A Privacy-Preserving Approach for Vulnerability Scanning Detection

A Privacy-Preserving Approach for Vulnerability Scanning Detection

Leonardo Regano[1,*,†], Daniele Canavese[2,†] and Luca Mannella[3,†]

[1]Dipartimento di Ingegneria Elettrica ed Elettronica, Università degli Studi di Cagliari, Piazza d'Armi, 09123, Cagliari, Italy

[2]IRIT, CNRS, 118 Route de Narbonne, CEDEX 9, F-31062 Toulouse, France

[3]Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Turin, Italy

Abstract

This paper presents an approach leveraging machine learning techniques to monitor network traffic in search of vulnerability scanning activities. Indeed, attackers typically perform an initial reconnaissance phase to identify the vulnerabilities their target platforms expose, which they can abuse to perform cyberattacks. Classical network monitoring approaches have multiple limitations. Indeed, they are typically hindered by the presence of encrypted traffic, hamper user privacy resorting to Deep Packet Inspection (DPI), and cannot identify advanced scanning techniques such as slow scans. The research presented in this paper overcomes such limitations through machine learning classifiers that can detect vulnerability scans with flow-level granularity, employing statistical features evaluated on Layer 3 and 4 network packet headers. We demonstrate the feasibility of our approach training classifiers able to detect traffic originated by three well-known vulnerability scanning tools: OpenVAS, sqlmap, and Wapiti. The presented Proof-of-Concept classifiers are characterized by a high classification accuracy, with the best classifier reaching a balanced accuracy of 98%.

Keywords

Intrusion Detection Systems (IDS), Intrusion Prevention Systems (IPS), Machine Learning (ML), Network Security, User Privacy, Vulnerability Scanning

1. Introduction

In recent years, cyberattacks have been steadily on the rise. In particular, interactive intrusion techniques, where human malicious actors manually interact with target hosts instead of depending on malware, are becoming widespread, as reported by Crowdstrike in a recent report [1]. Indeed, these intrusion techniques typically leverage the vulnerabilities of hosts, exposing the services targeted by the malicious actors. These cyberattacks typically start with a reconnaissance phase, where attackers employ automated scanners to detect vulnerabilities, misconfigurations, or weaknesses of the target host or network, which they could exploit to

ITASEC 2024: The Italian Conference on CyberSecurity, April 08–12, 2024, Salerno, Italy

*Corresponding author.

†These authors contributed equally.

leonardo.regano@unica.it (L. Regano); daniele.canavese@irit.fr (D. Canavese); luca.mannella@polito.it (L. Mannella)

https://web.unica.it/unica/page/it/leonardo_regano (L. Regano); https://github.com/daniele-canavese (D. Canavese); http://github.com/LucaMannella (L. Mannella)

0000-0002-9259-5157 (L. Regano); 0000-0002-4265-7743 (D. Canavese); 0000-0001-5738-9094 (L. Mannella)

obtain entry points to take control of their target. Thus, promptly identifying vulnerability scans is paramount when protecting corporate networks with their associated services.

Attackers can count on many tools to conduct vulnerability scans, both general-purpose or specifically tailored to test multiple components of corporate networks, e.g., DBMSs, web servers, and network security appliances. For example, the OWASP foundation maintains an extensive list[1] of vulnerability scanning tools targeting web applications. Traditional defense techniques for vulnerability scan detection typically rely on identifying patterns, anomalies, and deviations from established baselines in the monitored network traffic. For example, volumetric traffic monitors are typically triggered when an abnormal number of requests in a short period originate from the same network client. This may indicate a vulnerability scan due to the large number of vulnerability tests executed by automated tools. Nevertheless, volumetric techniques are often unable to identify slow vulnerability scans properly [2], where attackers deliberately extend the scanning process, reducing the rate of requests to the target host to avoid suspicious traffic peaks that network traffic monitors may identify. Other approaches rely on Deep Packet Inspection (DPI) to identify known patterns in the traffic packets' payload. However, resorting to DPI techniques is becoming increasingly difficult due to the widespread adoption of end-to-end encryption. Furthermore, DPI raises concerns about user privacy and may not be compatible with related regulations like the European General Data Protection Regulation (GDPR).

This paper proposes a novel approach for automatically detecting vulnerability scans that can address the aforementioned challenges. In particular, we employ machine learning algorithms trained on network traffic statistics evaluated exclusively on Layer 3 and 4 header data. By avoiding DPI, our approach safeguards user privacy. Furthermore, since each TCP flow is evaluated separately, our approach overcomes the limitations of traditional volumetric traffic monitoring techniques, avoiding detection evasion from attackers employing slow scanning techniques. We demonstrate the feasibility of our approach by describing two machine learning models, employing respectively random forests and fully connected neural networks, trained on a data set we have built starting from network traffic captures of vulnerability scans executed on both vulnerable and secure websites using three well-known vulnerability scanners: OpenVAS, sqlmap, and Wapiti.

The rest of the paper is organized as follows. Section 2 presents the most relevant works applying Machine Learning techniques for traffic classification. Section 3 provides an overview of the vulnerability scanning tools we investigate in this study. Section 4 illustrates the dataset we have built to train the machine learning models, while Section 5 elaborates on their accuracy in detecting vulnerability scans. Finally, Section 6 concludes the paper, providing some insights into the future work we plan to conduct in this field.

2. Related Work

The simplest method of traffic classification relies on port numbers. However, using dynamic ports rendered this approach less effective, leading to the adoption of Deep Packet Inspection (DPI) techniques. Subsequently, the rise of encrypted traffic reduced the accuracy of DPI. As a

[1] https://owasp.org/www-community/Vulnerability_Scanning_Tools, last visited on February 25, 2024.

result, Machine Learning-based methods to classify packets have recently gained prominence. These methods only require unencrypted headers or information from encrypted data.

This paper presents an ML-based approach to detect vulnerability scanning, which could be integrated into the existing Intrusion Detection/Prevention System (IDS/IPS). IDS and IPS recognize possible security violations and anomalies by analyzing network packets. Some IDS/IPS software, such as Suricata[2], Snort[3], and Zeek[4], can detect scanners. However, they have limitations. For example, they may not be able to recognize the nature of very long scans [3] or identify the phenomenon collectively when multiple IPs perform scans simultaneously. Our proposed solution overcomes the aforementioned problems by analyzing TCP flow regardless of length and without relying on metrics such as IP address. Instead, the analysis is based solely on metrics extracted from packet headers.

To the best of our knowledge, this paper proposes the first general-purpose tool for detecting any type of vulnerability scanning. The closest work in literature, albeit limited to the detection of SQL injection attacks, is the one from Crespo et al. [4]. The authors created a dataset by executing SQL injection, capturing the related traffic alongside benign traffic, and calculating network traffic statistics employing Cisco Netflow [5]. They consequently trained multiple Machine Learning models, including k-Nearest Neighbour (KNN) and Random Forests (RF), achieving an accuracy score higher than 96%. In the remainder of this section, we present the most relevant works applying Machine Learning techniques to detect cyberattacks.

Wang et al. [6] introduced the end-to-end learning paradigm for protocol classification. They utilized a 1D CNN on raw data from the SCX VPN-nonVPN dataset. The authors did not report training or evaluation times or provide a complete comparison against traditional ML techniques. In their study, Lopez-Martin et al. [7] utilized Deep Learning (DL) techniques with time-series features to analyze traffic captures of the RedIRIS network. They proposed several classifiers based on either CNN or Recurrent Neural Network (RNN), or a combination of the two. Although they achieved good results, the authors did not compare their findings to classical machine learning algorithms.

In their study on app fingerprinting, Taylor et al. [8] proposed AppScanner, a framework capable of classifying traffic generated by 100 different applications. They employed either Support Vector Classifiers (SVC) or random forests for each traffic flow. Chen et al [9] demonstrated the suitability of Deep Neural Networks (DNN) for app fingerprinting. The pipeline was tested to classify the application on the first dataset and the protocol on the second dataset, which was used to generate the traffic data. A classic 2D CNN was used after projecting six time-series features into a multi-channel image.

Vargas et al. [10] developed a Bayesian Network model to classify attack types, such as worms and (Distributed) Denial of Service (DDoS/DoS) attacks, using time series. The reported accuracy is notably high. However, the model only considers terminated flows, making it unsuitable for real-time applications.

Naseer et al. [11] investigated the suitability of DL for anomaly detection. The authors trained several Deep Learning architectures on the NLS-KDD dataset, including Auto Encoders (AE),

[2]https://suricata.io, last visited on February 15, 2024.
[3]https://www.snort.org, last visited on February 15, 2024.
[4]https://zeek.org, last visited on February 15, 2024.

CNN, and Long Short-Term Memory (LSTM). This dataset comprises four attack typologies: DoS, User to Root (U2R), Root to Local (R2L), and Probe. Their comparison with traditional machine learning techniques showed that DL techniques improve accuracy by a few percentage points, albeit at the cost of a training time at least three orders of magnitude longer.

Wang et al. [12] proposed an approach where they constructed a dataset consisting of both malware and real traffic data. They used this dataset to train a CNN model. Although the authors claimed that their framework has early-stage detection capability, they did not provide the evaluation time of the overall pipeline. Therefore, it is uncertain whether such a framework can be used for real-time detection.

More recently, a systematic literature review by Abdulganiyu et al. [13] includes a section on applications of ML techniques to increase accuracy in Intrusion Detection Systems (IDS). Interestingly, we could not find any works related to vulnerability scanning detection in this review, excluding one from Vijayanand et al. [14]. In the latter, the authors develop an Intrusion Detection System tailored for wireless mesh networks based on Support Vector Machines (SVM). The authors employ a feature selection strategy based on a genetic algorithm and train their classifiers on the CICIDS2017 dataset[5]. The latter includes, among various attacks, also port scans executed with Nmap[6].

3. Vulnerability Scanning Tools

This Section presents the vulnerability scanning tools adopted to train our solution. A *vulnerability scanner* is a tool that can identify potential weaknesses in a system or network through automated rules and analysis. The primary objective of this software is to generate a comprehensive evaluation report or assessment that accurately reflects the current state of the system. On the contrary, the process of conducting a simulated and authorized computer attack is commonly referred to as a *penetration test*. This test enables the identification of potential vulnerabilities from the perspective of a potential attacker. The most common vulnerabilities, related to both popular software and different operating systems, are collected and cataloged by type and version of the program in various public lists, such as the *Common Vulnerabilities and Exposures* (CVE)[7]. It is important to stay informed about these vulnerabilities, periodically verify if they are affecting machines under our control, and take appropriate measures to mitigate any potential risks.

To identify these vulnerabilities, network administrators can use vulnerability scanner tools such as Burpsuite, Nmap, Nessus, OpenVAS, Qualys VMDR, sqlmap, Wapiti, and many more. Among all the available vulnerability scanners, those selected for the study are OpenVAS, sqlmap, and Wapiti. These tools were chosen due to their widespread use and open-source nature, in contrast to enterprise solutions like Nessus[8] and QualysVMDR[9] which require commercial relationships between companies (and are less likely to be used for malicious use). Burpsuite[10]

[5] https://www.unb.ca/cic/datasets/ids-2017.html, last visited on February 16, 2024.
[6] https://nmap.org, last visited on February 16, 2024.
[7] https://cve.mitre.org/, last visited on February 23, 2024.
[8] https://www.tenable.com/products/nessus, last visited on February 23, 2024.
[9] https://www.qualys.com, last visited on February 23, 2024.
[10] https://portswigger.net/burp, last visited on February 23, 2024.

is another example of free and open-source software, but its main function is to be used as an application proxy, thus moving away from the purpose of the study.

OpenVAS[11] (Open Vulnerability Assessment System) is a framework for vulnerability analysis and management. OpenVAS empowers users to perform a wide range of tests starting from the low level, analyzing the various network protocols, up to web applications and their functionalities. OpenVAS integrates the Nmap software from which it is possible to perform scans of TCP and UDP ports as a preliminary operation before testing for vulnerabilities. OpenVAS can monitor and generate system reports to identify weak points and improve system status. OpenVAS also enables automation and integration with other vulnerability assessment tools. OpenVAS maintains its own database, which is regularly updated with newly discovered CVEs and scripts capable of identifying them, along with a danger classification. This ensures that the resulting reports are as up-to-date as possible and enables the identification of the most critical vulnerabilities through vulnerability assessment.

sqlmap[12] is an open-source penetration testing tool that automates the process of identifying SQL injection vulnerabilities. An *SQL injection* is a particular case of injection technique that uses SQL instructions, a type of attack that usually allows discovering the structure of an application's database and retrieving sensitive information [15]. The use of this vulnerability is possible due to poor control of the data received as input that is used within SQL queries.

Wapiti[13] is a vulnerability analysis tool for web applications. It automates security checks of sites and performs black-box analysis which means it conducts non-specific tests for the victim, as it is not possible to know its source code. Through bots and scripts that read web pages, Wapiti tries to inject data into URLs and forms to uncover cross-site scripting (XSS) vulnerabilities [16]. An XSS vulnerability is made possible by poor control and/or poor handling of the data that a user can enter by interacting with the site.

4. Data Set Construction

This section reports how we built our data set for the analysis of vulnerability scanning.

4.1. Traffic acquisition

We designed a simple but effective scenario for capturing the vulnerability scanning traffic. In our scenario, we have only two end-points: a web server (the victim) and the attacker conducting the vulnerability scans. To achieve more realistic data, both the victim and attacker are connected through the Internet.

We tested three of the most used vulnerability scanners: OpenVAS[14], sqlmap, and Wapiti. Table 1 shows our attack machine's hardware and software specifications.

On the other hand, for the victim, we decided to test a variety of vulnerable websites (i.e., Juice Shop and DVWA) mixed with several traditional websites found on GitHub[15] running

[11] https://www.openvas.org/, last visited on February 23, 2024.
[12] http://sqlmap.org/, last visited on February 23, 2024.
[13] http://wapiti.sourceforge.net/, last visited on February 23, 2024.
[14] We performed the scanning on the target websites using the default Full and Fast Scan configuration.
[15] We downloaded several websites from https://github.com/search?q=website&type=repositories.

COMPONENT	SPECIFICATIONS
CPU	Intel® Core™ i7-1065G7 CPU @ 1.30GHz
RAM	8 GiB
operating system	GNU/Linux Ubuntu 20.04.1
kernel	5.15.0-72-generic
OpenVAS	22.6.2
sqlmap	1.7
Wapiti	3.1.6

Table 1
Attacker specifications.

on Apache and nginx. We used Tshark to record all the incoming vulnerability scans from the attacker. Table 2 reports the victim node's specifications instead.

For the benign traffic, we used some traffic captures that we had previously gathered for another work [17, 18]. These captures include human browsing traffic performed under Linux and Windows with Firefox, Chrome, and Edge on various websites.

Table 3 lists how many packets and TCP web flows we gathered. The benign class is the majority since it also includes multimedia traffic, which transports more data than traditional web pages and textual data. It is important to emphasize that, in our study, we are mostly interested in the flows since they represent our observations for the machine learning models.

COMPONENT	SPECIFICATIONS
CPU	Intel® Core™ i7-10875H CPU @ 2.30GHz
RAM	8 GiB
operating system	GNU/Linux Ubuntu 20.04.1
kernel	5.15.0-72-generic
Juice Shop	8.7.2
DVWA	1.10
Apache	2.3.4
nginx	1.22
Tshark	4.2.2

Table 2
Victim specifications.

4.2. Data analysis

Once we gathered the traffic, we created a Python script with scapy[16] to extract the statistics for each flow.

These metrics represent the classification features we used to categorize the data using our machine-learning algorithms. The features marked with 'both directions' (e.g., the number

[16] https://scapy.net/, last visited on February 28, 2024.

CLASS	PACKETS	FLOWS
benign	818 616	280 597
OpenVAS	43 123	17 147
sqlmap	2229	938
Wapiti	57 424	11 691

Table 3
Data set composition.

of sent packets) include two values: one from the attacker to the victim and vice versa. We purposely ignored the IP addresses and the ports since they are easy to spoof.

We then randomly split our data set into three sub-sets: the training set (containing 70 % of the samples), the development set for the hyperparameter optimization (15 % of the samples), and the test set (the remaining 15 % of the observations). Table 4 lists the number of samples (TCP flows) per set, while Table 5 enumerates the flow statistics we employed.

NAME	FRACTION [%]	SAMPLES
training set	70	217 261
development set	15	46 556
test set	15	46 556
total	100	310 373

Table 4
Samples in our experiments.

5. Classification Results

We trained two machine learning models for detecting vulnerability scanning: a random forest and a fully connected neural network. We opted for these models since, based on our experience, they generally offer high accuracy for detecting anomalous traffic conditions. We decided to avoid a cross-validation phase since we considered the size and variability of the dataset sufficient.

5.1. Random Forest

We used scikit-learn[17] to train a multiclass random forest, and we used a grid search for the hyperparameter optimization to maximize the accuracy. Table 6 reports the optimal parameters of our random forest.

Table 7 shows several classification metrics of our final optimized random forest. Our model performs excellently, reaching a balanced accuracy of nearly 99 %. SQLmap is the less recognizable class by the model; this is better emphasized in the confusion matrix, reported in

[17] https://scikit-learn.org/, last visited on February 28, 2024.

FEATURE	UNIT
# transmitted packets (both directions)	packet
# packets with payload (both directions)	packet
# retransmitted packets (both directions)	packet
# out of sequence packets (both directions)	packet
# packets with RST set (both directions)	packet
# packets with ACK set (both directions)	packet
# packets with ACK set and no payload (both directions)	packet
# packets with SYN set (both directions)	packet
# packets with FIN set (both directions)	packet
# packets with PSH set (both directions)	packet
# packets with URG set (both directions)	packet
# packets with ECE set (both directions)	packet
# packets with CWR set (both directions)	packet
# packets with NR set (both directions)	packet
# payload bytes excluding retransmissions (both directions)	B
# payload bytes including retransmissions (both directions)	B
# retransmitted bytes (both directions)	B
flow duration	ms
relative time of first payload packet (both directions)	ms
relative time of last payload packet (both directions)	ms
relative time of first ACK packet (both directions)	ms

Table 5
TCP features.

NAME	VALUE
criterion for measuring the quality of the tree splits	Gini index
maximum number of features per tree	20
maximum tree depth	8
number of trees	500

Table 6
Hyperparameters of our random forest.

Figure 1. Regarding live vulnerability scan detection, we report an AUC greater than 90% for flows longer than 5 packets.

sqlmap flows are frequently misclassified as OpenVAS or benign connections. This is most likely due to the fact that OpenVAS also performs SQL injection attacks such as sqlmap. On the other hand, being able to distinguish a SQL injection connection from a regular SQL request is complex without recurring to a deep packet inspection approach.

NAME	TOTAL	OPENVAS	SQLMAP	WAPITI	BENIGN
balanced accuracy	98.791				
accuracy	97.581	98.284	99.484	99.474	97.921
(macro) AUC	96.787	96.602	92.994	98.864	98.686
(macro) F1-score	81.488	84.905	48.936	93.267	98.844
(macro) recall	94.288	94.730	86.466	98.206	97.751
(macro) precision	74.953	76.926	34.125	88.802	99.961

Table 7
Statistics (in percentages) of our random forest.

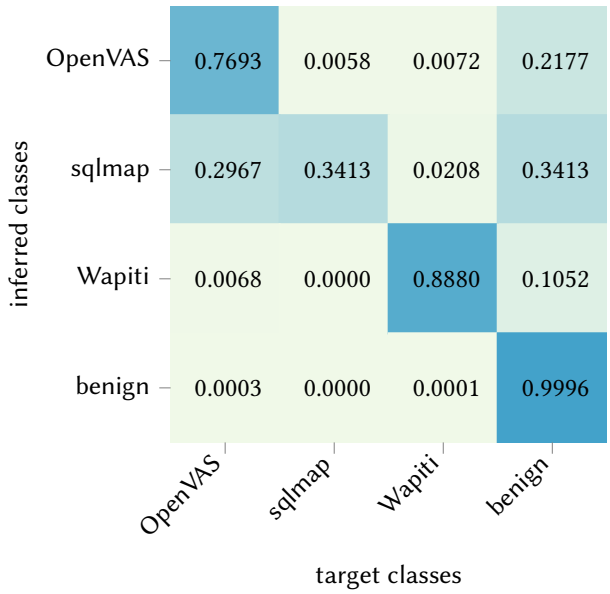

Figure 1: Normalized confusion matrix of our random forest.

5.2. Neural Network

In addition to the random forest, we also trained a fully connected neural network with skorch[18]. The hyperparameter optimization phase selected the values shown in Table 8.

NAME	VALUE
learning rate (for Adam)	0.0007
neurons per layer	512, 256, 128, and 64
activation function	ReLU

Table 8
Hyperparameters of our neural network.

[18]https://github.com/skorch-dev/skorch, last visited on February 25, 2024.

Table 9 lists the usual classification metrics for our neural network. The overall balanced accuracy of our model is close to 95 %, slightly below the random forest. Similarly to its counterpart, the neural network shows that sqlmap is harder to correctly identify. Regarding live vulnerability scan detection, we report an AUC greater than 85% for flows longer than 5 packets.

By comparing the two models, although the random forest shows a greater overall accuracy, the neural network seems better suited to distinguish sqlmap attacks.

NAME	TOTAL	OPENVAS	SQLMAP	WAPITI	BENIGN
balanced accuracy	94.684				
accuracy	89.368	92.626	95.496	99.519	91.095
(macro) AUC	89.815	80.975	88.745	95.744	93.794
(macro) F1-score	61.531	48.446	9.417	93.396	94.865
(macro) recall	83.029	68.002	81.955	91.667	90.495
(macro) precision	59.373	37.625	4.995	95.192	99.608

Table 9
Statistics (in percentages) of our neural network.

Figure 2 instead reports the normalized confusion matrix of our classifier.

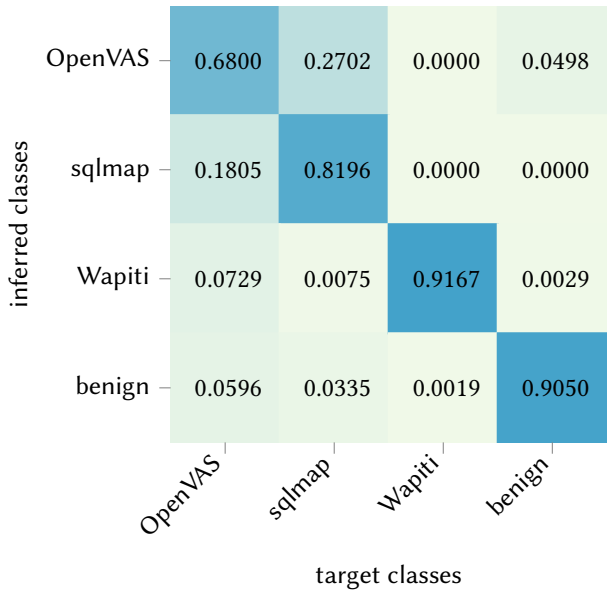

Figure 2: Normalized confusion matrix of our neural network.

6. Conclusions

In this paper, we presented a novel approach for detecting vulnerability scans. In particular, our approach can discriminate legitimate traffic from vulnerability scans by employing machine learning models using a set of traffic statistics computed on the IP and TCP headers as features. The Proof-of-Concept classifiers presented in this paper, based respectively on random forests and neural networks, can, in most cases, correctly classify vulnerability scans executed using three well-known tools: OpenVAS, sqlmap, and Wapiti. The random forests classifier performs best, with a balanced accuracy of 98.7%. The presented approach does not resort to DPI techniques, thus being fully compatible with encrypted traffic and respecting user privacy. Both classifiers can react swiftly to attempted vulnerability scans, needing only 5 packets in a flow to classify it accurately.

In the future, we plan to conduct an ad-hoc study targeting vulnerability scanning tools specifically tailored for IoT scenarios. Indeed, one of the vulnerability scanners we studied in this paper, OpenVAS, includes vulnerability scans tailored for IoT devices[19]. In previous work, we presented the IoT proxy [19], an architectural component to offload security controls from IoT devices, which typically have limited computational resources. The IoT proxy has a modular approach that permits the stacking of multiple security controls as virtual Network Security Functions (vNSF). We plan to investigate the feasibility of the approach presented in this paper for IoT vulnerability scanners and their effectiveness in bolstering the security of IoT architectures (e.g., Industrial Control Systems) when deployed as vNSF running on the IoT proxy.

Acknowledgments

The authors thank Alberto Solaro for his valuable contributions to this research activity during his master's thesis.

This work was partially supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

[1] CrowdStrike 2024 Global Threat Report, Technical Report, CrowdStrike, 2024. URL: https://go.crowdstrike.com/global-threat-report-2024.html.

[2] T. Yamashita, D. Miyamoto, Y. Sekiya, H. Nakamura, Slow scan attack detection based on communication behavior, in: Proceedings of the 2020 10th International Conference on Communication and Network Security, ICCNS '20, Association for Computing Machinery, New York, NY, USA, 2021, p. 14–20. URL: https://doi.org/10.1145/3442520.3442525. doi:10.1145/3442520.3442525.

[3] T. Yamashita, D. Miyamoto, Y. Sekiya, H. Nakamura, Slow scan attack detection based on communication behavior, in: Proceedings of the 2020 10th International Conference on Communication and Network Security, ICCNS '20, Association for Computing Machinery,

[19] https://www.greenbone.net/wp-content/uploads/solution_comparison_EN.pdf, last visited on February 22, 2024.

New York, NY, USA, 2021, p. 14–20. URL: https://doi.org/10.1145/3442520.3442525. doi:10.1145/3442520.3442525.

[4] I. S. Crespo-Martínez, A. Campazas-Vega, Ángel Manuel Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez Aparicio, C. Fernández-Llamas, Sql injection attack detection in network flow data, Computers & Security 127 (2023) 103093. URL: https://www.sciencedirect.com/science/article/pii/S0167404823000032. doi:https://doi.org/10.1016/j.cose.2023.103093.

[5] B. Claise, Cisco systems netflow services export version 9, Technical Report, Cisco System, Inc., 2004.

[6] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: 2017 IEEE Int. Conf. on Intell. and Secur. Inform., 2017, pp. 43–48.

[7] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things, IEEE Access 5 (2017) 18042–18050. doi:10.1109/ACCESS.2017.2747560.

[8] V. F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic, in: 2016 IEEE Eur. Symp. on Secur. and Privacy, 2016, pp. 439–454.

[9] Z. Chen, K. He, J. Li, Y. Geng, Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks, in: 2017 IEEE Int. Conf. on Big Data, 2017, pp. 1271–1276. doi:10.1109/BigData.2017.8258054.

[10] M. J. Vargas-Munoz, R. Martinez-Pelaez, P. Velarde-Alvarado, E. Moreno-Garcia, D. L. Torres-Roman, J. J. Ceballos-Mejia, Classification of network anomalies in flow level network traffic using Bayesian networks, in: 2018 28th Int. Conf. on Electron., Commun. and Comput., volume 2018-Janua, 2018, pp. 238–243. doi:10.1109/CONIELECOMP.2018.8327205.

[11] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks, IEEE Access 6 (2018) 48231–48246. doi:10.1109/ACCESS.2018.2863036.

[12] W. Wang, M. Zhu, X. Zeng, X. Ye, Y. Sheng, Malware traffic classification using convolutional neural network for representation learning, in: Int. Conf. on Inf. Networking, IEEE, 2017, pp. 712–717. doi:10.1109/ICOIN.2017.7899588.

[13] O. H. Abdulganiyu, T. Ait Tchakoucht, Y. K. Saheed, A systematic literature review for network intrusion detection system (IDS), International Journal of Information Security 22 (2023) 1125–1162. URL: https://doi.org/10.1007/s10207-023-00682-2. doi:10.1007/s10207-023-00682-2.

[14] R. Vijayanand, D. Devaraj, B. Kannapiran, Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection, Computers & Security 77 (2018) 304–314. URL: https://www.sciencedirect.com/science/article/pii/S0167404818303766. doi:https://doi.org/10.1016/j.cose.2018.04.010.

[15] L. K. Shar, H. B. K. Tan, Defeating sql injection, Computer 46 (2013) 69–77. doi:10.1109/MC.2012.283.

[16] S. Gupta, B. B. Gupta, Cross-site scripting (xss) attacks and defense mechanisms: clas-

sification and state-of-the-art, International Journal of System Assurance Engineering and Management 8 (2017) 512–530. URL: https://doi.org/10.1007/s13198-015-0376-0. doi:10.1007/s13198-015-0376-0.

[17] D. Canavese, L. Regano, C. Basile, G. Ciravegna, A. Lioy, Encryption-agnostic classifiers of traffic originators and their application to anomaly detection, Computers & Electrical Engineering 97 (2022) 107621. URL: https://www.sciencedirect.com/science/article/pii/S0045790621005528. doi:https://doi.org/10.1016/j.compeleceng.2021.107621.

[18] D. Canavese, L. Regano, C. Basile, G. Ciravegna, A. Lioy, Data set and machine learning models for the classification of network traffic originators, Data in Brief 41 (2022). URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85126104702&doi=10.1016%2fj.dib.2022.107968&partnerID=40&md5=d74892c6b0daf14665ac5d4457e8aa19. doi:10.1016/j.dib.2022.107968, cited by: 3; All Open Access, Gold Open Access, Green Open Access.

[19] D. Canavese, L. Mannella, L. Regano, C. Basile, Security at the edge for resource-limited iot devices, Sensors 24 (2024). URL: https://www.mdpi.com/1424-8220/24/2/590. doi:10.3390/s24020590.