Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 191 (2021) 192–199

Procedia Computer Science

www.elsevier.com/locate/procedia

The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC)
August 9-12, 2021, Leuven, Belgium

On the analysis of open source datasets: validating IDS implementation for well-known and zero day attack detection

Benedetto Marco Serinelli[a,*], Anastasija Collen[a], Niels Alexander Nijdam[a]

[a]*Centre Universitaire d'Informatique, University of Geneva, Route de Drize 7, CH-1227 Carouge, Switzerland*

Abstract

This paper presents the implementation of an anomaly-based Intrusion Detection System (IDS), capable to detect well-known and zero-day attacks. First, we extend our previous work by generating the Machine Learning (ML) predictors based on KDD99, NSL-KDD and CIC-IDS2018 datasets, and providing the programming language evaluation and the final validation platform. We have built IDS detection solution in two phases. The first *Training* phase explores available datasets to generate the predictors. The second phase is composed of two processes. *Extraction* generates the statistical network traffic metrics from the PCAP files and processes them into commma separated values (CSV) files. The *Prediction* loads predictors in main memory and feeds them with CSV files to predict the well-known and zero-day attacks. The aforementioned initial datasets contain the statistical network traffic metrics of the well-known attacks, collected at runtime execution of the malicious software. Zero day attacks can generate a statistical network traffic metrics similar to well-known attacks. Therefore, to showcase the zero-day anomaly detection, we realise a validation platform. Six attacks (three Denial of Service (DoS) and three scanning), not recorded in the initial datasets, are executed in an isolated environment. The achieved result indicates a misclassification prediction error that inhibits the application of the automatic attack responses, although the misclassification errors were minimised, during the *Training* phase.

Keywords: intrusion detection system; anomaly-based detection; machine learning; KDD99; NSL-KDD; CIC-IDS2018; near real time anomaly detection; well-known attack detection; zero-day attack detection; open source datasets; anomaly detection validation platform

1. Introduction

The academic research shows an ever increasing interest in the development of Intrusion Detection Systems (IDSs) designed upon Machine Learning (ML) approaches [1, 2, 3]. We present an *anomaly-based* IDS, able to detect Denial of Service (DoS), Probe, Distributed Denial of Service (DDoS), SSH Brute Force (SBF), Brute Force Web (BFW), Brute Force XSS (BFX), SQL Injection (SQLI), FTP Brute Force (FBF) and Infiltration (INF) attacks in near-to-real-

* Corresponding author.
E-mail address: benedetto.serinelli@unige.ch

1877-0509
10.1016/j.procs.2021.07.024

time. It employs the use of ML methodologies to classify the statistical network traffic metrics for detecting the well-known and zero-day attacks. We have started with the selection of three ML models (Support Vector Machine (SVM), Random Forest (RF), XGBoost (XGB)), which were trained on three open source datasets (KDD99, NSL-KDD and CIC-IDS2018) with the purpose to create the distinct predictors that can detect the network attacks. Thereafter, the attack detection execution phase is conducted in two processes: *Extraction* and *Prediction*. The *Extraction* distils the most relevant statistical network metrics, such as the total number of exchanged packets and the mean of flow duration, from PCAP files and stores them in commma separated values (CSV) file format. The *Prediction* process loads the trained predictors, monitors the file system for new CSV files from the *Extraction* and performs the anomaly pattern detection and classification.

Guided by the above mentioned setup, this paper builds on top of our previous work [4] with the following advancements:

- Analysis of selected public and open source datasets for future comparison studies;
- Evaluation of the programming language in terms of Python and R prediction performance;
- Development of tools for statistical network traffic metrics generation and anomaly detection;
- Realisation of the validation platform to showcase the zero-day attack detection.

The Related work is presented in Section 2. In Section 3, we discuss the *Training* phase, followed by results on the optimal IDS prediction metrics in Section 4. Our IDS architecture is presented in Section 5, describing the *Extraction* and the *Prediction* processes. In Section 6, we present the execution and detection of six zero-day attacks on the validation platform. Finally, the paper is concluded in Section 7.

2. Related work

Recent research identified the knowledge gap in the selection process of the training datasets and the data analysis methodologies, as the basis to build effective IDSs [5]. Despite the availability of a large number of public and open source datasets, such as DARPA, KDD99, DEFCON, CAIDA, LBNL, CDX, Kyoto, Twente, UMASS, ISCX2012, ADFA and NSL-KDD, these contain many flaws and are not sufficient for anomaly-based IDS [1, 6]. Nevertheless, the KDD99 and NSL-KDD remain very popular and are frequently used to benchmark different IDS solutions. Furthermore, the recent dataset, CIC-IDS2018[1], attempts to solve several identified problems, such as the lack of the statistical network traffic analysis metrics, the description of the network behaviour and the evolution of intrusions features [6]. In addition, it incorporates examples of new kinds of malicious activities, such as Heartbleed, Hulk-DoS tools and an embedded backdoor (assembled through the Metasploit framework), for vulnerable application's exploitation.

Previously referenced works do not report detailed data analysis methodologies and the used programming language specifications, hindering to replicate the results and making it difficult to compare between various implementations and achieved results. Although H. Hindy et al. [7] provide details about the NSL-KDD dataset and their results, they do not provide an accurate detection metrics analysis, such as the use of testing time, False Alarm Rate (FAR) and Undetected Attack (UA). In relation to the CIC-IDS2018 dataset, Gamage and Samarabandu [5] do not provide an accurate exploratory and cleaning dataset description.

Our work aims to provide an accurate description of datasets cleaning, training and anomaly prediction processes, while targeting to realise an anomaly-based IDS.

3. Training phase

Prior to training, the datasets cleansing operations should be performed, as outlined in the Fig. 1. The ultimate goal of this process is to improve the predictors' classification performance and to eliminate introduction of the classification bias.

[1] https://www.unb.ca/cic/datasets/ids-2018.html

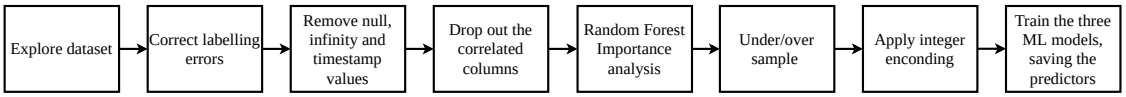

Fig. 1. Dataset cleansing operations

Each of the following paragraphs describes the datasets used for creating the individual predictors together with analysis on necessary dataset optimisations and their recorded attacks. We chose two classification problems appearing to be the most relevant to the detection problem: multi-class and binary classification. The multi-class classification problem is more suitable to datasets that recorded more than one type of attack, while the binary-class trains on the datasets containing only one type of attack. The integer encoding operation was applied for both detection problems to map the type of attacks, reducing the classification errors and predictors complexity. From exploring the selected datasets, we know that they are unbalanced. Thus, under/over-sampling operations are mandatory before training the predictors. On a side note, a human network operator would need to identify for each attack class the correct prediction metrics in the real use case.

The predictors trained on KDD99 and NSL-KDD are built to detect the DoS, Probe, User to Root (U2R) and Root to Local (R2L) well-known attacks [4]. The attack examples are under sampled: 7000 examples for the benign examples, rounding to 5000 the DoS examples. The Probe, U2R and R2L examples are not under/over-sampled [4].

The CIC-IDS2018 is a collection of the following sub-datasets: Wednesday-14-02-2018, Thursday-15-02-2018, Friday-16-02-2018, Wednesday-21-02-2018, Thursday-22-02-2018, Friday-23-02-2018, Wednesday-28-02-2018 and Thursday-01-03-2018.

The predictors, trained on Wednesday-14-02-2018 dataset, are designed upon FBF and SBF network traffic. The statistical network traffic metrics were recorded while launching the Patator[2] for both attacks through a large password dictionary (about 90 million words). The benign examples are under-sampled to the same length of FBF class examples. The SBF class is over-sampled by the same size of FBF class.

The predictors, trained on Thursday-15-02-2018 dataset, are built on the DoS attacks detection, running DoS-GoldenEye[3] and DoS-Slowloris tools. The first tool is an Hypertext Transfer Protocol (HTTP) DoS test that exploits the HTTP Keep Alive and No-Cache attack vector. The second tool is a Perl based DoS tool for incomplete TCP connection, preventing the its closing. The dataset is under-sampled, using the 7% of benign examples.

The predictors trained on Friday-16-02-2018 dataset, containing the statistical network metrics of the DoS-Slow-HTTPTest[4] and DoS-Hulk tools, can detect DoS attacks. The first tool prolongs the HTTP connections simulating the Application Layer DoS attacks. The second one generates unique and obfuscated traffic volumes to make unavailable web servers or web-service running on remote host. We have found through data exploratory analysis, that the row number 1 000 000 is wrongly labelled and we removed it before the training. Although the difference between benign and DoS attacks is about 2000 rows, the examples are balanced and we do not over/under-sample them.

The Wednesday-21-02-2018 dataset generates the predictors for DDoS binary classification. DDoS-LOIC-UDP[5] and DDoS-HOIC tools are used to launch Web-based DDoS attack. The first tool is a set of C# scripts for network stress testing and DoS attack application, flooding the target with TCP, UDP, or HTTP packets. The second tool, written in Visual Basic and C#, can attack 256 URLs simultaneously and is designed to overcome the DDoS-LOIC-UDP users limits. The benign examples are under-sampled to 30% of original examples.

The predictors trained on Thursday-22-02-2018 and Friday-23-02-2018 datasets are capable to detect the BFW, BFX and SQLI well-known attacks. The brute force attacks attempt to break the account (username and password) using a dictionary. The Friday-23-02-2018 and Thursday-22-02-2018 are under-sampled with a threshold of 1% of original benign examples.

The Wednesday-28-02-2018 and Thursday-01-03-2018 datasets build the INF statistical network traffic attack predictors. The INF attacks are performed by sending an email with malicious file to exploit vulnerability. The malicious

[2] https://github.com/lanjelot/patator
[3] https://github.com/jseidl/GoldenEye
[4] https://github.com/shekyan/slowhttptest
[5] https://github.com/NewEraCracker/LOIC

file is created through Metasploit's backdoor to launch a port scanning process after the exploitation. Thus, we consider the INF attack as a Probe attack, due to the port scanning attacks execution after machine exploitation. The Thursday-01-03-2018, after data exploratory process, is wrongly labelled in the row number 414, where the label is "Label", which is removed before to train the models. In addition, the latter dataset is under-sampled, using the 37% of original benign examples to balance the training examples.

4. Predictor performance evaluation

We extended our previous work with a predictor performance evaluation and use of the additional dataset CIC-IDS2018, conducted on a common laptop with 16 GB of RAM and Intel i7-10510U CPU. We compared between R and Python implementations, with findings presented in Tab 1. The results are sorted by accuracy to allow the comparison between this work and others. In general, the prediction time of Python models is faster than R models. To ensure the best IDS detection performance, we only present the predictors with the fastest prediction time and lowest FAR and UA values. Finally, due to the lack of published details of analysed works, we could only provide a non-granular view on the CIC-IDS2018 dataset comparative analysis, while we operate on sub-dataset level in our implementation. In the Tab. 2 and the Tab. 3, we depict the best performance predictors for each dataset, comparing

Table 1. Comparison of the ML models' performance. PL = programming language, ACC = accuracy, PT = prediction time and REF = references

Dataset	Model	PL	ACC (%)	PT (s)	REF
Wednesday-21-02-2018	XGB	Python	99.999	0.150	This work
Wednesday-14-02-2018	XGB	Python	99.998	0.033	This work
Friday-16-02-2018	XGB	Python	99.997	0.042	This work
Thursday-15-02-2018	XGB	Python	99.992	0.022	This work
Thursday-01-03-2018	XGB	Python	99.983	0.039	This work
NSL-KDD	RF	Weka	99.600	N.A.	[8]
KDD99	SVM	Scala	98.900	1.370	[9]
Friday-23-02-2018	XGB	Python	98.513	0.010	This work
CIC-IDS2018	RF	Python	98.340	2.550	[5]
KDD99	XGB	R	98.321	0.240	[4]
Wednesday-28-02-2018	RF	Python	98.270	0.031	This work
NSL-KDD	RF	R	97.930	7.820	[4]
KDD99	XGB	Python	95.919	0.119	This work
NSL-KDD	SVM	N.A.	94.440	N.A.	[7]
KDD99	RF	Python	92.340	7.700	[5]

the FAR, UA and prediction time values. In addition, we also propose the models for the same datasets that reach an accuracy between [98%-99%] to extend a future comparison with this work and the future research works.

The FAR and UA metrics demonstrate the predictors' resilience against the false detection and capability to detect a large number of attack occurrences. Specifically, the FAR means that the network traffic is incorrectly detected as anomaly, increasing the network operator alarm fatigue and the number of wasteful attack responses when used in real deployment environment. The UA describes the percentage of malicious network profiles that are not detected as attacks. The execution of different predictors with different level of FAR and UA can reduce the general detection errors, if all predictors are combined. The human operator can compare and evaluate the predictor outcomes, merging different FAR and UA levels. Conversely, the fast prediction time is suitable for the near-to-real-time anomaly prediction.

Although we previously described the knowledge gaps, we cannot compare our work and the other works due to the lack of the accurate descriptions of used datasets and training methods. For instance, we have identified two kinds of labelling errors in the CIC-IDS2018 dataset. The first labelling error is the column "CWE", which is not a proper TCP flag and is present in all CIC-IDS2018 sub-datasets. The correct TCP flag is Congestion Window Reduced (CWR), which has to be manually renamed. The second labelling error appears in the rows of Thursday-01-03-2018 (row 414) and Friday-16-02-2018 (row 1 000 000) CIC-IDS2018 which have to be removed or corrected prior to the training.

Table 2. Comparison of results for multi-class classification

Model	Acc(%)	FAR(%)	UA(%)	PT (s)
Dataset: KDD99				
XGB	95.919	DoS:0.010 Probe:0.830 U2R:0.031 R2L:0.033	7.671	0.119
Dataset: NSL-KDD				
RF	93.998	DoS:0.010 Probe:3.684 U2R:0.010 R2L:0.042	5.671	0.227
Dataset: Wednesday-14-02-2018				
RF	99.997	FBF:0.001 SBF:0.003	0.001	0.155
XGB	99.998	FBF:0.001 SBF:0.003	0.001	0.033
Dataset: Thursday-22-02-2018				
SVM	98.058	BFW:1.854 BFX:2.422 SQLI:0.286	36.667	0.061
RF	99.723	BFW:0.006 BFX:1.932 SQLI:0.191	0.001	0.028
XGB	98.146	BFW:0.063 BFX:2.226 SQLI:0.340	0.001	0.020
Dataset: Friday-23-02-2018				
XGB	98.513	BFW:0.630 BFX:3.215 SQLI:0.308	8.571	0.010

Table 3. Comparison of results for binary classification

Model	Acc(%)	FAR(%)	UA(%)	PT (s)
Dataset: Thursday-15-02-2018				
RF	99.997	DoS:0.001	0.007	0.157
XGB	99.992	DoS:0.002	0.019	0.022
Dataset: Friday-16-02-2018				
RF	99.996	DoS:0.003	0.002	1.323
XGB	99.997	DoS:0.005	0.002	0.042
Dataset: Wednesday-21-02-2018				
XGB	99.999	DDoS:0.345	0.852	0.150
Dataset: Wednesday-28-02-2018				
RF	98.270	INF:38.255	30.180	0.030
Dataset: Thursday-01-03-2018				
XGB	99.983	INF:0.002	0.001	0.039

We already reported on the issue of using accuracy as a prediction metric in [4]. Accuracy as prediction metric can only portray the overall predictors performance and cannot function as an insightful global IDS metrics. Conversely,

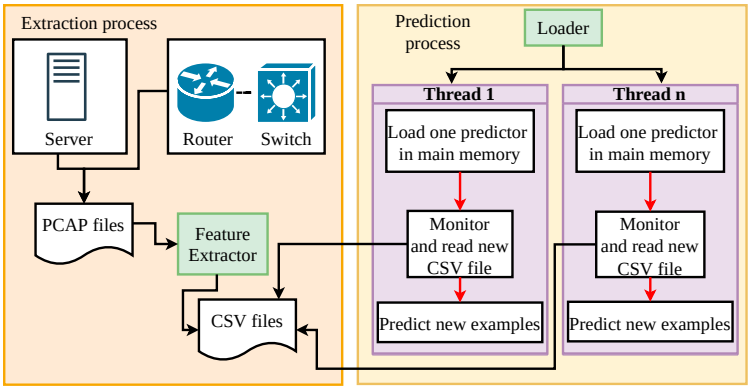

Fig. 2. The tools overview of proposed IDS detection solution

a human operators wants to know how much, for each predicted class, the predictors wrongly classified, FAR, or not-classified, UA, the examples to execute the attack responses and to check the network Quality of Service in case it is or not under-attack. The prediction time is strictly linked to the hardware machine and other works hinder the experiment replication, due to the missing of clear dataset cleaning strategy description. We can only compare this work with our previous work [4], executed on the same machine. The Python prediction time is faster than R at parity of FAR and UA.

5. IDS architecture

The Fig. 2 shows the proposed IDS detection solution, highlighting the developed tools. The proposed IDS detection solution starts from the *Extraction* process, responsible for the processing of the statistical network traffic metrics from PCAP files. It mainly focuses on the *feature extractor* tool implementation. The PCAP files can be captured from the hosts, routers and switches, exploiting Shared Ethernet, Switched Ethernet, Ethernet hub, monitoring port, machine in the middle and switch tap configurations. *Feature extractor* reads all PCAP files from a source directory by aggregating all captured packets information and computing such metrics as forward flow duration, standard deviation of backward flow and the counting of SYN flag for all packets. Finally, it dumps the computed network traffic metrics into the CSV files. The *feature extractor* is designed upon the CICFlowMeter tool [6], adapting the latter open source tool for our scope. Furthermore, we addressed known dataset labelling error, requiring changing the CWE flag to the CWR [6].

The *Prediction* process of our architecture aims to develop the *loader* and is the core of the IDS. The *loader* tool dynamically allocates a new thread to read the predictors objects' representation via Pickle library. Hence, the tool loads in main memory each predictor in a separated thread to call the *prediction functions*. The latter function takes as an input the CSV files, and so statistical network traffic metrics, to predict the attacks or benign traffic examples. The *loader* monitors and reads the CSV files, generated by the *feature extractor*. The *prediction functions* can predict the new statistical network traffic metrics as a new data example, detecting a benign or anomaly patterns.

The memory analysis of the *feature extractor* and *loader* was performed on the same training machine. The *feature extractor* allocates up to 700 MB of RAM. Conversely, 1.5GB of RAM was reserved by the *loader*, opening about 1500 file descriptors with a startup peak of 5000.

6. Validation

We create the validation platform, where malicious software, not used in the original datasets, was launched with the purpose to capture generated network traffic. The use of previously non recorded malicious software allows ver-

[6] https://github.com/ahlashkari/CICFlowMeter/issues/81

ification of the zero-day attacks detection, due to the lack of their statistical network traffic metrics recorded in the original datasets. Six attacks are performed in Virtual-Box environment and recorded the traffic network via Wireshark (host network capturing). We validate the proposed IDS solution on the same performance evaluation machine through a *detection window* of one minute. The *detection window* was chosen to compute how many predictors the IDS is able to allocate in memory and to predict a non-fixed input example size. It is noteworthy that *detection time* differs from *prediction time*. The first is computed via a fixed time to execute the IDS detection pipeline to carry out both *Extraction* and *Prediction* processes, using an undetermined input example size. Conversely, the *prediction time* is the time needed by the predictors to classify a fixed example size. Thus, we chose a reasonable *windows time* to generate a new data samples for all six zero-day attacks and classify the new examples. The result gives us a time performance metric to compare our IDS solution in real use case when the human operator should merge different FAR and UA predictors' output levels.

Three DoS attacks (*hping3 –flood -F -q -d*, *hping3 –flood -S -q -d* and *hping3 –flood -R -P -A -U -q -d*) were executed via **hping3** tool, aiming to deny the victim machine service via FIN (-F), SYN (-S), RST (-R), PUSH (-P), ACK (-A) and URG (-U) flags packet flooding and a packet body size of 15 000 bytes (+ 40 bytes of header). The **nmap** tool was employed to execute a scanning attack via TCP SYN scans (-sS), OS detection (-O), probing through the open port (-sV), network surveys generating 100 000 000 hosts (-iR) and skipping the host discovery (-Pn) parameters (*nmap -sS -O*, *nmap -sV* and *nmap -iR 100000000 -Pn*).

The Tab. 4 depicts the detection results in terms of predictors classification output. We map the attack ID and the predictors outputs for analysing the misclassification errors. Thus, the attack IDs #1-#2-#3 are DoS attacks and the predictors should predict them as DoS or DDoS are classified, labelling as correct detected (CD). Conversely, the attack IDs #4-#5-#6 are scanning attacks and the predictors should classify them as Probe or INF, labelling as CD. As depicted in Tab. 4, the validation platform demonstrates a misclassification error of the attack classes, due to the

Table 4. Validation platform results and attacks mapping

Dataset	SVM	RF	XGB
Friday-16-02-2018		#3:CD, #4-#5-#6:DoS	#1-#2-#3:CD, #4-#5-#6:DoS
Friday-23-02-2018			#4-#5-#6:BFW
KDD			#1-#2-#3-#5-#6:CD, #4:DoS
NSL-KDD			#1-#2-#3-#5-#6:CD, #4:DoS
Thursday-01-03-2018	#5:DoS, #6:CD	#1-#2-#3:INF,#5-#6:CD	#1-#2-#3:INF, #4-#6:CD
Thursday-15-02-2018	#4-#6:DoS		#1:CD, #4:DoS
Wednesday-14-02-2018	#4-#5:SBF		#1-#2-#6:FBF, #4-#5:SBF
Wednesday-21-02-2018	#6:DoS	#1:CD, #4-#6:INF	#1-#2-#3:CD, #4-#6:DoS
Wednesday-28-02-2018	#5:CD	#2:INF, #4-#5-#6:CD	#1-#2-#3:INF, #4.#5-#6:CD

network traffic similarity between the attack classes. For example, the six attacks are classified as FBF or SBF or BFW, although DoS and Probe were performed and test infrastructure does not expose the File Transfer Protocol (FTP), Secure Shell (SSH) and web application services. However, the FBF, SBF and BFW generate the same traffic of DoS, DDoS, Probe and INF attacks, where the adversary launches repetitive FTP, SSH and HTTP connections to guess the credentials, via password dictionary, opening many TCP connections. We noted that FBF, SBF and BFW traffic are closer to DoS, DDoS, Probe and INF, comparing the TCP traffic. Diving deeply into analysis of the traffic metrics, the DoS and DDoS attacks overflow the victim machine with a huge quantity of TCP packets like the FBF, SBF and BFW for breaking the Application Layer services related to the user accounts. Conversely, the Probe and and INF attacks explore the host alive via TCP packets, such as FBF, SBF and BFW attacks guess to break the accounts via a brute force dictionary, sending a TCP packets. Thus, the DoS, DDoS, Probe, INF, FBF, SBF and BFW generate the same traffic at the Transport level (TCP packets). While, the FBF, SBF and BFW differ at Application Layer (FTP, SSH and HTTP) to DoS, DDoS, Probe and INF, where dictionary username and password are included in the Application Layer packet.

In addition, the validation platform depicts the misclassification error between the DoS, DDoS, Probe, INF attack classes. The Probe and INF attacks send contemporary and parallel multiple TCP and UDP packets to check if a host is alive, scanning all port numbers, as a DoS or DDoS attacks do. As can be noted, some cells are empty due to the

non availability of the prediction results. It is caused by the *detection window* for testing the predictors to classify the statistical network traffic metrics in less than one minute. The validation platform shows that the network human operator should confirm the attack response triggering, avoiding to enable automatic responses to address up the misclassification errors. Thus, the attack response decision making should be a semi-automated tool through human operators validations.

7. Conclusions

This work provides a detailed analysis on three datasets, the CIC-IDS2018, which is an improvement over other open source datasets, and the old but very popular KDD99 and NSL-KDD. A performance evaluation on several models showed that the Python provided implementation as superior. The presented architectural approach validates the IDS implementation by employing the ML methodologies. The proposed IDS solution extends our previous work, by training and comparing the models and realising the *Extraction* and *Prediction* processes. The Python predictors reach the best performance in terms of prediction time, which ensures the near-to-real-time detection of anomalies. From the performed analysis it is evident that achieving reproduction of reported results is a difficult task. Furthermore, our data analysis enables the reader to replicate the presented results and encourage future comparison.

We consider that observed misclassification error on attack classes prediction enables our IDS to detect the zero-day attacks, if the aforementioned attacks generate the network profile similar to the well-known attacks. In addition, we note a similar Transport Layer network traffic between DoS, DDoS, Probe, INF, FBF, SBF and BFW, highlighting a misclassification errors.

Lastly, the validation platform show-cased through simulation a realistic anomaly-based IDS deployment scenario, where an attacker executes DoS and Probe attacks against a victim machine, capturing the network traffic from the host machine via a Wireshark.

Acknowledgements

This work has received funding by the European Union's Horizon 2020 Research and Innovation Programme through AVENUE project (https://h2020-avenue.eu/) under grant agreement No 769033 and nIoVe project (https://www.niove.eu/) under grant agreement No 833742.

References

[1] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Survey of intrusion detection systems: techniques, datasets and challenges, Cybersecurity 2 (1) (2019). doi:10.1186/s42400-019-0038-7.

[2] W. Wu, R. Li, G. Xie, J. An, Y. Bai, J. Zhou, K. Li, A survey of intrusion detection for in-vehicle networks, IEEE Transactions on Intelligent Transportation Systems 21 (3) (2020) 919–933. doi:10.1109/TITS.2019.2908074.

[3] B. M. Coronado, U. Mori, A. Mendiburu, J. Miguel-Alonso, Survey of Network Intrusion Detection Methods from the Perspective of the Knowledge Discovery in Databases Process, IEEE Transactions on Network and Service Management 4537 (c) (2020) 1–1. doi:10.1109/tnsm.2020.3016246.

[4] B. M. Serinelli, A. Collen, N. A. Nijdam, Training Guidance with KDD Cup 1999 and NSL-KDD Data Sets of ANIDINR: Anomaly-Based Network Intrusion Detection System, Procedia Computer Science 175 (2019) (2020) 560–565. doi:10.1016/j.procs.2020.07.080. URL https://doi.org/10.1016/j.procs.2020.07.080

[5] S. Gamage, J. Samarabandu, Deep learning methods in network intrusion detection: A survey and an objective comparison, Journal of Network and Computer Applications 169 (July) (2020) 102767. doi:10.1016/j.jnca.2020.102767. URL https://doi.org/10.1016/j.jnca.2020.102767

[6] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy 2018-Janua (Cic) (2018) 108–116. doi:10.5220/0006639801080116.

[7] H. Hindy, R. Atkinson, C. Tachtatzis, J. N. Colin, E. Bayne, X. Bellekens, Utilising deep learning techniques for effective zero-day attack detection, Electronics (Switzerland) 9 (10) (2020) 1–16. doi:10.3390/electronics9101684.

[8] J. Zala, A. Panchal, A. Thakkar, B. Prajapati, P. Puvar, Intrusion Detection System using Machine Learning, International Journal of Scientific Research in Computer Science, Engineering and Information Technology 42 (3) (2020) 61–71. doi:10.32628/cseit2062166.

[9] L. Mohammadpour, T. C. Ling, C. S. Liew, C. Y. Chong, A Convolutional Neural Network for Network Intrusion Detection System, Proceedings of the Asia-Pacific Advanced Network 46 (0) (2018) 50–55.