USB-IDS-1: a Public Multilayer Dataset of Labeled Network Flows for IDS Evaluation

Marta Catillo, Andrea Del Vecchio, Luciano Ocone, Antonio Pecchia, Umberto Villano
Dipartimento di Ingegneria
Università degli Studi del Sannio
Benevento, Italy
{marta.catillo, andrea.delvecchio, luciano.ocone, antonio.pecchia, villano}@unisannio.it

***Abstract*—The scarceness of real-life data derived from operational enterprise networks is a critical problem for the security research community. Therefore, public labeled datasets are necessary for evaluating anomaly-based Intrusion Detection Systems. In recent years, many public intrusion detection datasets have spread; however, they may lack relevant details concerning the victim applications. This paper describes USB-IDS-1, a novel public intrusion detection dataset developed at the University of Sannio at Benevento, Italy. It contains DoS attacks and considers both network traffic and application-level facets, such as server-side performance, configuration and defense modules of the victim server. The paper describes the collection environment, provides key insights into attack traffic and demonstrates the impact of the attacks adopted against the server in hand.**

***Index Terms*—Denial of Service, network flows, security, defense, web server**

I. Introduction

The increasing availability of **public intrusion detection datasets** is fostering measurement-driven studies by a wide community of academics and practitioners. Public datasets, such as KDD-CUP'99 [1], UNSW-NB15 [2], NDSec-1 2016 [3], CICIDS2017 [4] and many more –interested readers are referred to [5] for a survey– have become common benchmarks for evaluating novel Intrusion Detection System (IDS) techniques. For example, with the rapid growth of *deep learning* frameworks, many attack detectors have spread in the literature [6]; noteworthy, some of these detectors achieve very encouraging results in terms of detection rate, which is close to 100% on public datasets. We claim that the contributions that leverage public datasets for IDS research "blindly" trust and reuse existing data by overlooking the representativeness of the network traffic therein and its cybersecurity implications, i.e., in terms of the **real impact** on service continuity and performance of operations of the victim applications.

The way existing datasets are thought, collected and used should be significanlty changed. In a previous work [7] we demonstrated that Denial of Service (DoS) attacks from the widely-used CICIDS2017 dataset, although somewhat relevant at *network-level* –due to the straightforward abusive consumption of network bandwidth– are negligible at *application-level*, i.e., they do not significantly impact operations of the victim under a properly-tuned configuration. **Future datasets** must be conceived with a *multilayer perspective*, in order to make them suitable for both application- and network-level analysis, and to drive security claims on servers or IDS techiques. To this aim, network data should be enriched by **performance measurements** of the victim under attacks to make it clear whether abusive consumption of network bandwidth affected or not the victim at the time the dataset was collected. Moreover, it should be stated the **configuration** of the victim server, which pertains both to *capacity* and *multithreading capability* of the server, and –even more important– to any potential **defense** mechanism enabled at the time the attacks were conducted. Surprisingly, *none* of the major existing datasets addresses threats and bias induced by commodity defense modules that can be found in many real-life server installations.

This paper describes a novel public intrusion dataset developed at the University of Sannio at Benevento (USB), Italy, hence named **USB-IDS-1**. It is a concrete example of **multilayer dataset** encompassing both *network traffic* and *application-level facets*, such as server-side *performance*, *configuration* and *defenses*, which is a first step towards overcoming the limitations of existing datasets. The dataset is instantiated in the context of DoS protocol exploit attacks against a web server [8]. All the attacks of the dataset are proven to be effective against the server in hand; attacks are executed in face of different defense modules.

We release the data in the form of comma-separated values (`csv`) files. They consist of **labeled network flows**, i.e., large collections of fixed-length records (84 values *per record*, including the label) that summarize bi-directional network traffic features between pairs of endpoints, such as *number* and *length of packets*, *flag counts*, *min*, *max*, *mean*, and *standard deviations* of many attributes. Flows are obtained by running CICFlowMeter[1] on the top of the raw network packets collected in our testbed during the attacks. Overall, the files are ready-to-use and specially crafted for prospective users aiming to apply modern machine learning techniques.

The paper is organized as follows. Section II discusses related work in the area and key novelties of USB-IDS-1. Section III describes the experimental testbed, while Section IV summarizes attack types and defense modules used to elicit the network traffic. Section V provides information on the dataset and descriptive statistics on the traffic and performance measurements. Section VI concludes the work.

[1] https://github.com/ahlashkari/CICFlowMeter

II. Related Work

Nowadays most of the intrusion detection techniques proposed in the literature are tuned and tested by means of **public intrusion datasets**. These are extensively used by researchers and practitioners, as the data they contain are easy to find and available in different formats. For example, they might be accessible in a raw format, such as `pcap` packet data files, or in more "refined" formats, such as network flows organized in comma-separated values files –specially suited to apply modern machine learning techniques– or both. Although every single dataset has peculiar characteristics, all of them have *usability* as a key feature. This makes the evaluation of any detection algorithm straightforward, and justifies the massive use of public security datasets in machine learning research.

Over the years many public security datasets have spread. Some of these have gained strong popularity in the literature, as **KDD-CUP'99** [1]. This can be regarded as the "pioneer" dataset for machine-learning-based intrusion detection. This dataset was collected in 1999 and used as intrusion detection benchmark for the Third International Knowledge Discovery and Data Mining Tools Competition. It includes two weeks of attacks-free instances and five weeks of attack instances that make it suitable for anomaly detection. Numerous intrusion detection techniques have been tested using the KDD-CUP'99 dataset over the last few decades [9], [10] and the number of published studies shows that it has been the *de facto* dataset for this research area. It is worth pointing out that, although KDD-CUP'99 it still largely used [11], [12] and considered a landmark in the intrusion detection field, it has many known drawbacks [13], [14]. Furthermore, after about two decades, it is hardly ever a perfect representative of traffic on present-day networks. This is also true for the more recent NSL-KDD[2] [15], a version of KDD-CUP'99 dataset with duplicates removed and reduced in size.

In recent years, studies that look at security datasets with more critical thinking are spreading. For example, in [16] the Authors analyze the reliability of KDD-CUP'99 by identifying some statistical flaws that might introduce bias when training intrusion detection models. Among the recent public intrusion datasets, the one that gained the greatest popularity in the literature is certainly **CICIDS2017** [4]. It was released by the Canadian Institute for Cybersecurity (CIC) in 2017 and it is publicly available for researchers. Its Authors implemented a testbed framework in order to generate benign and attack data systematically using different profiles. The rapid diffusion of CICIDS2017 is also due to its structure and organization. The dataset offers both ready-to-use labeled flows, for those who want to apply machine learning techniques, and raw `pcap` files. In addition, its Authors provide CICFlowMeter [17], which allows to produce network flows from raw `pcap` files. Another recent public intrusion detection dataset is **UNSW-NB15** [2]. It was created by means of the IXIA PerfectStorm tool[3] in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) and contains both real modern normal activities and synthetic contemporary attack behaviors. The dataset is accessible in comma-separated values file format and in `pcap` raw format. Since the aforementioned datasets are generated in a synthetic environment, they might fail to represent real-life network behaviors. The Authors of the **UGR'16** dataset [18], proposed by the University of Granada, attempt to overcome this limitation. In particular, their dataset is a more "pragmatic" attempt to collect netflow traces representing four months of network traffic from an Internet Service Provider (ISP). UGR'16 includes unidirectional flows, which identify both benign traffic and attacks. Other known public intrusion datasets are **NDSec-2016** [19], **MILCOM2016** [20] and **TRAbID** [21]. They are all available both as network flows and as raw `pcaps` and contain different types of attacks. The interested reader is referred to [5] for a complete survey of existing literature on intrusion detection datasets.

Our contribution. The dataset we propose in this work –different from those mentioned above– also takes into account *"accessory"* parameters of the experimental testbed, such as commodity defense modules. This makes the data collection environment more realistic, since defense modules are typically used, *if not enabled by default*, in real-life server installations. More important, all the attacks have been tested and proven effective against the victim. This is demonstrated by performance measurements (more on this later) carried out with the web server. To the best of our knowledge, there are no similar DoS datasets in the literature that take into account both network traffic and application-level facets.

III. Collection Environment and Procedure

The flows in our dataset have been obtained by collecting traffic data on a network where a victim web server is subjected to DoS attacks. As mentioned in the introduction, traffic data are enriched with service metrics gathered during the progression of the attacks by monitoring the victim. In the following we present the experimental environment and the data collection procedure.

A. Experimental Testbed

Data collection has been performed in a private network infrastructure at the University of Sannio. The experimental testbed consists of three Ubuntu 18.04 LTS nodes, equipped with Intel Xeon E5-2650V2 8 cores (with multithreading) 2.60 GHz CPU and 64 GB RAM within a local area network (LAN). Nodes are presented in Fig. 1.

The **"victim" node** hosts the Apache web server 2.4.29, which was chosen because of its wide use for hosting real-world sites and web apps. Furthermore, the Apache web server is a typical attack target in many public intrusion datasets. The web server supports a variety of modules –including security-related capabilities– that can be purposely enabled/disabled by suitable configuration of the server installation, such as we did during the experiments with the defense modules presented in Section IV-B.

[2]https://www.unb.ca/cic/datasets/nsl.html

[3]https://www.ixiacom.com/products/perfectstorm

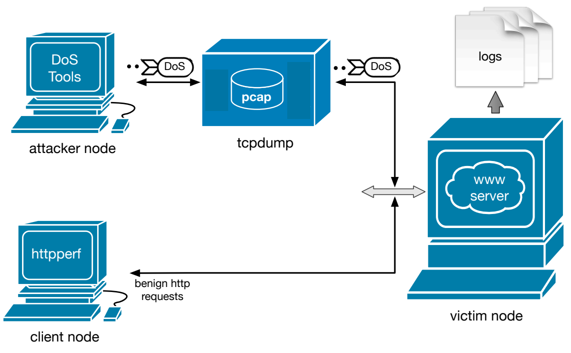


Fig. 1: Experimental testbed.

The **“attacker” node** is intended to generate DoS traffic towards the victim, aiming to disrupt its server operations. Attacks are performed by means of various state-of-the-art tools presented in Section IV-A. The attacker node runs also an instance of `tcpdump`, which is used to capture the traffic between the attacker and the victim in a `pcap` packet data file. It is worth noting that the `pcap` file obtained after a given attack is processed to obtain the network flows.

The **“client” node** hosts `httperf`[4], which is a well-known load generator. It is used here to probe the web server by gathering convenient service metrics that summarize its operational status.

Experiments consist of the following steps:

1) *setup*: boot of `tcpdump` and the web server;
2) *metrics collection*: generation and storage of the service metrics by means of `httperf` during the progression of the attack;
3) *attack*: execution of a DoS attack by means of a dedicated tool; the web server is now exercised with *benign* HTTP requests from `httperf` –referred to as the *load* (L) in the following– and DoS traffic;
4) *experiment completion*: shutdown of the attack tool, `httperf`, `tcpdump` and web server, storage of the `pcap` packet data file, service metrics and event logs for subsequent analysis.

Noteworthy, between pairs of subsequent experiments we clear the logs of the web server (i.e., access and error log), stop the workload generator, attack scripts and the web server, and reboot the nodes to enforce independent experimental conditions. The web server is operated with the *default configuration* –in terms of thread limits and maximum workers– hardened by none or one of the defense modules assessed in this study. By default configuration we mean the one available after a typical installation of the web server (e.g., by means of `apt-get install apache2` pointing to the standard Ubuntu repository[5]). This is done to avoid any potential bias or inadvertent mitigation of the attack caused by changes to the configuration beside the defense modules.

[4]https://github.com/httperf/httperf
[5]http://it.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages

IV. Attacks and Defenses

A. Attack tools

All the attacks contained in the dataset presented in this work were carried out by means of publicly-available DoS scripts and command line utility programs. These are widely used by the security community and enable the execution of different DoS attacks that vary greatly in severity. The attacks and the related information on the corresponding tools are listed in the following:

- **Hulk**: it is conceived as an HTTP flood, which can spawn a large volume of obfuscated and unique traffic. In particular, it generates numerous distinct requests in order to prevent the server defenses from recognizing a pattern and filtering the attack traffic. This makes the requests difficult to be detected by means of signatures. The main goal of the tool is to overwhelm a given web server by starting a load of threads to fire off a flood of HTTP GET requests with randomly generated header and URL parameter values. We use one of the most popular Hulk implementations for our experiments[6].
- **TCPFlood**: it is another well-known DoS attack tool. The attacker sends TCP connection requests locking the available ports on the server and causing incapability to accept legitimate TCP connection requests from other hosts; therefore, it can be considered as a flooding attack. For our experiments we used a GitHub TCPFlood script[7]. It is a `Python` tool that allows to launch a TCPFlood attack against the victim host in seconds.
- **Slowloris**: it is a tool that implements DoS attacks by sending slow HTTP requests (*slow DoS attacks*) against a victim server. This category of attacks uses *low-bandwidth* approaches, which exploit a weakness in the management of TCP fragmentation of the HTTP protocol. We launched this attack by means of a well-known `Python` attack script[8]. In particular, it implements a *slow header* attack by sending incomplete HTTP requests (i.e., without ever ending the header). If the server closes a malicious connection, this is re-established by keeping constant the total number of open connections.
- **Slowhttptest**: it is a versatile tool[9] that allows to launch *slow DoS* application-layer attacks. The tool can prolong HTTP connections in different ways. For our experiments we used Slowhttptest in the “slowloris” mode, which allows to send incomplete HTTP requests to the victim server.

B. Defense modules

Here we take a closer look at the security modules assessed in this study. They are designed by the Apache Software Foundation and can be typically found in real-life installations of the web server:

[6]https://github.com/grafov/hulk
[7]https://github.com/Leeon123/TCP-UDP-Flood
[8]https://github.com/gkbrk/slowloris
[9]https://tools.kali.org/stress-testing/slowhttptest

- **Reqtimeout**: the module `mod_reqtimeout` aims at protecting an HTTP Server from attacks that, as Slowloris, exploit flaws and vulnerabilities of HTTP and TCP protocols related to retransmission timeout and requests transmission data rate. In particular, this module allows to specify –according to the environment and domain where the web server is deployed– timeouts and minimum data rates, which need to be meet in order to keep a connection open. If a connection appears to be too slow or sends only few data *per request*, the server will immediately close the connection.
- **Evasive**: the module `mod_evasive`, instead, is designed to protect a server from those attacks, which try to make a server unavailable by consuming its resources through a huge amount of requests. Among these attacks we can mention DoS Hulk attacks. This module monitors the incoming requests looking for suspicious IPs and related activities, such as multiple requests for the same pages in a short amount of time or multiple requests per second. If one of the aforementioned events is spotted, the server responds with 403 error code and the IP is blacklisted for a certain amount of time.
- **Security2**: the module `mod_security2` is a defense module which acts as a sort of intrusion detection and prevention system (IDPS). Therefore, it is able to identify and respond to a wide variety of attacks, such as SQL and code Injection. Just like a regular signature-based IDPS, it relies on a set of rules, available from free and pay-per-use repositories, related to known attack patterns. These patterns may be used to check different sections of an incoming request, according to the type of attack and to the underlying protocol vulnerabilities they refer to. For instance, a rule may refer to the content of HTTP POST requests related to specific attacks, while another may report the list of malevolent IPs.

The modules can be downloaded and installed through many different package managers (e.g., `apt`, `yum`) or from publicly accessible online repositories. We carefully tuned and tested the correct functioning of the modules according to the instructions from detailed tech blogs and references.

V. Dataset and Measurements

A. Data Organization

For each of the four DoS tools described in Section IV-A, we run four independent experiments. The first experiment consists in running the DoS tool against the web server with **no defense** module in place; the remaining three experiments are done by running the DoS tool after having started the web server by enabling one defense module out of Reqtimeout, Evasive or Security2 (modules are tested *one by one*, i.e., one module per experiment). The duration of each experiment is 600 *s*; the web server is exercised with a client load L = 1000 *reqs/s* by `httperf` during the entire progression of the attack. At the end of each experiment we collect the `pcap` file gathered at the attacker node and service measurements according to the procedure described in Section III-A.

TABLE I: *Total*, *attack* and *benign* network flows by `csv` file.

Name of the `csv` file	**TOTAL**	**ATTACK**	**BENIGN**
Hulk-NoDefense	870485	870156	329
Hulk-Reqtimeout	874382	874039	343
Hulk-Evasive	1478961	770984	707977
Hulk-Security2	1461541	762070	699471
TCPFlood-NoDefense	330543	48189	282354
TCPFlood-Reqtimeout	341483	59102	282381
TCPFlood-Evasive	341493	59113	282380
TCPFlood-Security2	341089	58716	282373
Slowloris-NoDefense	2179	1787	392
Slowloris-Reqtimeout	13610	13191	419
Slowloris-Evasive	2176	1784	392
Slowloris-Security2	2181	1790	391
Slowhttptest-NoDefense	7094	6695	399
Slowhttptest-Reqtimeout	7851	7751	100
Slowhttptest-Evasive	7087	6694	393
Slowhttptest-Security2	7090	6700	390

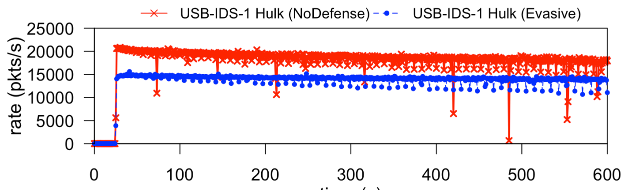
USB-IDS-1 **consists of 16 `csv` files** providing ready-to-use network flows; each file corresponds to a (*DoS tool – defense module*) combination. As previously mentioned, network flows are obtained through CICFlowMeter, which is applied to the `pcap` files obtained after the experiments. The naming scheme of the `csv` files allows to identify the collection scenario. For example, *Hulk-NoDefense.csv* provides the flows obtained by executing Hulk with no defense in place; similarly *Slowloris-Reqtimeout.csv* provides the flows obtained during the experiment where Slowloris is launched against the server hardened with Reqtimeout.

TABLE I provides the cardinality of the files of the dataset and the breakdown attack-benign flows. *A small sample of the dataset is made available for review purposes at a temporary private link*[10]*. The link will be replaced with a full-fledged institutional webpage of the dataset (providing access to all the files) if the paper is accepted.*

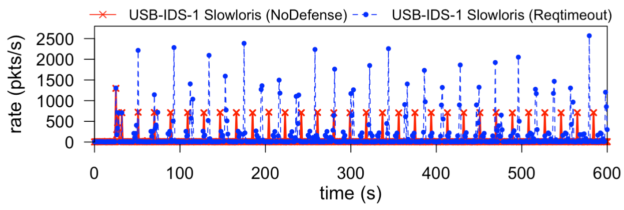
Use cases of USB-IDS-1. At the current stage of development, USB-IDS-1 pertains only to attacks, and does not even try to record benign network traffic profiles over a large network. The benign records available in the dataset consist in either (i) the flows between the victim and the attacker[11] or (ii) a "byproduct" of spurious traffic –intercepted by `tcpdump` at the attacker node– that does not directly involve the victim. There are several *use cases* of USB-IDS-1. For example, our flows can be used to **test** the coverage of machine-learning-based IDS techniques or to verify the **transferability** of an IDS model; moreover, USB-IDS-1 flows can be used to **augment** training data of other related datasets, such as CICIS2017 –based on CICFlowMeter as well– because they reflect novel collection settings accounting for the impact of several defense mechanisms.

[10]https://sannio-box.unisannio.it/index.php/s/e6QF0gpIJi05kop

[11]In this work we follow the convention to label as (i) *attack*, the flows originated by the attacker, and (ii) *benign*, those originated by the victim. The same labelling approach is used by CICIDS2017.

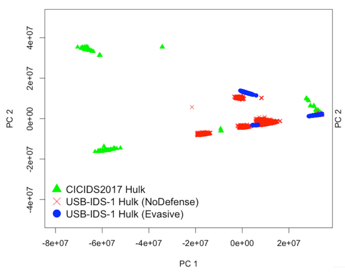


(a) Hulk USB-IDS-1

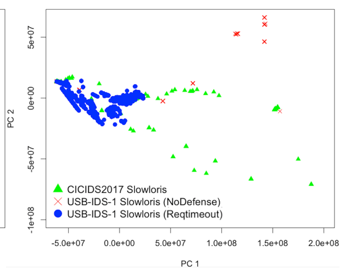


(b) Slowloris USB-IDS-1

Fig. 2: Packet rate measured during the progression of two USB-IDS-1 attacks in case of *no* and *with* defense.

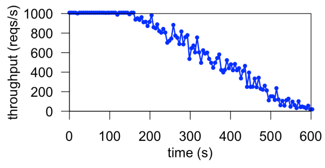


(a) Hulk

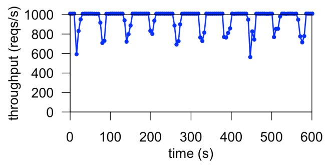


(b) Slowloris

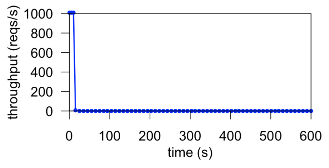
Fig. 3: PCA-based comparison of USB-IDS-1 flows with respect to CICIDS2017 for two attacks.



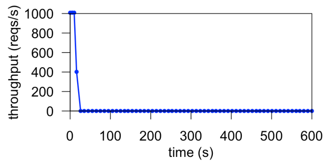
(a) Hulk



(b) TCPFlood



(c) Slowloris



(d) Slowhttptest

Fig. 4: Throughput of the web server during the progression of the attacks (*no defense* experiments).

B. Impact of the Defense

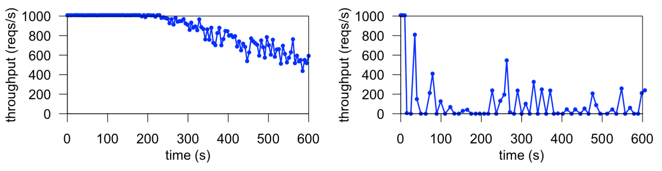
Defense modules strongly alter the network traffic under attack and the related flows. An attempt to learn intrusion detectors on top data obtained under *no defense* may lead to partial –or even incorrect– patterns because the behavior of a given attack depends on the specific defense in hand. One of the key novelties of USB-IDS-1 is the availability of data obtained under *no defense* and a variety of widely-used defense modules.

Fig. 2 provides the rate of the packets (pkts) –measured in *pkts/s*– captured by `tcpdump` at the attacker node during the progression of Hulk and Slowloris in case of *no* and *with defense*, respectively. On average, the rate measured for Hulk in case of *no defense* (×-*marked* series in Fig. 2a) is around 4000 *pkts/s* higher than the rate measured in the experiment with the Evasive module; on the other hand, Reqtimeout causes Slowloris to emit a larger number of packets than the paired *no defense* experiment, as it can be noted by the magnitude of the spikes in Fig. 2b. It is worth noting that different packet distributions impact the number of flows. For example, as shown in TABLE I the number of flows of the file *Hulk-NoDefense.csv* is 870485; this value increases up to 1478961 for *Hulk-Evasive.csv*. Similar considerations hold for other combinations attack/defense.

We conduct a Principal Component Analysis (PCA) to infer a *visual* representation of the network flows. PCA is a dimensionality reduction technique whose objective is to find the directions along which a set of high-dimensional points line up best. To this aim, network flows are regarded as "points" of a Euclidean space: we use a PCA to reduce the dimensionality of the flows, i.e., 83 (label excluded) to 2, for visualization purposes. Fig. 3 shows the 2D scatterplot of the network flows –now represented by means of their coordinates along the first two principal components (PC)– for two attacks of USB-IDS-1. The focus is on Hulk (Fig. 3a) and Slowloris (Fig. 3b) because we aim to provide a comparison with CICIDS2017, which features these two attacks as well. It is interesting to note that USB-IDS-1 flows, shown both in case of *no* and *with* defense (×- and •-*marked* data points, respectively), have *minor* to *none* overlap with CICIDS2017.

C. Performance of the Victim Server

The client node continuously probes the victim by means of `httperf` with a load L = 1000 *reqs/s*. In response to the load, `httperf` generates several service metrics. We provide here the reply rate or **throughput**, i.e., HTTP requests accomplished by the web server within the time unit –measured in *reqs/s*– to show the effectiveness of the attacks available in USB-IDS-1. Fig. 4 shows the throughput of the web server during the progression of the attacks for the *no defense* experiments. In case of attack-free conditions we expect the throughput to be steady at 1000 *reqs/s*; on the contrary, it can be noted that all the attacks significantly impact the throughput. Interestingly, the attacks cause a variety of responses by the victim web server, which range from the progressive absorption of victim resources for Hulk (Fig. 4a) to periodic drops caused by TCPFlood (Fig. 4b) to the typical "on-off" behavior of slowloris attacks (Fig. 4c and 4d), which drop from 1000 to 0 *reqs/s* outright.



(a) Hulk (Evasive) (b) Slowloris (Reqtimeout)

Fig. 5: Throughput of the web server during the progression of two example attacks in case of defense.

As for the impact of the defense on the throughput, it should be noted that the defense modules –in spite of the alteration of the packet rate noted in Section V-B– *do not* guarantee "full" mitigation of the attacks. Fig. 5 shows two instances available in USB-IDS-1 obtained for the pairs (Hulk, Evasive) and (Slowloris, Reqtimeout). In the former case, the module can only delay the throughput depletion of the server when compared to the corresponding *no defense* run in Fig. 4a; in the latter, similarly to Fig. 4c obtained for *no defense*, the throughput remains close to 0 *reqs/s* if not for sporadic spikes.

VI. Conclusion

In this paper, we propose USB-IDS-1, a novel public intrusion detection dataset. It is an example of a *multilayer perspective* dataset, containing both DoS network traffic and application-level facets. Public intrusion detection datasets are of paramount importance to ensure valuable and productive research activities in the network security field. The major problem is the lack of "real-life" data and systematic metrics for assessing and quantify the realism of any public intrusion detection dataset.

Existing public datasets tend to not disclose any useful service metric of the victim at the time the attacks were performed. Overall, they miss the *multilayer perspective*, which allows to establish whether the attacks caused just marginal fluctuations at the network-level or "real damage" to the victim applications. USB-IDS-1 makes a step ahead of other common intrusion detection datasets: all the attacks are proven to be effective against the victim. Another important outcome obtained through our data collection is that defense modules might not provide complete protections from DoS attacks and –in turn– modern IDS techniques should be properly tuned even when the server is hardened. In this respect, USB-IDS-1 provides network flows obtained under various defense modules, which can be used to augment third-party training data or to refine existing IDSs. In the future we will extend the analysis to further attacks and victims, in order to release to the community a complete intrusion detection dataset; moreover, we will address the problem of modeling regular operational profiles for large networks.

References

[1] (1999, Oct) Kdd cup data. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

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

[3] F. Beer and U. Buehler, "Feature selection for flow-based intrusion detection using rough set theory," in *Proc. International Conference on Networking, Sensing and Control*. IEEE, 2017, pp. 617–624.

[4] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani., "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. International Conference on Information Systems Security and Privacy*. SciTePress, 2018, pp. 108–116.

[5] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computer & Security*, vol. 86, pp. 147–167, 2019.

[6] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, p. 4396, 2019.

[7] M. Catillo, A. Pecchia, M. Rak, and U. Villano, "A case study on the representativeness of public DoS network traffic data for cybersecurity research," in *Proc. International Conference on Availability, Reliability and Security*. ACM, 2020, pp. 1–10 Art. no. 6.

[8] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, pp. 1–33, 2017.

[9] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.

[10] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[11] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *Proc. International Symposium on Intelligent Systems and Informatics*. IEEE, 2017, pp. 277–282.

[12] P. Kushwaha, H. Buckchash, and B. Raman, "Anomaly based intrusion detection using filter based feature selection on KDD-CUP 99," in *Proc. TENCON*. IEEE, 2017, pp. 839–844.

[13] 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*, vol. 3, no. 4, pp. 262–294, 2000.

[14] G. Kayacık and N. Zincir-Heywood, "Analysis of three intrusion detection system benchmark datasets using machine learning algorithms," in *Intelligence and Security Informatics*. Springer Berlin Heidelberg, 2005, pp. 362–367.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.

[16] J. V. V. Silva, M. A. Lopez, and D. M. F. Mattos, "Attackers are not stealthy: Statistical analysis of the well-known and infamous KDD network security dataset," in *Proc. Conference on Cloud and Internet of Things*, 2020, pp. 1–8.

[17] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor traffic using time based features," in *Proc. International Conference on Information Systems Security and Privacy*, 2017, pp. 253–262.

[18] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "UGR'16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411 – 424, 2017.

[19] F. Beer, T. Hofer, D. Karimi, and U. Bhler, "A new attack composition for network security," in *10. DFN-Forum Kommunikationstechnologien*. Gesellschaft fr Informatik e.V., 2017, pp. 11–20.

[20] T. Bowen, A. Poylisher, C. Serban, R. Chadha, C. Jason Chiang, and L. M. Marvel, "Enabling reproducible cyber research - Four labeled datasets," in *Proc. Military Communications Conference*. IEEE, 2016, pp. 539–544.

[21] E. K. Viegas, A. O. Santin, and L. S. Oliveira, "Toward a reliable anomaly-based intrusion detection in real-world environments," *Comput. Netw.*, vol. 127, no. C, p. 200216, 2017.