Network Monitoring and Enumerating Vulnerabilities in Large Heterogeneous Networks

Martin Laštovička*†, Martin Husák*, Lukáš Sadlek*
*Masaryk University, Institute of Computer Science, Brno, Czech Republic
†Masaryk University, Faculty of Informatics, Brno, Czech Republic
lastovicka@ics.muni.cz, husakm@ics.muni.cz, sadlek@mail.muni.cz

***Abstract*—In this paper, we present an empirical study on vulnerability enumeration in computer networks using common network probing and monitoring tools. We conducted active network scans and passive network monitoring to enumerate software resources and their version present in the network. Further, we used the data from third-party sources, such as Internet-wide scanner Shodan. We correlated the measurements with the list of recent vulnerabilities obtained from NVD using the CPE as a common identifier used in both domains. Subsequently, we compared the approaches in terms of network coverage and precision of system identification. Finally, we present a sample list of vulnerabilities observed in our campus network. Our work helps in approximating the number of vulnerabilities and vulnerable hosts in large networks, where it is often impractical or costly to perform vulnerability scans using specialized tools, and in situations, where a quick estimate is more important than thorough analysis.**

I. Introduction

Keeping track of assets in a computer network, enumerating hosts and services, and assessing vulnerabilities are everyday tasks in the operations of computer security incident response teams (CSIRT) and security operation centers (SOC). These tasks are perceived as prerequisites to what is referenced in the literature as cyber situational awareness (CSA), or network-wide situational awareness when discussing computer networks specifically [1]. CSA is an application of general situational awareness, which is a continuous process of perceiving the environment, understanding the processes in it, and projecting future changes in the situation [2]. Although the research and development in the field of CSA brought numerous results [1], most of the proposed approaches and processes depend on high-quality inputs, such as reliable observation of a computer network.

Observing large networks is a problem of its own [3], especially when the administration of the network is distributed among numerous departments and the infrastructure is continuously changing and heterogeneous in terms of used hardware and software. For example, a campus network of a university with many departments, desktops, servers, research equipment, and numerous students and employees connecting with their own devices is a challenging environment for keeping CSA [1]. Even though important assets are typically known, it is hard to keep track of other devices and systems in the network. Thus, when a new vulnerability appears, or we want to know how harmful can a certain vulnerability be for a given network, it is hard to make even a rough estimate. Numerous vulnerability scanners are available for detecting vulnerabilities in the network, but they are often highly specialized or impractical for use in the large scale. They come useful when performing a detailed measurement, but they are also expensive, may not scale well, and the tools for discovering particular vulnerabilities may take time to be delivered or may not exist at all. Thus, we are looking for methods and tools that may cover the empty space between casual network monitoring and vulnerability assessment. Specifically, we want to know, for any given vulnerability, how many devices in a large network may be vulnerable because even a rough estimate helps in prioritizing the threats the vulnerabilities posses.

To formalize the scope of our work, we pose the following research questions:

1) How precise are network monitoring tools in recognizing software and its version on hosts in a computer network?
2) How precise are descriptions of vulnerabilities with regards to name and version of the vulnerable software?
3) How precisely can we estimate the number of vulnerable hosts for a given vulnerability?

To resolve these problems, we propose an approach based on common network monitoring methods and the combination of their outputs with vulnerability databases. First, active and passive network monitoring methods are evaluated from the perspective of host and service assessment, e.g., OS fingerprinting and service discovery. We used active approach using Nmap [4], and passive approach using flow-based monitoring (NetFlow) [3] and corresponding data analytics. NVD[1] is a *de facto* standard database of vulnerabilities in CVE (Common Vulnerabilities and Exposures) format. Although other databases exist, they are not comparable in terms of popularity, size, and access to the data. The data from NVD are combined with the outputs of network monitoring using a CPE as a common identifier. CPE (Common Platform Enumeration) is a system descriptor consisting of the software vendor, name, version, and other entries, which is found in CVE description and can be constructed from the outputs of network monitoring. Matching CPEs are the primary indicators of potentially vulnerable device that we use in this work.

978-1-7281-4973-820$31.00 © 2020 IEEE

[1]National Vulnerability Database, https://nvd.nist.gov/

The rest of this paper is organized as follows. Section II summarizes technical background and related work. Section III presents measurements conducted to answer the research questions, i.e., assessment of recent vulnerabilities from NVD and processing the results of campus network monitoring using passive and active approaches. The data from vulnerability assessment and network monitoring are correlated and discussed in Section IV. Section V concludes the paper.

II. Background and Related Work

The background for this work is in two main areas, vulnerability assessment and network monitoring. First, we briefly discuss vulnerability databases, the main source of information on vulnerabilities. Subsequently, we discuss various approaches to assess potentially vulnerable hosts in the network. In this paper, we focus on network-based approaches applicable on a large scale and, thus, omit host-based agents and other approaches that discover vulnerabilities only locally.

Vulnerability assessment is commonly understood as a process of identifying and prioritizing the vulnerabilities in a system, such as in a computer network. However, identification of vulnerabilities in information systems and computer networks is a hard problem of its own. However, in this paper, we are not interested in the discovery of particular vulnerabilities and calculating their impact, but instead on one of its fundamentals steps, enumerating potentially vulnerable hosts. Such network-wide awareness of hosts and vulnerabilities is a cornerstone for any further assessments [1]. Vulnerabilities in cyber security are fortunately documented by the community [5]. The information on vulnerabilities may be obtained from various sources, such as NVD. Although many other databases exist, their content usually overlaps with NVD. Interesting data can be found in vulnerability databases operated by various vendors, who publish information on vulnerabilities related to their products, but with more detailed descriptions, including vulnerable system versions and patches.

Passive monitoring of network traffic allows continuous assessment of the network and identification of assets. Using the NetFlow technology [3] enables this monitoring even on high-speed backbone network by aggregating the packets into flows without the loss of information necessary for identification of the software producing the traffic. Goodall et al. [6] proposed using NetFlow to map assets in the network for the needs of cyber situational awareness. An example of a tool for network assessment based on NetFlow is Nfsight [7]. Lippmann et al. [8] proposed a method for operating system identification from TCP/IP packet headers and their normalization for use in different networks. these characteristics were complemented by analysis of application layer headers of DNS traffic [9] or update procedures [10] or user-agent field [11]. The following work showed that passive identification is possible even in encrypted traffic [12] and that combination or the approaches [13] can enhance the results in heterogeneous networks. In this paper, we are interested in practical applications of the theoretical methods in combination with machine learning algorithms [14] to build the identification models semi-autonomously.

Active network monitoring has an advantage over passive methods that it can create requests as needed and analyze the response. Hence, it can easily identify known services running on the hosts in the network. However, active probing heavily relies on the host availability during the scan and firewall settings allowing scans. The most widespread tool for active scanning is currently Nmap [4]. It offers a wide range of scanning techniques and the ability to identify service running on a scanned port. Beside internal fingerprinting a human-readable name of the software, it can map the software to the corresponding CPE identifier. This allows us to match vulnerabilities with the host providing service with the same software without the need to create special request to identify new vulnerabilities.

There is a plethora of specialized vulnerability scanners that are mostly based on using active probing to find potentially vulnerable hosts and services and verify the vulnerability. We are mostly interested in network-based scanners in contrary to host-based scanners and agents that need to be installed on hosts. However, network-based vulnerability scanners have their limitations, too. They are hindered by firewalls and might be imprecise. Nessus[2] is a commercial vulnerability scanning tool that scans for vulnerabilities such as buffer overflows, directory traversals, default password, known backdoors, and misconfigurations. OpenVAS[3] is an independent open source project forked from Nessus before it became a proprietary license. Retina[4] is another commercial vulnerability scanner. It searches for vulnerabilities and helps to prioritize the remediations. Open Vulnerability Assessment Language (OVAL) is an XML-based language for specifying machine configuration. For some vulnerabilities, the vulnerable configuration is specified in OVAL. The language standardizes the configuration information of systems, analyzing the system for the presence of the specified state (vulnerability, configuration, etc.), and reporting the results of this assessment. Finally, given the specific nature of web application, there are numerous vulnerability scanners focused on finding vulnerabilities in web pages and popular content management systems (CMS) and their plugins. Examples of such tools are CMSscan[5], CMSeeK[6], and WhatWeb[7]. It is worth noting that almost all of them are primarily vulnerability scanners, although they might be used as CMS identification tools as well.

Apart from the approaches presented above, there are also third-party tools and activities with similar intentions. These activities are typically based on Internet-wide active network scanning. The results of such scans are collected, processed, and distributed under varying conditions, including commercial interest. A well-known organization performing Internet-

[2]https://www.tenable.com/products/nessus/nessus-professional
[3]http://www.openvas.org
[4]https://www.beyondtrust.com/products/retina/
[5]https://github.com/ajinabraham/CMSScan
[6]https://github.com/Tuhinshubhra/CMSeeK
[7]https://github.com/urbanadventurer/WhatWeb

wide scans is Shadowserver[8]. Shodan[9] has become a popular tool among threat analysts as it provides various information on scanned hosts using the techniques of device fingerprinting for various purposes, such as discovery of specific devices [15] and vulnerability assessment [16]. Research on active network measurements led to the development of Censys, another popular tool that utilizes the data from Internet-wide scans [17]. The limitations of scanning by third-parties are significant. The scanned devices need to be running and responding to scans, many networks are protecting themselves from attackers by firewalls and detecting and mitigating the scans [18], other networks may drop the responses due to overload caused by the scan, and the scans themselves are difficult to perform in such a large scale.

III. Monitoring and Measurements

In this section, we present the necessary measurements and data retrievals conducted to answer the research questions. We conducted the measurements in the week starting on March 18, 2019 in our campus network described further in the text. First, we assessed the information on recent vulnerabilities to enumerate vulnerable system identifiers. Then, we conducted a series of measurements, both active and passive, to enumerate the system identifiers present in our campus network. The active and passive measurements are accompanied by the data obtained from third-party tools with similar functionality. The processes and preliminary results are presented here; the correlations of the data are presented in the following section.

We performed all measurements in the campus network of Masaryk University. The network has more than 40,000 users and consists of one /16 network segment in CIDR notation. From this pool of 65,535 IPv4 addresses, roughly 25,000 are actively used, and around 19,000 unique IP addresses actively communicate each day. Altogether, they produce 6.91k network flows per second on average with peaks to 11k flows per second in rush hours.

The active scans suffer from probe discarding and filtering on firewalls. Only moderate filtering rules are applied globally to preserve the network neutrality of the academic environment, and the scanning itself was carried out from within the university network. However, each department might have its own independent firewall rules, and the endpoint departments or segments can apply their rules on the routers or hosts which hinders the active scan activity.

A. Vulnerability Assessment

We used the NVD as a source of information on vulnerabilities. Following our proposed use cases, we focused on recent vulnerabilities, specifically those published or updated from the start of the year 2019 to the beginning of an experiment (March 17, 2019). From this period, we downloaded information on 1,441 vulnerabilities identified by their CVE number. Out of those number, 1,422 CVEs contained at least one CPE identifier. CPE describes a system affected by a vulnerability using the string of the following structure:

[8]https://www.shadowserver.org/

[9]https://www.shodan.io/

TABLE I
Level of detail in CPEs in CVEs published from January 1 to March 17, 2019.

Vendor	Product	Version	Update	Edition	Count
✓	*	*	-	-	0
✓	✓	*	-	-	184
✓	✓	✓	-	-	1,105
✓	✓	✓	✓	-	132
✓	✓	✓	✓	✓	1

```
cpe:/<part>:<vendor>:<product>:<version>:
  <update>:<edition>:<language>
```

The <part> field designate if the CPE relates to an operating system, hardware platform, or application. Other fields are self-explanatory.

We decomposed the CPE content in Table I. As we can see, the vast majority of CPEs present in recent CVEs describe a system's vendor, product, and version. In some cases, CPEs include identifiers of the system's update and edition, which might be hard to distinguish from other, non-vulnerable instances. Information on product name and version are not always included. However, an asterisk is used instead, denoting that the vulnerability affects all the vendor's products or all products' versions. Thus, we may assume that the precision of CPEs is sufficient to identify a vulnerable system.

B. Passive Monitoring

At first, we inspected the university traffic to get a baseline of how many devices communicated during the experiment. We mark an IP address as active if it sent at least one UDP packet or a TCP packet with SYN flag set during the one-day interval. By this heuristic, we estimate that 18,749 IP addresses were active. All measurements were conducted on the backbone network connecting the university to the Internet. Hence, we omit hosts that communicate only within campus internal network, and their traffic is not routed through the main backbone router with monitoring point.

Operating system identification runs continually and processes data in 5-minute batches which is a typical interval for NetFlow monitoring [3]. In each batch, an OS was assigned to every IP from the university range according to classification result. The logic of identification followed the process described in [13] with the TCP/IP parameters method updated according to [14] to use machine learning model instead of a manually created dictionary. The results were then stored in a database for further evaluation. At the end of our measurement, the total number of 18,018 IP addresses were assigned at least one operating system. From this, 43 IPs were assigned to OS "Unknown" and nothing else. Otherwise, the classification was able to identify at least the vendor of OS for the hosts. To quantify the classifier usability in vulnerability assessment, we inspect the level of detail for each IP address. Only the vendor and nothing else was identified in 166 cases. The vendor and product without version were identified for 7,813 addresses. For the highest number of 10,039 addresses, the operating system was identified with its version. Our classifier is unable to detect update or edition of the software; hence, we cannot compare its results this precisely.

C. Active Scanning

We have used Nmap 6.47 wrapped in Python-nmap library version 0.6.1 to scan the university network. The arguments for scanning were *-sV* to detect services running on ports including their CPE, and arguments *-n* and *-T5* were used to speed up the scan.

The scan started on March 18, took 16 hours and 9 minutes to finish, and found a total number of 5,771 hosts with *state up* in Nmap terminology. From those, 3,152 hosts were detected with all ports *closed* or *filtered* which means they are behind a firewall and we cannot detect the services running and evaluate them for vulnerabilities. The rest of this paper thus work only with the hosts with at least one port with status *open*. Next, we inspected the Nmap capability to identify the software behind an open port. Nmap scanner can identify the product, version, and CPE, where in some cases the CPE is missing even though the product was identified. The number of hosts with the product identified without a version is 252. Both product and version were identified for 1,620 hosts.

Similarly, we have used specialized tool WhatWeb version 0.4.9 to detect specific software of university websites. We have created a list of all domains registered and run the scanner on them. This procedure discovered 382 active IP addresses from the campus network running a webserver. From those, only in 19 cases, the web scan was unable to identify any CPE. The identified CPEs were 3 with the vendor only, 56 with the product and 304 IPs with a CPE to the level of the software version.

D. Third-Party Tools

The usage of third-party scanners is quite straightforward. Shodan Search Engine periodically scans the whole internet to detect any potentially vulnerable devices and stores the results. One of their scanning activity is aimed to discover known vulnerabilities from CVEs on the targets, which is directly our goal. Shodan allows querying its database to discover all hosts in specified IP range on which a specific CVE was detected.

IV. Data Evaluation and Discussion

In this section, we correlate the data retrieved from measurements and vulnerability assessments presented in the previous section. First, we compare and discuss the monitoring coverage and precision of active and passive network monitoring approaches. This summarizes the first and second research questions of this paper that were partially answered in the previous section. Subsequently, we correlate system identifiers (CPEs) from vulnerabilities and network monitoring to enumerate potentially vulnerable hosts in the network, which answers the third research question of this paper. Finally, we discuss the results and their usability for large-scale network security assessment.

TABLE II
Coverage of monitoring tools.

Method	Vendor	Product	Version	Update	Total
NetFlow	0,89 %	41,67 %	53,54 %	0,00 %	96,10 %
Nmap	0,00 %	1,34 %	8,64 %	0,00 %	9,98 %
WhatWeb	0,02 %	0,30 %	1,62 %	0,00 %	1,94 %

A. Monitoring Coverage

Our first research question aims for the precision of software recognition on hosts. To answer this, we take a look at coverage of the methods in the sense of how many hosts they can identify the software and on which level of detail. The results are summarized in Table II, and the percentage represents the fraction of hosts with SW identified to the corresponding level taken to the total number of active hosts in the network. We have omitted the Edition column as no method could identify any edition neither update. Nmap was evaluated separately for the identification of product and Nmap produced CPE.

The active scanning methods can discover only a small fraction of hosts, but when they do, the result is usually with software version included. The web scan was limited only to web servers; hence, its results are not directly comparable. The passive methods have much higher coverage as they can continuously monitor the network and iteratively hone the results to identify software on almost every active host in the network. However, passive monitoring relies on the captured traffic and cannot influence the data processed, which hinders its identification accuracy. As shown in the related work, the accuracy of flow-based passive fingerprinting is 85.8 % [13], and we need to take it into account when expressing the results as they contain such volume of error.

B. Enumerating Vulnerable Hosts

After having done software identification and vulnerability evaluation, the next logical question is how can we join the results to discover vulnerable hosts in the network. We mapped the CVEs to hosts according to a few simple rules:

- Full match of CVE CPE and detected CPE.
- Match of the vendor, product, and version with CVE, including update and edition.
- Match of detected vendor and product with CVE for a specific version.
- Match of the detected vendor with CVE for a product.

Doing this, we tried to avoid mapping of a vulnerability of a specific build of a product version to a general detection of a product. Generally, we allowed the detection to have one level of detail less than the CVE. By this procedure, out of the 1441 CVE disclosed in 2019, we have assumed 223 CVEs to be present on the hosts in the network. Those CVEs were mapped on 40 distinct CPEs, where a CVE could map to multiple CPEs or multiple CVEs map to a single CPE. The mapping is summarized in Table III. AV/N denotes the number of CVEs that have *network* as their attack vector. Finally, vulnerabilities discovered by Shodan scanner are directly mapped on the

IP addresses without any CPE. The overlap between the Nmap and passive monitoring was only in six CPEs which demonstrates their different scope in found hosts and software.

TABLE III
MAPPING VULNERABILITIES TO HOSTS.

Method	Unique CPEs	CVEs	AV/N	Hosts
NetFlow	71	221	118	17,418
Nmap	169	37	13	815
WhatWeb	223	29	29	136
Shodan	N/A	19	19	38

Table IV presents top vulnerabilities according to the number of affected hosts for each detection method. The results follow the coverage of methods. The passive monitoring discovered significantly more vulnerable hosts than active scanning. The lower accuracy of passive methods does not does influence mapping of generic CPEs like *cpe:/o:linux:linux_kernel:** or *cpe:/o:microsoft:windows_10:**. Both methods identified similar numbers of linux-based systems, while Microsoft operating systems were rarely seen in the results of active scanning, but prominent in the results of passive monitoring. This might be explained by the differences in the behavior of desktops (mostly Windows stations) and servers (mostly linux-based) on the network. However, when comparable, the two methods provided similar results. The third-party scanner, Shodan, discovered a very limited number of vulnerable hosts compared to active scanning performed locally. This was expected as it might be hindered by firewalls and other obstacles. Interestingly, Shodan provided very similar results to an active scan of webs using WhatWeb. Thus, we may conclude that Shodan is a complement, rather than an alternative, to Nmap. Overall, the results of passive monitoring can be considered as a superset of all the other results, while the results of Nmap do not overlap with results of Shodan, which overlaps with the results of WhatWeb. Finally, CPEs related to operating systems are present more often than CPEs related to applications, with the exception of scanning with WhatWeb.

C. Discussion

The coverage of monitoring methods is the most striking difference. Passive network monitoring can enumerate almost every active device in the network, and even a low number of observed packets is enough for its identification with tolerable accuracy. Active scanning, in comparison, only provides a view of the network in a specific moment and is influenced by settings of all firewalls on the route between the scanner and the target host. This result in the enumeration of a small fraction of hosts. To increase the coverage, active scans can be performed repeatedly at distinct times to cover the hosts which are up for a limited amount of time during the day. A solution for this could be a hybrid approach which detects active devices using passive monitoring and then run the active scan only for the hosts that are confirmed to be turned on.

The reliability of identification can be expressed by standard metrics of accuracy, precision, and recall, but is usually very hard to measure in large scale experiments. Passive monitoring methods generally have lower accuracy as they depend on the traffic they observe. They match the captured packets with a predefined pattern or machine learning model and aims for the highest probability of correct identification which results in a special case of false positives important for vulnerability assessment. A popular system, such as Windows, will suppress an obscure system with similar traffic characteristics by dominating the identification probability. However, such popular systems are more likely to obtain patches, and we would prefer different behavior of passive identification for vulnerability enumeration. Active monitoring has the advantage of sending probes crafted specially to identify the selected service, which is reflected in their high accuracy. However, the scope of the scan is limited by the capabilities of available tools. Furthermore, the results depend on the settings of active scanner whether it scans only top used ports (e.g., top 1000 used ports identifies around 93 % of ports [4]) or the whole port range. One must always balance the scan speed and precision when using the active approach.

The next criterion of the quality of vulnerability enumeration is the level of detail of software identification. It is understandable that passive or active measurements have limits and are not capable of providing a specific build of the software. On the other hand, when disclosing a vulnerability as a CVE, the author should know exactly where he found the vulnerability and fill in the CPE accordingly. However, as shown in Section III-A, the majority of published CVEs contain CPEs that identify only the software and its version, thus making the usability of such CVEs and CPEs questionable. With such generic CPE, the mapping of vulnerabilities to hosts will always end up identifying thousands of false positives, i.e., already patched systems. Thus, there is a need to either identify the particular version of the system or at least find out if the system is being updated or not.

The last topic for discussion is what kind of vulnerabilities is interesting to enumerate in large networks. With the large numbers of hosts and their vulnerabilities, the administrator needs to prioritize their work. The first step could be to focus the remotely exploitable vulnerabilities as they are more likely to be exploited. From the 223 unique CVEs that we found in the campus network, 120 have remote exploitability in CVSS score. Similarly, CVEs could be prioritized by their severity or expected impact [5]. Apart from these static metrics, we could also use the results of enumeration which are sufficient for rough estimates and network-wide situational awareness. Then, the top priority might be assigned to a vulnerability in the most hosts or to a host with most vulnerabilities.

V. CONCLUSION AND FUTURE WORK

In this paper, we demonstrated and compared several approaches to vulnerability enumeration in large heterogeneous networks. We used a list of vulnerabilities published in NVD for almost three months before the measurements. Using the CPE identifiers, we were able to match vulnerabilities to hosts in a campus network. Methods based on passive network traffic monitoring provided the most results, but are also more likely

TABLE IV
TOP-5 VULNERABILITIES DISCOVERED IN THE NETWORK BY EACH METHOD.

Method	Vulnerability	CPE	Nmap	WhatWeb	NetFlow	Shodan
NetFlow	CVE-2019-0663, CVE-2019-0662, CVE-2019-0660, and more than 5 others	cpe:/o:microsoft:windows_10:* cpe:/o:microsoft:windows_7:* cpe:/o:microsoft:windows_8.1:*	0	0	12,626	0
Nmap	CVE-2019-9003, CVE-2019-7308	cpe:/o:linux:linux_kernel:*	739	0	2,011	0
	CVE-2019-8912	cpe:/o:redhat:enterprise_linux:* cpe:/o:linux:linux_kernel:*	729	0	2,010	0
	CVE-2019-9213	cpe:/o:linux:linux_kernel:*	729	0	2,010	0
	CVE-2019-8980 and 5 others	cpe:/o:linux:linux_kernel:*	728	0	2,010	0
WhatWeb	CVE-2019-0190	cpe:/a:apache:http_server:*	75	55	0	2
	CVE-2019-9024, CVE-2019-9023	cpe:/a:php:php	0	52	2,012	18
	CVE-2019-9638, CVE-2019-9637 and 5 others	cpe:/a:php:php	0	52	470	13
Shodan	CVE-2019-9024, CVE-2019-9023, CVE-2019-9021, CVE-2019-9020, CVE-2019-6977	N/A	0	52	2,012	18

to be imprecise. Active network scanning achieved quite good precision, but its scope is limited to unfiltered and responding hosts. Both methods can be used for rough vulnerability enumeration. Finally, third-party scanners like Shodan turned out to be limited in scope and detection capabilities and are not suitable for estimating the security situation in the network.

In our future work, we are going to focus more on the correlation of outputs of passive network monitoring. Enumerating hosts in the network, identifying their operating system, and estimating which hosts are vulnerable could be further improved by estimating which hosts are patched or running antivirus software. Using passive monitoring, we can infer such information using one of the approaches to passive OS fingerprinting, i.e., detecting communication between hosts and update servers of OS and antivirus vendors. Patched hosts that run an updated antivirus software may then be excluded from the enumeration of vulnerable hosts, while unpatched hosts without an antivirus may be located more easily. Thus, we may achieve more detailed network-wide situational awareness using only passive network monitoring. Our results may also serve as a benchmark for testing vulnerability assessment and scanning tools, including network-wide scanners (Shodan, Censys, etc.) and dedicated vulnerability scanners. These tools offer many interesting features, but their scope and precision were not yet properly measured.

ACKNOWLEDGMENT

This research was supported by the Security Research Programme of the Czech Republic 2015 - 2020 (BV III / 1 VS) granted by the Ministry of the Interior of the Czech Republic under No. VI20172020070 Research of Tools for Cyber Situational Awareness and Decision Support of CSIRT Teams in Protection of Critical Infrastructure. Martin Laštovička is Brno Ph.D. Talent Scholarship Holder – Funded by the Brno City Municipality.

REFERENCES

[1] A. Kott, C. Wang, and R. F. Erbacher, *Cyber defense and situational awareness*. Springer, 2014.

[2] M. R. Endsley, "Situation awareness global assessment technique (SAGAT)," in *Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National*. IEEE, 1988, pp. 789–795.

[3] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 2037–2064, Fourthquarter 2014.

[4] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

[5] J. Komárková, L. Sadlek, and M. Laštovička, "Community based platform for vulnerability categorization," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018.

[6] J. R. Goodall, A. D'Amico, and J. K. Kopylec, "Camus: Automatically mapping cyber assets to missions and users," in *MILCOM 2009 - 2009 IEEE Military Communications Conference*, Oct 2009.

[7] R. Berthier, M. Cukier, M. Hiltunen, D. Kormann, G. Vesonder, and D. Sheleheda, "Nfsight: netflow-based network awareness tool," in *Proceedings of LISA'10: 24th Large Installation System Administration Conference*, 2010.

[8] R. Lippmann, D. Fried, K. Piwowarski, and W. Streilein, "Passive operating system identification from TCP/IP packet headers," in *Workshop on Data Mining for Computer Security*, 2003.

[9] T. Matsunaka, A. Yamada, and A. Kubota, "Passive OS fingerprinting by DNS traffic analysis," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference On*. IEEE, 2013, pp. 243–250.

[10] S. Mossel, "Passive OS detection by monitoring network flows," 2012.

[11] S. Shah, "HTTP Fingerprinting and Advanced Assessment Techniques," *BlackHat Asia*, 2003. [Online]. Available: http://www.blackhat.com/presentations/bh-asia-03/bh-asia-03-shah/bh-asia-03-shah.pdf

[12] M. Husák, M. Čermák, T. Jirsík, and P. Čeleda, "HTTPS traffic analysis and client identification using passive SSL/TLS fingerprinting," *EURASIP Journal on Information Security*, vol. 2016, no. 1, p. 6, Feb 2016.

[13] M. Lastovicka, T. Jirsik, P. Celeda, S. Spacek, and D. Filakovsky, "Passive OS fingerprinting methods in the jungle of wireless networks," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018.

[14] M. Laštovička, A. Dufka, and J. Komárková, "Machine learning fingerprinting methods in cyber security domain: Which one to use?" in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, June 2018, pp. 542–547.

[15] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114 – 123, 2014.

[16] B. Genge and C. Enăchescu, "Shovat: Shodan-based vulnerability assessment tool for internet-facing services," *Security and communication networks*, vol. 9, no. 15, pp. 2696–2714, 2016.

[17] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. ACM, 2015, pp. 542–553.

[18] S. Lee, S. Shin, and B. Roh, "Abnormal behavior-based detection of shodan and censys-like scanning," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2017, pp. 1048–1052.