A Critique on the Use of Machine Learning on Public Datasets for Intrusion Detection

Marta Catillo, Andrea Del Vecchio, Antonio Pecchia, and Umberto Villano

Dipartimento di Ingegneria
Università degli Studi del Sannio, Benevento, Italy
{marta.catillo,andrea.delvecchio,antonio.pecchia,villano}@unisannio.it

Abstract. Intrusion detection has become an open challenge in all the latest ICT systems due to an ever-growing urge towards safety in present day networks. Various machine learning-based methods have been already proposed for finding an effective solution to detect and prevent networks intrusions. Many approaches, tuned and tested by means of public datasets, capitalize around well-known classifiers, which often reach detection rates close to 1. However, these results strongly depend on the training data, which are not representative of real production network environments and ever-evolving attacks. This paper is an initial exploration around this problem. After training a classifier on the top of a public intrusion detection dataset, the detector is tested against "held out" data not used for the learning phase. The experiments presented are focused on DoS attacks, and based on the CICIDS2017 dataset and on data collected by emulation of common attacks. Overall, the figures gathered confirm that results obtained in the context of synthetic datasets may not generalize in practice.

Keywords: Denial of Service · machine learning · public intrusion datasets.

1 Introduction

The research community strongly relies on **public intrusion datasets**, such UNSW-NB15 [15], NDSec-1 2016 [3] and CICIDS2017 [17], for designing, evaluating and comparing novel Intrusion Detection Systems (IDS). To this aim, a large number of public datasets have been proposed over the past years [16]. Datasets provide ready-to-use network packets and labeled numeric records – known as *network flows*– collected under normative operations and attack conditions, which makes it straightforward to develop **machine** and **deep learning** models for intrusion detection. Not surprisingly, the intersection of intrusion detection and machine learning is an extremely hyped research topic. A pletora of attack detectors have spread in the literature [12] [6]. Noteworthy, some of these detectors achieve astonishing results. For example, solutions proposed in [9] and [2] achieve an accuracy of 0.999 and 0.996 respectively. At the time being, intrusion detection would seem a perfectly solved problem with no room for further improvements.

Most of the existing –and impressive– intrusion detection results hold just in the context of the datasets that were used to obtain the results themselves. We believe that the results obtained on the top of synthetic and "lab-made" attacks (such as those provided by many public datasets around) cannot be generalized to production networks. Synthetic intrusion datasets simply do not summarize *complexity* and *uncertainty* of **production networks**, which is intertwined with ever-evolving sophistication of the attacks, heterogeneous and non-stationary workloads, configurations and defense mechanisms of real-life servers. In consequence, an attempt to learn intrusion detectors on top of a public dataset may lead to partial –if not incorrect– patterns, which cannot be used to drive general and rigorous security claims on the effectiveness of a given IDS technique. In our opinion, the implications of using public datasets for advancing the state-of-the-practice in intrusion detection and cybersecurity remain quite opaque.

This paper proposes an initial exploration of the proposition above with a focus on the detection of Denial of Service (DoS) attacks. Detection is pursued by capitalizing on network flows, which summarize the conversation between pairs of endpoints, e.g., the attacker and the victim of a DoS attack, through quantitative features, such as *duration*, *number* and *length of packets* and *flag counts*. We leverage benign and DoS network flows both (i) available in CICIDS2017 and (ii) obtained by direct emulation of DoS attacks against a victim server in a controlled testbed. It is worth noting that CICIDS2017 is a public dataset that is gaining increasing attention by the community; as such, it is strongly relevant in the context of our work.

Our critique is based on a twofold experiment. First, we *learn* an intrusion detector on the top of the flows of CICIDS2017, which encompasses benign traffic and various types of DoS attacks and related tools, such as hulk, slowloris and showhttptest. Second, we *test* the detector against "held-out", i.e., not used for learning, benign and DoS flows of CICIDS2017 and those gathered in our testbed after the emulation of slowloris attacks (a specific type of DoS attack available also in CICIDS2017). The effectiveness of the detector is assessed with the consolidated metrics of *accuracy*, *precision*, *recall* and *F1 score*. The detector achieves 0.9991 accuracy when tested with CICIDS2017 data, which is extremely high and consistent with existing literature. Surprisingly, the same detector –instructed to detect a wider class of DoS attacks beside slowloris– performs quite poorly, i.e., 0.821 accuracy, against a slowloris attack conducted in our testbed; even much worse, it achieves only 0.257 detection accuracy for a mitigated variant of slowloris –obtained by hardening the configuration of the victim server through a defense module– being capable of significantly disrupt operations in spite of the defense. Overall, the experiment indicates that results obtained within the "ideal" world of a synthetic dataset may not generalize in practice.

The rest of the paper is organized as follows. Section 2 presents related work in the area. Section 3 describes the experimental testbed and how experiments have been conducted. Section 4 provides an overview of the datasets available in

this study. Section 5 presents the results and lessons learned from our experiment. Section 6 concludes the paper and provide future perspective of our work.

2 Related Work

Nowadays **intrusion detection datasets** have become increasingly pervasive among researchers and practitioners, due to their *usability* and *availability*. In general, data play a key role for the validation of any intrusion detection approach. However, datasets composed of network packets or flows from real-life environments are not easily available due to privacy issues. Therefore in recent years public intrusion detection datasets have been widely used by the security community with the aim of tuning and testing detection algorithms. The majority of these datasets are generated in synthetic environments under normative conditions and different intrusion scenarios. They emulate real network traffic –at least in theory– and they do not contain any confidential data. Most datasets are distributed as labeled network flows, organized in comma-separated values files specially crafted to apply modern machine learning techniques. In particular, each record is a flow and the label states if it is malicious or not. Customarily, data are also distributed as Packet Capture (PCAP) files. These files are an ordered collection of network packets originating from one or more benign or malicious sources. The usability of any intrusion detection dataset reflects its power to provide information necessary for training intrusion detection models efficiently. This is confirmed by numerous literature solutions, which capitalize around these datasets by achieving high level of *accuracy* and *recall*, often close to 1.

The earliest effort to create a public intrusion detection dataset was made by DARPA (Defence Advanced Research Project Agency) in 1999 by providing a comprehensive and realistic intrusion detection benchmarking dataset, named KDD-CUP'99 [1]. It includes two weeks of attacks-free instances and five weeks of attack instances that make it suitable for anomaly detection. Numerous intrusion detection solutions have been tested using the KDD-CUP'99 dataset over the last few decades, such as [20]. Howewer, although this dataset was an essential contribution to the research on intrusion detection, its accuracy and capability to consider real-life conditions have been widely criticized [14] [7]. This is also true for the more recent NSL-KDL [2] [19], a version of KDD-CUP'99 dataset with duplicates removed and reduced in size. A public intrusion detection dataset that has experienced strong popularity among the security world-wide researchers is certainly **CICIDS2017** [3] [17]. Released by the Canadian Institute for Cybersecurity (CIC) in 2017, it simulates real-world network data (PCAPs) and uses the tool CICFlowMeter [10] to extract key statistics on network connections in order to produce labeled flows. Its Authors implemented a testbed framework with the aim to generate benign and attack data systematically using different profiles.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://www.unb.ca/cic/datasets/nsl.html
[3] https://www.unb.ca/cic/datasets/ids-2017.html

Another recent public intrusion detection dataset is **UNSW-NB15**[4] [15], whose synthetic network data were collected by the Australian Center for Cyber Security (ACCS) by means of the IXIA Perfect-Storm tool[5], used as a normal and abnormal traffic generator. It exploits the CVE vulnerability database[6] to create a modern threat environment. The dataset is accessible in comma-separated values file and in PCAP raw format. Since the aforementioned datasets are generated in a synthetic environment, they might fail to represent real-life network behaviors. An attempt to overcome this limitation is suggested by the Authors of the **UGR'16** dataset[7] [13], proposed by the University of Granada. In particular, this dataset is a collection of 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**[8] [3], **MILCOM2016**[9] [4] and **TRAbID**[10] [21]. They are all accessible both as network flows and as raw PCAP and contain different types of attacks. The interested reader is referred to refrence [16] for a complete survey of existing literature on intrusion detection datasets.

It is worth noting that in the last few years works that look more critically at these datasets have spread. In particular, some of them, such as [18], consider the quality of the data by analyzing statistical flaws that might introduce bias in the model training phase. Other papers, such as [5], analyze instead the representativeness of the data contained in public intrusion detection datasets. In reference [8] it is reported a detailed analysis that considers the majority of public intrusion detection datasets issues. In particular, the Authors state that public datasets don't fit real-life conditions, and therefore the value of analysis performed against them may be of questionable value.

Over the years, the usability of public datasets has fostered the spread of machine-learning based intrusion detection systems tuned and tested on such data. Frequently, intrusion detectors are implemented with well-known classifiers, which are able to detect almost all the attacks contained in the dataset used for the training phase. For example, a comparative analysis between different classifiers is reported in [1]. All algorithms are evaluated by means of the CICIDS2017 dataset. In [9], instead, it is reported a feature reduction approach based on the combination of filter-based algorithms, namely InformatiIn paron Gain Ratio (IGR), Correlation (CR), and ReliefF (ReF). The proposed approach aims to reduce the number of features and exploits a rule-based classifier called Projective Adaptive Resonance Theory (PART) in order to detect DoS attacks. The Authors obtain 99.9593% accuracy with the CICIDS2017 dataset. The solution proposed in [22] is Specifically focused on DoS detection; a neural-

[4] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
[5] https://www.ixiacom.com/products/perfectstorm
[6] https://cve.mitre.org
[7] https://nesg.ugr.es/nesg-ugr16/
[8] https://www2.hs-fulda.de/NDSec/NDSec-1/Files/
[9] https://www.netresec.com/?page=ACS_MILCOM_2016
[10] https://secplab.ppgia.pucpr.br/?q=trabid

network based approach relying on the implementation of a simple Multi-Layer Perceptron is compared to the Random Forest technique. Again focused on DoS detection is the paper [11], where well-known machine learning approaches (e.g., Naïve Bayes and Logistic Regression) are used to distinguish normative conditions from malicious ones. In [2], instead, the Authors propose a method that exploits the Bayesian Regularization (BR) backpropagation and Scaled Conjugate Gradient (SCG) descent backpropagation algorithm. The results are promising for the detection of DoS attacks. In particular, the model achieves an accuracy of 99.6% using Bayesian Regularization and of 97.7% in Scaled Conjugate Gradient Descent. It is worth pointing out that all the aforementioned works achieve encouraging results in terms of performance metrics such as *accuracy* and *recall*. However, all of them *blindly* use data and none of them make a *speculative analysis* of the attacks considered during the experiments.

3 Experimental Testbed

The flows that we consider during the experimentation come from both the DoS CICIDS2017 dataset, and from our testbed after the emulation of slowloris attacks against a victim web server. In order to evaluate the progression and the effect of the emulated attack, we collect service metrics by monitoring the victim. In the following we present the experimental environment and the data used for our experiments.

3.1 Experimental Testbed

Our experiments were conducted on a private network infrastructure. 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). The structure of the testbed is sketched in Fig. 1.

The **"victim" node** hosts an installation of Apache web server 2.4.29. This server is a significant case study, due its wide use for hosting real-world sites and web apps. Furthermore, it can fit a wide range of attack targets available in public intrusion datasets. The Apache web server supports a variety of modules –including security-related ones– that can be enabled by adjusting the configuration of the baseline server installation. In particular, for our case study, we have selected `mod_reqtimeout`. This module can mitigate some DoS attacks and is typically enabled by default in the baseline server after installation from the standard Ubuntu repository, which means that its disablement requires explicit changes of the configuration by the user. In particular, it allows to set –according to the environment and domain where the web server is deployed– minimum data rates and timeouts for receiving HTTP request headers and body from clients. These conditions need to be met in order to keep a connection open. If the limits are violated, the connection is dropped and the server sends a `408 REQUEST TIMEOUT` error. We configured the `mod_reqtimeout` according to the instructions

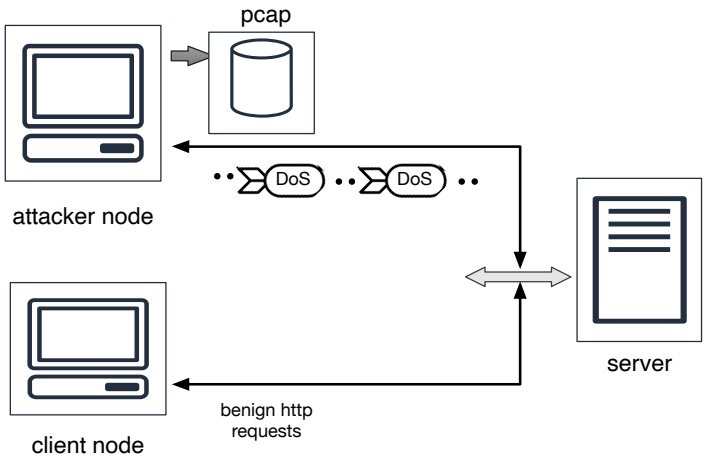


Fig. 1: Experimental testbed.

from the Apache docs[11]. At any time, `mod_reqtimeout` can be seamlessly enabled or disabled by acting on the configuration and re-starting the web server.

The **"attacker" node** generates potentially dangerous DoS traffic against the victim server. Attacks are performed by means of the Slowloris tool (more on this later). 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 successively processed to obtain the network flows.

The **"client" node** hosts `httperf`[12], which is a well-known load generator. This tool makes it possible to set a desired level of workload by setting several parameters. In our testbed, it is here to probe the web server by collecting several convenient metrics that summarize its operational status.

Our experiments are performed according to the following schedule:

1. *setup*: boot of `tcpdump` and the web server;
2. *metrics collection*: start of `httperf`, which exercises the web server with *benign* HTTP requests –referred to as *load* (L) in the following– and collects service metrics during the whole progression of the experiment;
3. *attack*: execution of a DoS attack by means of a dedicated tool; the web server is under both benign load 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. The `pcap` data file is processed to obtain network flows, as previously mentioned.

[11] `https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html`
[12] `https://github.com/httperf/httperf`

It is worth pointing out explicitly that, in order to consider independent experimental conditions, 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. The web server is operated with the *default configuration* –in terms of thread limits and maximum workers– available after a typical installation of the web server (e.g., by means of `apt-get install apache2`, pointing to the standard Ubuntu repository[13]).

4 Datasets

4.1 CICIDS2017

CICIDS2017[14] is a public dataset created in 2017 by the Canadian Institute for Cybersecurity (CIC) [17]. It consists of benign traffic synthesized by the abstract behavior of 25 users, mixed with malicious traffic from most common attacks. In order to create the dataset, a laboratory environment with attacker and victim networks has been set up. The dataset is delivered both as a set of packet capture (`pcap`) files and bidirectional flow labeled format (`csv`). In the latter format, each record is a labeled flow, obtained from the network traffic by means of the tool CICFlowMeter and identified by 85 features. These are mainly network traffic features (source IP, destination IP, source port, destination port, protocol, etc.) along with the label, stating if the flow belongs to normal traffic or to an attack. The data capture period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m., Friday, July 7, 2017, for a total of 5 days. *Monday* is the "normal day" and contains only benign traffic; In the morning and afternoon of Tuesday, Wednesday, Thursday and Friday, in addition to normal traffic, attacks were performed belonging to the categories Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS. DoS attacks, such as `hulk`, `slowloris` and `slowhttptest`, belong to the capture of *"Wednesday"*, i.e., the "DoS day". In particular, the attacker was a Kali Linux node and the victim an Ubuntu 16.04 system with an Apache web server. However, it is not clear whether the Apache `mod_reqtimeout` module was enabled or not at the time the data were collected.

4.2 Slowloris data

All the attacks performed in our testbed were carried out by means of a publicly-available DoS tool: `slowloris`. It accomplishes a DoS attack 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[15]. In particular, this implements

[13] `http://it.archive.ubuntu.com/ubuntubionic-updates/main` amd64 Packages
[14] `https://www.unb.ca/cic/datasets/ids-2017.html`
[15] `https://github.com/gkbrk/slowloris`

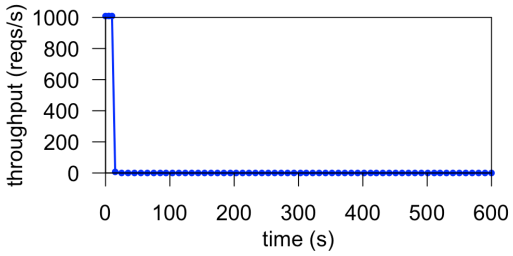


Fig. 2: Throughput of the web server during the progression of the Slowloris attack (*no defense* experiment).

the slow DoS attack by sending incomplete HTTP requests. If the server closes a malicious connection, this is re-established by keeping constant the total number of open connections. We gathered regular network traffic and the attacks `pcap` files as described in Section 3. In order to obtain network labeled flows the `pcap` files obtained from the attack are successively processed by means of the CICFlowMeter [16] tool. The attacks collected from our testbed are obtained in two different operating conditions. In particular:

- **NoDefense-TEST**: data obtained by running the `slowloris` DoS tool against the web server with no defense module in place;
- **Reqtimeout-TEST**: data obtained by running the `slowloris` DoS tool after starting the web server with the Reqtimeout defense module enabled.

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. It is worth pointing out that data collected within our testbed can be considered as *inferred data*, since we also provide information on the effectiveness of the attack. In particular, Fig. 2 and Fig. 3 show the throughput (T), i.e., HTTP requests accomplished by the web server within the time unit measured in reqs/s, during the progression of the attacks. Fig. 2 clearly shows the effectiveness of `slowloris`: the throughput (T) is almost zero for entire duration of the attack. Fig. 3, instead, shows an interesting result. The `slowloris` attack still remains effective despite the Reqtimeout defense module. Except for few spikes, which indicate a recovery of the server, very low values of the throughput (T) are recorded for the entire duration of the attack. Therefore, we can state that even the mitigated version of the attack is effective. It is worth pointing out that, for both cases, in attack-free conditions the throughput of the server is steady at 1000 reqs/s; on the contrary, it can be noted that the attack significantly impact the throughput.

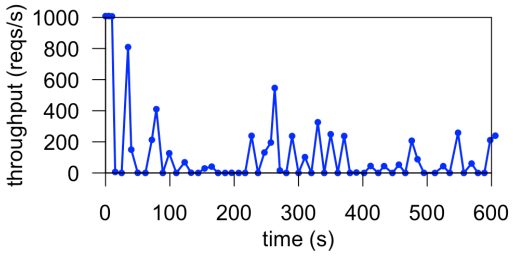


Fig. 3: Throughput of the web server during the progression of the Slowloris attack in case of defense (Reqtimeout experiment).

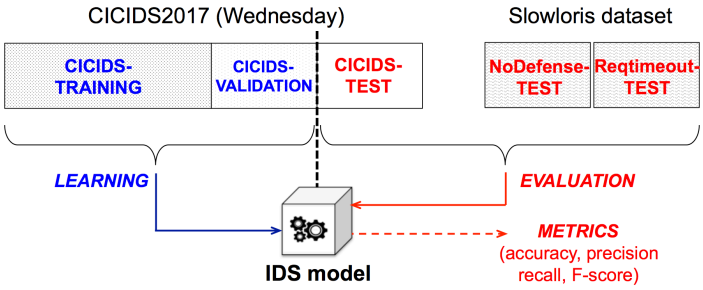


Fig. 4: IDS learning and evaluation framework.

5 Results

5.1 Data Preprocessing and Analysis Framework

As for any machine learning experiment, we preprocess the CICIDS2017 "Wednesday" file and our Slowloris data to make them suitable for the analysis. First, we remove non-relevant or biasing features, i.e., *timestamp* and *id* of the flows, *source address* and *port*, *destination address* and *port*, which leads to total 78 remaining features (label included). Moreover, it is worth noting that the presented experiment refers to a binary classification scenario. As such, flows referring to different types of attacks are considered as belonging to a unique general class named `ATTACK` – encoded with the *0* numeric label; on the other hand, `BENIGN` flows are assigned *1* as label.

Flows contained in the aforementioned CICIDS2017 "Wednesday" file are split into three **disjoint subsets** used for the *training*, *validation* and *test* of the IDS model. While splitting the file, we adopt a stratified sampling strategy with no replacement, which means that (i) the ratio between benign and attack classes of the original file is preserved in the output splits and (ii) each flow of the

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

original file is assigned to a unique split. The original CICIDS2017 file contains 692,703 flows, where 1,297 were discarded due to the presence of malformed or unsuitable values (e.g., "Infinity" or "NaN"). The remaining 691,406 total flows are divided as follows:

- **CICIDS-TRAINING**: *70%* of the total (i.e., 483,982) divided into 307,778 `BENIGN` and 176,204 `ATTACK` flows;
- **CICIDS-VALIDATION**: *15%* of the total (i.e., 103,707), divided into 65,952 `BENIGN` and 37,755 `ATTACK` flows;
- **CICIDS-TEST**: *15%* of the total (i.e., 103,707), divided into 65,952 `BENIGN` and 37,755 `ATTACK` flows.

It should be noted that the three splits above sum up to 691,396, i.e., 10 flows less than the total. Occasionally, the chosen percentages did not return an integer number of flows to be assigned to a given split; in such cases, the number is rounded down to the highest preceding integer.

Fig. 4 provides a representation of the learning and evaluation framework on the top of available datasets. CICIDS-TRAINING and CICIDS-VALIDATION are used to learn the IDS model; CICIDS-TEST jointly with NoDefense-TEST and Reqtimeout-TEST –our Slowloris dataset– represent the test sets used for evaluating the IDS model. CICIDS-TEST, NoDefense-TEST and Reqtimeout-TEST provide "held-out" benign and attack flows, i.e., not seen at all by the IDS model during learning.

Evaluation is based on the typical metrics of *accuracy* (A), *precision* (P), *recall* (R) and *F-score* (F). They are computed from the total number of true negative (TN), true positive (TP), false negative (FN) and false positive (FP) obtained by running the test sets against the IDS model. For instance, a TN is a `BENIGN` flow of the test set that is classified `BENIGN` by the model; a FN is an `ATTACK` flow of the test set that is deemed `BENIGN` by the model. Metrics are computed as follows:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \quad P = \frac{TP}{TP + FP} \quad R = \frac{TP}{TP + FN} \quad F = 2 \cdot \frac{P \cdot R}{P + R} \quad (1)$$

5.2 IDS Learning

Our experiment is based on a very popular machine learning technique, i.e., the **decision tree**. This technique is widely used in the literature to learn IDS models because of its capability to infer explicable rules for classifying network flows. The decision tree consists of predicates, i.e., *nodes* of the tree, which are tested on a given input flow to be classified: based on the outcomes of the tests, decision moves down through the tree until it is reached the class –either `BENIGN` or `ATTACK` in our study– of the flow, i.e., a *leaf* of the tree.

The tree is learned from both `BENIGN` and `ATTACK` flows. Most notably, decision tree is a supervised technique: in consequence, learning needs for the

availability of the labels of the flows. We capitalize on the `python` implementation of the decision tree provided by the package `scikit-learn`[17]. According to Fig. 4, we learn the decision tree with the flows in CICIDS-TRAINING and CICIDS-VALIDATION. Learning is based on the `hypopt`[18] package. More detailed, `hypopt` performs an exhaustive search over desired ranges and combinations of the hyperparameters of the tree, such as the maximum depth of the tree, which is the length of the path from the root to the furthest leaf of the tree, or the minimum number of samples per leaf, i.e. the minimum number of samples that needs to be collected by a leaf during the training phase, in order to be accepted in the final configuration. During the search, `hypopt` (i) trains the decision tree by means of CICIDS-TRAINING, and (ii) tests it with the flows in CICIDS-VALIDATION. The learning stops when `hypopt` finds the optimal combination of hyperparameters. We decided to run the optimizer on a small set of hyperparameter, in order to reduce the search time. We considered the already described maximum depth, minimum number of samples per leaf and random state, a value set for reproducibility purposes, since some of the training steps for the decision tree rely indeed on random splits of the training set.

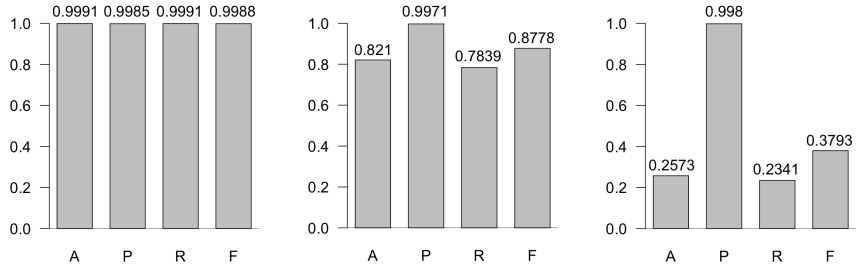
5.3 Evaluation of the Detection Metrics

Firstly we test the decision tree with **CICIDS-TEST**. The obtained metrics are shown in Fig. 5a. The most striking result is that all the metrics are above 0.99, i.e., almost *perfect detection*. In fact, this is the finding achieved by most of the papers on IDSs when machine learning techniques are applied to public intrusion datasets. The values of the metrics represent an "ideal" baseline. Our critique is that such impressive results will likely not hold outside the public dataset itself.

In order to explore this proposition, we test the model against the network flows of slowloris obtained in our testbed. Fig. 5b shows the results for **NoDefense-TEST**. It can be noted that the IDS model, although trained to detect different types of DoS attacks –including slowloris– performs quite poorly. For example, A and R drop from 0.991 (Fig. 5a) to 0.8210 and 0.7839, respectively. This finding is quite surprising because the slowloris attack in NoDefense-TEST is so obvious and proven to be 100% disruptive through all the duration of the data collection, as clearly shown in Fig. 2 in Section 4. The IDS model was purposely trained on a variety of DoS attacks to achieve more flexibility and avoid that the model was "overfitted" only on slowloris data. However, according to the outcome, embedding such attack knowledge in the model did not help to achieve satisfactory results.

As for **Reqtimeout-TEST** in Fig. 5c, the evaluation metrics get even worse. In this case accuracy and recall drop to 0.2573 and 0.2341, respectively. Reqtimeout-TEST hinges on a mitigated variant of slowloris obtained by enabling the `req_timeout` Apache defense module. In principle, the reader may think that an attack done

[17] `https://scikit-learn.org/stable/`
[18] `https://pypi.org/project/hypopt/`



(a) CICIDS-TEST (b) NoDefense-TEST (c) Reqtimeout-TEST

Fig. 5: Evaluation metrics of the IDS model across the test sets.

under defense is harmless and not worthy to be detected; however, Fig. 3 in Section 4 demonstrates that it is not the case for slowloris, which strongly disrupts operations in spite of the defense. Our experiment indicates that a model trained to detect a given attack might be ineffective to reveal a "weaker" variant of the same attack.

5.4 Lessons Learned

There are several interesting lessons learned from our experience. We notice that the "ideal" IDS model obtained in the context of its originating dataset –CICIDS2017 in this study– *does not* generalize to a much more simple proof-of-concept experiment. In consequence, it is hard to see if/how that IDS model would generalize to a real-life production network affected by all the sources of complexity and uncertainty that do not exist in our small-scale, controlled, testbed.

More important, a minor difference with respect to the data gathering environment of the public dataset, such as the enablement of a defense module, can totally invalidate an IDS model inferred on the top of it. It must be noted that defense modules are just a marginal example out of the large number of uncontrollable factors (e.g., sophistication of the attacks, workloads and configuration) that characterize a production network. Public intrusion datasets provide only a limited and incomplete view: our initial experiment demonstrates that *one single variation of the factors changes it all.*

Overall, the implications of using public datasets for advancing the state-of-the-practice of real-life networks and to drive general and rigorous security claims on machine learning and IDS techniques remain quite opaque.

6 Conclusion

Intrusion detection currently arouses great interest as a key part of a defense system. The recent spread of machine learning techniques has boosted significantly the performance of intrusion detection systems. Machine learning models can learn normal and anomalous patterns from training data and generate classifiers that are successively used to detect attacks. Most proposals in the literature exploit public intrusion detection datasets and achieve detection rates that often are very impressive. However, these classifiers are hardly ever employed in real-life networks as they could be ineffective under realistic traffic conditions.

This paper has proposed an initial investigation of the inefficacy of machine learning on public datasets, with a focus on DoS attacks. In particular, we trained an intrusion detector based on the flows of CICIDS2017 dataset, by considering both benign and DoS attack traffic. In order to validate the effectiveness of this detector, we tested it with "held-out" data, i.e., not used for learning. We leveraged benign and DoS network test flows from the CICIDS2017 dataset and from a testbed emulating a `slowloris` attack. The detector exhibits 99% accuracy when tested with data from CICIDS2017. Notably, performance drops against the `slowloris` attack conducted in our testbed. This indicates that the "ideal" conditions that identify most intrusion detection datasets are not generalizable to real-life environments. The finding contributes to establish new knowledge in this area and poses novel open challenges.

Our results could be relevant both for the release of datasets and for the implementation of machine learning algorithms, for the purpose of designing increasingly robust and performing intrusion detection systems. In our future work, we will extend the analysis by emulating other different DoS attacks. More important, our long-term objective is to extend our study to other similar public datasets and machine learning approaches. Our intention is also to analyze *deep learning* techniques as far as their use on real-world data is concerned, since at least in theory they could not be affected by the issues pointed out in this paper.

References

1. Ahmim, A., Maglaras, L., Ferrag, M.A., Derdour, M., Janicke, H.: A novel hierarchical intrusion detection system based on decision tree and rules-based models. In: 15th International Conference on Distributed Computing in Sensor Systems (DCOSS). pp. 228–233 (2019)
2. Ali, O., Cotae, P.: Towards DoS/DDoS attack detection using artificial neural networks. In: 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON). pp. 229–234 (2018)
3. Beer, F., Hofer, T., Karimi, D., Bühler, U.: A new attack composition for network security. In: 10. DFN-Forum Kommunikationstechnologien. pp. 11–20. Gesellschaft für Informatik e.V. (2017)
4. Bowen, T., Poylisher, A., Serban, C., Chadha, R., Jason Chiang, C., Marvel, L.M.: Enabling reproducible cyber research - Four labeled datasets. In: Proc. Military Communications Conference. pp. 539–544. IEEE (2016)

5. Catillo, M., Pecchia, A., Rak, M., Villano, U.: A case study on the representativeness of public DoS network traffic data for cybersecurity research. In: Proc. International Conference on Availability, Reliability and Security. pp. 1–10 Art. no. 6. ACM (2020)
6. Catillo, M., Rak, M., Villano, U.: Discovery of DoS attacks by the ZED-IDS anomaly detector. Journal of High Speed Networks **25**, 349–365 (2019)
7. Kayacık, G., Zincir-Heywood, N.: Analysis of three intrusion detection system benchmark datasets using machine learning algorithms. In: Intelligence and Security Informatics. pp. 362–367. Springer Berlin Heidelberg (2005)
8. Kenyon, A., Deka, L., Elizondo, D.: Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. Comput. & Secur. **99**, 102022 (2020)
9. Kshirsagar, D., Kumar, S.: An efficient feature reduction method for the detection of DoS attack. ICT Express (2021). https://doi.org/10.1016/j.icte.2020.12.006
10. Lashkari, A.H., Gil, G.D., Mamun, M.S.I., Ghorbani, A.A.: Characterization of Tor traffic using time based features. In: Proc. International Conference on Information Systems Security and Privacy. pp. 253–262 (2017)
11. Lee, J., Kim, J., Kim, I., Han, K.: Cyber threat detection based on artificial neural networks using event profiles. IEEE Access **7**, 165607–165626 (2019)
12. Liu, H., Lang, B.: Machine learning and deep learning methods for intrusion detection systems: A survey. Applied Sciences **9**(20), 4396 (2019)
13. Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Therón, R.: UGR'16: A new dataset for the evaluation of cyclostationarity-based network idss. Comput. & Secur. **73**, 411 – 424 (2017)
14. McHugh, J.: 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)
15. Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: Military Communications and Information Systems Conference. pp. 1–6. IEEE (2015)
16. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Comput. & Secur. **86**, 147–167 (2019)
17. Sharafaldin, I., Lashkari, A.H., Ghorbani., A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: International Conference on Information Systems Security and Privacy. pp. 108–116. SciTePress (2018)
18. Silva, J.V.V., Lopez, M.A., Mattos, D.M.F.: 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. pp. 1–8 (2020)
19. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: Symposium on Computational Intelligence for Security and Defense Applications. pp. 1–6. IEEE (2009)
20. Tavallaee, M., Stakhanova, N., Ghorbani, A.A.: Toward credible evaluation of anomaly-based intrusion-detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **40**(5), 516–524 (2010)
21. Viegas, E.K., Santin, A.O., Oliveira, L.S.: Toward a reliable anomaly-based intrusion detection in real-world environments. Comput. Netw. **127**(C), 200–216 (2017)
22. Wankhede, S., Kshirsagar, D.: DoS attack detection using machine learning and neural network. In: Fourth International Conference on Computing Communication Control and Automation (ICCUBEA). pp. 1–5 (2018)