KDD-CUP'99 dataset that could introduce bias during training of IDS models.

Among the latest publicly available intrusion detection datasets, CICIDS2017 is undoubtedly the one that has gained the greatest popularity. Released by the Canadian Institute for Cybersecurity (CIC), its reference paper (Sharafaldin et al., 2018) is currently (November 2024) cited almost 4000 times on Google Scholar, which places it among the most frequently used datasets. A testbed framework was implemented by the authors to generate benign and attack data systematically using different profiles. The dataset offers both ready-to-use labeled flows and raw PCAP files. Furthermore, the authors have developed CICFlowMeter (Lashkari et al., 2017), a tool to generate network flows from raw PCAP files, which has quickly become very popular. Regrettably, it was not until after a long period of "blind" utilization of the CICIDS2017 dataset by NIDS researchers that a number of studies identified major bugs and errors affecting CICFlowMeter, resulting in incorrect flow records from both CICIDS2017 and the younger CSE-CIC-IDS2018[4] (Engelen et al., 2021; Rosay et al., 2022; Liu et al., 2022; Lanvin et al., 2023).

[3]https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[4]https://registry.opendata.aws/cse-cic-ids2018/

Another widely known public intrusion detection dataset is UNSW-NB15 (Moustafa and Slay, 2015). Created by the Australian Centre for Cyber Security (ACCS), it contains both real normal activities and synthetic attack behaviors. The dataset is available in both CSV and raw PCAP formats.

More recently, there has been a shift in focus away from general purpose networks and towards attacking networks designed for specific applications. For example, the aforementioned CIC proposed datasets for IoT, IoMT and IoV environments (Neto et al., 2023; Dadkhah et al., 2024; Neto et al., 2024), and for electric vehicle (EV) charging infrastructures (Kim et al., 2023; Buedi et al., 2024).

Our Contribution. In our previous work (Catillo et al., 2021a), we introduced a novel dataset that took into account both performance metrics and application-level facets, including "accessory" parameters of the experimental testbed such as commodity defense mechanisms. Here instead we present a dataset that focuses on the influence of the network scenarios on the traffic data. USB-IDS-TC captures well known DoS attacks under diverse network scenarios, enabling researchers to investigate the impact of network characteristics on intrusion detection performance of ML/DL-based NIDS. To the best of our knowledge, this is the first dataset of its kind, preparing the way for a deeper understanding of the transferability of intrusion detection methods.

3 TESTBED

It is not realistically feasible to set up a testbed with enough hardware to provide the wide range of performances characterizing current communication networks. In order to perform the packet capture for our dataset, we decided to resort to an extensively-configurable testbed environment, which makes it possible to reproduce the behavior exhibited by most real-life networks. The environment is based on Docker containers, canonically connected through a user-defined internal network that employs the *bridge* network driver. It is crucial to note that the *bridge* Docker network is characterized by several key attributes: very high bandwidth[5], minimal latency, and absence of transmission errors (after all, it not a "real" network). Consequently, to introduce network conditions that are representative of real-world scenarios, it is only necessary to exploit techniques such as *bandwidth shaping*, injection of variable *latency*, and possibly inclusion of *transmission errors*.

[5]The `iperf3` utility reports 23.6 Gbits/s on the workstation used to collect the dataset

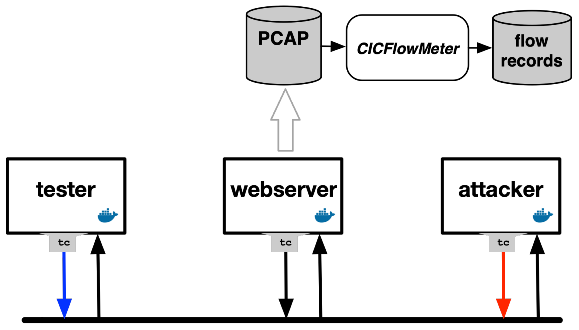


Figure 1: Experimental testbed.

The Internet Engineering Task Force (IETF) Request for Comments (RFC) 2475[6] defines **traffic shaping** as the act of delaying packets within a traffic stream. This delay is introduced to ensure that the stream conforms to a predefined *traffic profile*. A traffic profile is a specification that details the temporal properties of a traffic stream, such as its rate (data transmission speed) and burst size (maximum amount of data transmitted in a short period).

Our implementation relies on the use of the Linux `tc` tool. The `tc` tool is included within the `iproute2` suite[7], a collection of user-space utilities designed for controlling the networking functionality of the Linux kernel. The Linux manual page for the `tc` tool[8] suggests that shaping goes beyond simply adjusting bandwidth. In fact, `tc` offers functionalities that extend beyond basic bandwidth control. These functionalities include, but are not limited to, shaping *outgoing* network traffic by modifying the output rate and introducing packet delay (which can be fixed, or incorporate a suitably-distributed jitter). Additionally, `tc` allows for the introduction of packet loss, duplication and corruption.

3.1 Components of the Testbed

The detailed structure of the experimental testbed is shown in Fig. 1. Each container within the testbed executes a Debian 11 bullseye Linux instance. These containers are assigned specialized functions as outlined below:

- **webserver:** This container runs an Apache2 web server (version 2.4.57, default configuration). The web server receives normal traffic originating from the *tester* node and Denial-of-Service (DoS) traffic sent by the *attacker*. The container also runs the utility `tcpdump`, which captures all net-

[6]https://datatracker.ietf.org/doc/html/rfc2475
[7]https://git.kernel.org/pub/scm/network/iproute2/iproute2.git
[8]https://man7.org/linux/man-pages/man8/tc.8.html

Table 1: Parameters of the network scenarios.

Network	rate	delay	jitter	loss	corruption	duplication
1 - Enterprise wi-fi	1 Gbit/s	5 ms	1 ms	-	-	-
2 - Enterprise Branch Office	1 Gbit/s	20 ms	8 ms	-	-	-
3 - Site-to-Site VPN	1 Gbit/s	20 ms	15 ms	-	-	-
4 - Remote User VPN	100 Mbit/s	20 ms	15 ms	-	-	-
5 - Degrated	100 Mbit/s	20 ms	25 ms	5 %	5 %	2 %

work traffic received by the webserver, and saves it in PCAP format in permanent storage.

- **tester:** This container runs a load generator built on top of the well-known `httperf`[9] utility. First, as for the normal traffic, it is used to issue randomized web requests; second, the tester collects response statistics to check the availability/disruption of the Apache server functionality during DoS attacks.
- **attacker:** This container generates DoS attacks against the *webserver* by means of well known, public-available tools, which are described below.

As mentioned above, the `tc` tool makes it possible to control only *outgoing* network traffic. Hence the set-up of the testbed for the emulation of any given network requires to launch `tc` on the outgoing links of the three containers with the same settings, as reported in Fig. 1. The network scenarios reproduced in USB-IDS-TC are presented in Section 4.

3.2 Normal and DoS Traffic

Normal Traffic. The scripted load generator executed at the tester node exploits `httperf` to issue requests to the Apache web server for assorted contents (*small*, *medium* and *large* HTML files, images, PDF documents, ...). The normal traffic collection has a duration of 20 minutes. As for the **DoS traffic**, the attacker node launches four different DoS attacks (HTTP flood, two kinds of slowloris, slow POST) against the webserver, by the following tools:

- `hulk`[10]: it generates an HTTP flood, spawning a large volume of obfuscated and unique requests to prevent the recognition of a pattern that could allow the filtering of the anomalous traffic;
- `slowloris`[11]: a tool producing DoS traffic based on slow HTTP requests against the victim server, effective in the exploit of a weakness of the HTTP protocol related to the management of TCP fragmentation;
- `slowhttptest`[12]: this tool can extend anomalously the duration of HTTP connections in different ways. For the production of our dataset data, we use `slowhttptest` both in the (i) "slowloris" mode, which sends incomplete HTTP requests to the victim server, and (ii) "slow POST" mode, which sends message bodies at very slow speed.

The DoS tools, *one at a time*, are launched and kept running for 180 seconds, enough to disrupt the webserver service and to collect a significant sample of attack traffic.

4 DATASET COLLECTION AND ORGANIZATION

Normal and DoS traffic is collected in four representative network scenarios from (Fulkerson, 2017): *Enterprise Wi-Fi*, *Enterprise Branch Office*, *Site-to-Site VPN*, *Remote User VPN*. In addition, we constructed a fifth scenario representing a severely degraded network. This was deliberately selected to highlight the discrepancies from any network with satisfactory performance, and to examine the influence on the accumulated traffic flows. The configuration specifics of `tc` for the five networks are shown in Table 1.

In order to avoid any form of mislabeling – a problem affecting many datasets currently in use – the traffic capture for each network scenario is performed with 5 independent experiments: (1) the normal traffic, whose capture is named NOR, and (2, 3, 4, 5), i.e., each individual DoS attack, leading to four additional captures named HLK (`hulk`), GSL (`slowloris`), HSL (`slowhttptest` in slowloris mode) and HSP `slowhttptest` in slow POST mode. As mentioned before, during the DoS attacks, the tester runs `httperf`, which continuously sends HTTP requests to the webserver and records information on the service availability. This *probing* HTTP traffic (easily recognizable by its source node address) is not recorded by `tcpdump`. In consequence, the traffic collected in (2, 3, 4, 5) consists of "pure" attack packets,

[9]https://github.com/httperf/httperf

[10]https://github.com/grafov/hulk

[11]https://github.com/gkbrk/slowloris

[12]https://tools.kali.org/stress-testing/slowhttptest

Table 2: Number of flows by network scenario and capture.

Network	NOR	HLK	GSL	HSL	HSP
1	1658	79399	4072	5211	5244
2	1725	86281	3279	5107	5100
3	1677	82200	2795	5112	5126
4	1671	88525	2861	5102	5125
5	3285	98695	686	5494	5438

not interleaved with normal traffic (present during the attack, but not recorded at all). The complete separation of normal and attack traffic is useful to avoid any possibility of mislabeling the generated flows.

Overall, the five captures – conducted in the five network scenarios assessed – lead to a total 25 PCAP files. The PCAP files obtained have been processed with the `CICFlowMeter` tool (see Fig. 1), which is used to obtain the corresponding flow records. It is noteworthy that the original `CICFlowMeter`, which was utilized to generate the initial CICIDS2017 dataset, a widely used-resource in the machine learning community, was affected by a number of bugs that had a significant impact on the consistency of the resulting flow records. We have adopted a revised version of the tool[13] produced from independent studies (Liu et al., 2022). The number of flow records for each network scenario and each class of traffic is presented in Table 2. Although the duration of all the DoS traffic captures is identical, it is evident that `hulk` – producing a flood attack – generates a considerably higher number of flows than the slow counterparts. Furthermore, the introduction of packet errors and duplications during the capture of the network number 5 increases the number of flows of normal traffic and results in a delayed impact (only 686 GSL flows) of the `slowloris` script.

USB-IDS-TC is released in the form of five `csv` files, where each file provides normal and DoS flow records of one network scenario in Table 1. Each `csv` file provides ready-to-use **labeled** network flows, obtained appending the five previously-labeled flow collections relative to the same network scenario. The labels are the abbreviations already used in Table 2 (NOR, HLK, GSL, HSL, HSP).

General Observations and Use Cases of USB-IDS-TC. The dataset is not meant to be the "ultimate" solution for NIDS testing, but a first step to promote an in-depth understanding of the possible effect of the network scenarios on the performance of ML/DL NIDS, issue commonly neglected at the state of the art. It is worth pointing out that the choice of using `CICFlowMeter` to obtain the traffic flows makes USB-IDS-TC immediately interoperable with the high number of NIDS proposals based on the use of CICIDS2017 and other major datasets of the CIC collection. Moreover, any captured PCAP file processed by `CICFlowMeter` can be used in conjunction with USB-IDS-TC, paving the way for a study of the transferability of NIDS models over different network scenarios. As an aside, the proposed Docker/`tc` testbed presented here can also be used to generate realistic problem-space adversarial attacks by altering the timing of the packets sent by an attacker node (Catillo et al., 2024).

[13]https://github.com/GintsEngelen/CICFlowMeter

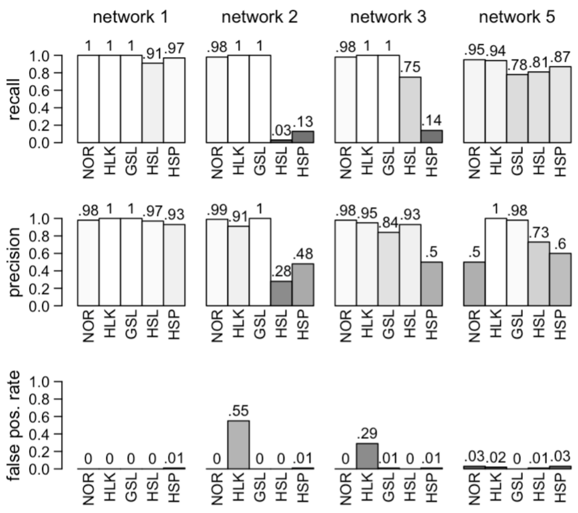
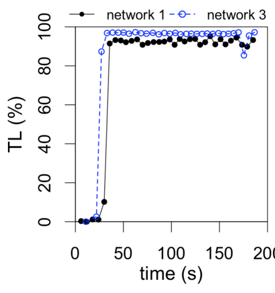


Figure 2: Recall, precision and false positive rate of an IDS model learned from network 1 and applied to network 1, 2, 3, and 5.

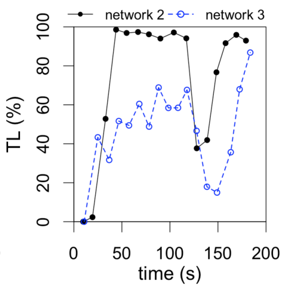
5 KEY INSIGHTS

5.1 Intrusion Detection Implications

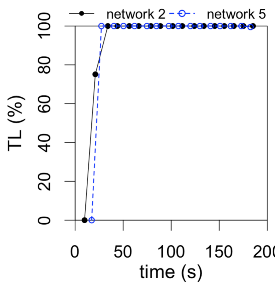
Our critical argument is that the specific network scenario influences the traffic data and – in turn – the value of the features extracted (especially those timing-related). In consequence, an attempt to learn a NIDS with the flow records obtained in a given network may return a detection model that *does not* transfer to the normal and DoS traffic of the other networks. Let us supplement this argument by a concrete example with USB-IDS-TC. At first, we learn a NIDS model with the flow records obtained in *network 1*. As for any ML/DL experiment, we remove non-relevant and biasing features (i.e., id, timestamp and protocol of the flow records, source and destination IP address and port) and split the records obtained in *network 1* into the typical training, validation and test set (60, 20 and 20% of the records, respectively). Regarding the



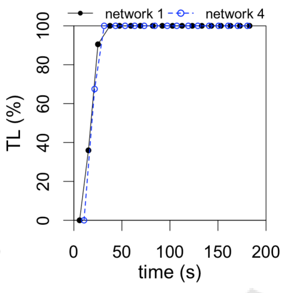
(a) hulk



(b) GSL slowloris



(c) HSL slowloris



(d) slow POST

Figure 3: Throughput measured during the progression of the attacks for some network scenarios.

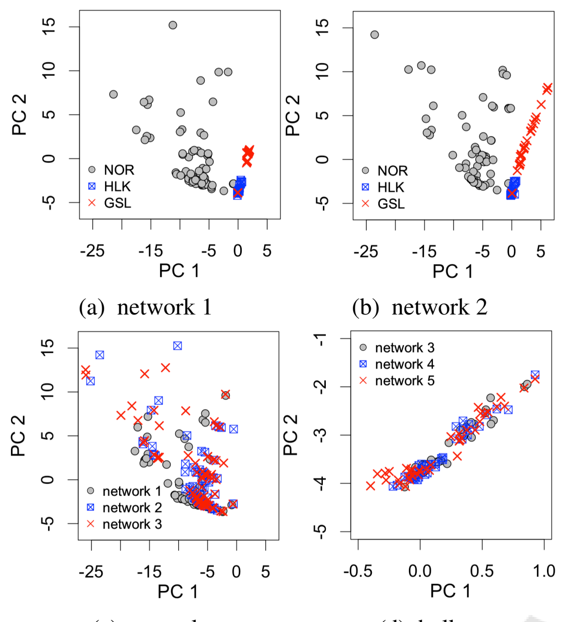
technique to infer the NIDS, we use a decision tree; however, any other classifier, e.g., Bayesian network, oneR or DNN (to mention a few), would have fit the scope of this example. The decision tree is trained with the training set; parameterization and overfitting issues are checked with the validation set.

The NIDS model obtained is tested with the test set of *network 1* (held-out from training and validation) and the entirety of records of *network 2*, *3* and *5* (*network 4* is not reported because the results are close to network 2). We compute the metrics of recall (R), precision (P) and false positive rate (FPR) to measure the capability of the NIDS at recognizing the classes of traffic among NOR, HLK, GSL, HSL, HSP. Fig. 2 shows the values of the metrics. As expected, the model – when trained and tested with the flow records collected in the same network scenario (as typically done in most of the NIDS papers) – achieves more than satisfactory results. The leftmost set of five bars of Fig. 2 (pertaining to *network 1*) indicate that four classes are detected with both R and P $\geq$0.97; the FPR is almost 0. The metrics obtained for *network 2*, *3* and *5* are shown by the remaining bars in Fig. 2. Compared to *network 1*, the drop is significant in many cases. For example, the recall of HSP is $\leq$0.14 in both *network 2* and *3*; as for *network 5*, three classes are far below 0.9 recall. Similar findings can be noted for P, which drops in many cases, e.g, HSL (*network 2* and 5), HSP (*network 2*, *3* and 5), NOR (*network 5*). FPR reaches the extremely high values of 0.55 and 0.29 for HLK (*network 2* and *3*).

Suggested Research Directions. The lack of *transferability* of ML/DL-based NIDS models is overlooked by the NIDS literature. This is exacerbated in the context of USB-IDS-TC because both normal and DoS traffic are indeed the same, although executed in different network scenarios. Our data can help researchers to strive for more generalizable and widely-applicable detectors. In this respect, future usages of USB-IDS-TC may include (but not limited to) the analysis of: the networks leading to more generalizable detectors, the attacks being most affected by the network parameters, the robustness of existing ML/DL techniques – and learning paradigms – with respect to the network, the features depending on the network or the construction/selection of more general features suited for detection.

5.2 Effectiveness of the Attacks

The reader may argue that the efficacy of the attacks might be affected by the specific network scenario; however, this is not the case of USB-IDS-TC. Differently from many existing datasets (that *do not* disclose any specific victim-side service availability measurements), we monitored the performance of the victim server during the progression of the attacks. As said above, in response to the probing HTTP load, `httperf` generates several service metrics. Here we provide some insights into the effectiveness of the attacks in USB-IDS-TC by discussing the *throughput loss*. The **throughput loss** (TL) is computed as $TL{=}\frac{T^{*}-T}{T^{*}}\cdot 100$ (with $T^{*}{\geq}T$ and $T^{*}{>}0$), where (i) T^{*} – a constant – is the throughput (i.e., successful `2xx` HTTP requests accomplished within the time unit) expected in attack-free conditions for a given network scenario and (ii) T is the actual throughput measured during a DoS attack. TL varies within $[0,100]$%, where 0 denotes “no loss” with respect to the attack-free condition. Any point where TL>0% indicates instead the presence of a DoS attack, because in our testbed the only source of legitimate requests is the tester node. Fig. 3 shows TL observed during the attacks; for each attack, we show two networks because all the cases produce similar results. Overall, the attacks cause a variety of responses. For example, TL raises from 0% to 90-95% in hulk (Fig. 3a), which means the attacks leaves almost no room to serve the legitimate requests; on the other hand, GSL slowloris induces major fluctuations of TL, as shown in Fig. 3b. HSL slowloris (Fig. 3c) and slow POST (Fig. 3d) present an on-off behavior, where the throughput goes from 0 to 100% in almost no time. The attacks are effective for all the network scenarios assessed.



(a) network 1 (b) network 2

(c) normal (d) hulk

Figure 4: PCA-based visualization of the flow records.

5.3 Visual Inspection

One more interesting finding on the traffic in USB-IDS-TC can be inferred through a visual inspection of the flow records. We conduct a Principal Component Analysis (PCA) to visualize the flow records in the feature space. PCA is a dimensionality reduction technique whose objective is to find the directions along which a set of high-dimensional points line up best. Flow records are regarded here as $\mathbb{R}^{86}$ points of a Euclidean space, where 86 is the number of features after removal of the label and non-relevant/biasing features. We retain the top 2 principal components (PC) explaining almost 40% of the total variance: this is highly satisfactory for 2D visualization purposes.

Fig. 4a and 4b show the records pertaining to three classes of traffic (i.e., NOR, HLK and GSL) for *network 1* and *network 2*, respectively. It can be noted that the classes are "well" separated, which means it is possible to infer a successful NIDS model on the top of an individual network scenario; however, the model obtained will not transfer – as shown in Section 5.1 – to a different network scenario. This aspect remains surely intriguing because the classes of traffic preserve their feature-space distribution across the network scenario. For example, Fig. 4c shows the normal flow records obtained in *network 1*, *2*, and *3*, each marked by ○, ⊠ and ×, respectively. The normal points obtained in the different networks are distributed over the same area: the different networks induce a "light" shift of the points. Similar considerations can be done for the hulk records in Fig. 4d, shown for *network 3*, *4*, and *5*.

We believe that the availability of different variants of normal and DoS traffic across different network scenarios, such as those in USB-IDS-TC, is strongly beneficial to the research community to learn more flexible detection models or to test the transferability of a given NIDS proposal.

6 CONCLUSION

Intrusion detection is a hot topic, and the research on NIDS should be fed with consistent and up-to-date datasets. The scientific community tends to rely on rather obsolete datasets, possibly obtained by collecting traffic relative to attacks that are not actually harmful against targeted services. Mislabeling is a further issue. In light of the above, new public datasets providing effective attacks are required.

Compared to existing dataset proposals, our work has gone in a novel direction. We have built a new dataset where the same – well known and rather customary – DoS attacks have been conducted over different network scenarios, in the belief that network has a non-negligible effect on traffic features and the detection capability of the NIDS. Furthermore, we have tested the effectiveness of all the attacks. Our initial analysis shows that the network scenario can affect the capability of ML/DL detection.

We believe that the USB-IDS-TC dataset can stimulate the research on the transferability of intrusion detection methods. Furthermore, the emulation environment presented here lends itself to the study of realistic problem-space attacks performed by altering the timing of the attack packets sent. These topics will be explored in our future work, which intends also to provide the community with additional datasets relative to non-DoS attacks and problem-space adversarial examples over multiple network scenarios.

ACKNOWLEDGMENT

This work has been partially funded by the European Union – Next-GenerationEU – National Recovery and Resilience Plan (NRRP) – MISSION 4 COMPONENT 2, INVESTMENT N. 1.1, CALL PRIN 2022 PNRR D.D. 1409 14-09-2022 – (Threat-driven security testing and proactive defense identification for edge-cloud systems) CUP N. F53D23009270001 (CUP Master N. E53D23016380001).

REFERENCES

Beer, F. and Buehler, U. (2017). Feature selection for flow-based intrusion detection using rough set theory. In *Proc. Int. Conf. on Networking, Sensing and Control*, pages 617–624. IEEE.

Buedi, E. D., Ghorbani, A. A., Dadkhah, S., and Ferreira, R. L. (2024). Enhancing EV charging station security using a multi-dimensional dataset: CICEVSE2024. In Ferrara, A. L. and Krishnan, R., editors, *Data and Applications Security and Privacy XXXVIII*, pages 171–190, Cham. Springer Nature Switzerland.

Catillo, M., Del Vecchio, A., Ocone, L., Pecchia, A., and Villano, U. (2021a). USB-IDS-1: a public multilayer dataset of labeled network flows for IDS evaluation. In *2021 51st IEEE/IFIP Int. Conf. on Dependable Systems and Networks Workshops (DSN-W)*, pages 1–6.

Catillo, M., Pecchia, A., Rak, M., and Villano, U. (2021b). Demystifying the role of public intrusion datasets: A replication study of DoS network traffic data. *Computers and Security*, 108:102341.

Catillo, M., Pecchia, A., Repola, A., and Villano, U. (2024). Towards realistic problem-space adversarial attacks against machine learning in network intrusion detection. In *Proc. of the 19th Int. Conf. on Availability, Reliability and Security*, ARES '24. ACM.

Dadkhah, S., Neto, E. C. P., Ferreira, R., Molokwu, R. C., Sadeghi, S., and Ghorbani, A. A. (2024). CIcIoMT2024: A benchmark dataset for multi-protocol security assessment in IoMT. *Internet of Things*, 28:101351.

Draper-Gil., G., Lashkari., A. H., Mamun., M. S. I., and A. Ghorbani., A. (2016). Characterization of encrypted and VPN traffic using time-related features. In *Proc. of the 2nd Int. Conf. on Information Systems Security and Privacy - ICISSP*, pages 407–414. SciTePress.

Engelen, G., Rimmer, V., and Joosen, W. (2021). Troubleshooting an intrusion detection dataset: the CICIDS2017 case study. In *2021 IEEE Security and Privacy Workshops (SPW)*, pages 7–12.

Fulkerson, J. (2017). 9 sets of sample tc commands to simulate common network scenarios. https://www.badunetworks.com/9-sets-of-sample-tc-commands-to-simulate-common-network-scenarios/. Accessed: 2024-03-30.

Kim, Y., Hakak, S., and Ghorbani, A. (2023). DDoS Attack Dataset (CICEV2023) against EV Authentication in Charging Infrastructure . In *2023 20th Int. Conf. on Privacy, Security and Trust (PST)*, pages 1–9. IEEE.

Kushwaha, P., Buckchash, H., and Raman, B. (2017). Anomaly based intrusion detection using filter based feature selection on KDD-CUP 99. In *Proc. TENCON IEEE Region 10 Conference*, pages 839–844. IEEE.

Lanvin, M., Gimenez, P.-F., Han, Y., Majorczyk, F., Mé, L., and Totel, É. (2023). Errors in the CICIDS2017 dataset and the significant differences in detection performances it makes. In Kallel, S. and et al., editors, *Risks and Security of Internet and Systems*, pages 18–33, Cham. Springer Nature Switzerland.

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., and Ghorbani, A. A. (2017). Characterization of Tor traffic using time based features. In *Proc. International Conference on Information Systems Security and Privacy*, pages 253–262. SciTePress.

Liu, L., Engelen, G., Lynar, T., Essam, D., and Joosen, W. (2022). Error prevalence in NIDS datasets: A case study on CIC-IDS-2017 and CSE-CIC-IDS-2018. In *2022 IEEE Conference on Communications and Network Security (CNS)*, pages 254–262.

McHugh, J. (2000). 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.

Moustafa, N. and Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proc. Military Communications and Information Systems Conference*, pages 1–6. IEEE.

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., and Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13).

Neto, E. C. P., Taslimasa, H., Dadkhah, S., Iqbal, S., Xiong, P., Rahman, T., and Ghorbani, A. A. (2024). CICIoV2024: Advancing realistic IDS approaches against DoS and spoofing attack in IoV CAN bus. *Internet of Things*, 26:101209.

Özgür, A. and Erdem, H. (2016). A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. *PeerJ Preprints*.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. *Computers and Security*, 86:147–167.

Rosay, A., Carlier, F., Cheval, E., and Leorux, P. (2022). From CIC-IDS2017 to LYCOS-IDS2017: A corrected dataset for better performance. In *IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology*, WI-IAT '21, page 570–575. ACM.

Sharafaldin, I., Lashkari, A. H., and Ghorbani., A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proc. Int. Conf. on Information Systems Security and Privacy*, pages 108–116. SciTePress.

Silva, J. V. V., Lopez, M. A., and Mattos, D. M. F. (2020). Attackers are not stealthy: Statistical analysis of the well-known and infamous KDD network security dataset. In *Proc. Conf. on Cloud and Internet of Things*, pages 1–8. IEEE.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the KDD-CUP'99 data set. In *Proc. Symp. on Computational Intelligence for Security and Defense Applications*, pages 1–6. IEEE.