A Performance Assessment of Free-to-Use Vulnerability Scanners - Revisited

Ricardo Araújo[1], António Pinto[2,4], and Pedro Pinto[3](✉)

[1] Instituto Politécnico de Viana do Castelo, Viana do Castelo, Portugal
riaraujo@ipvc.pt

[2] CIICESI, ESTG, Politécnico do Porto, Porto, Portugal
apinto@inesctec.pt

[3] Instituto Politécnico de Viana do Castelo IPVC, ISMAI & INESC TEC, Viana do Castelo and Porto, Portugal
pedropinto@estg.ipvc.pt

[4] CRACS & INESC TEC, Porto, Portugal

Abstract. Vulnerability scanning tools can help secure the computer networks of organisations. Triggered by the release of the Tsunami vulnerability scanner by Google, the authors analysed and compared the commonly used, free-to-use vulnerability scanners. The performance, accuracy and precision of these scanners are quite disparate and vary accordingly to the target systems. The computational, memory and network resources required be these scanners also differ. We present a recent and detailed comparison of such tools that are available for use by organisations with lower resources such as small and medium-sized enterprises.

Keywords: Vulnerability scanning · Comparison · Open source · Tsunami

1 Introduction

Hackers are launching more sophisticated attacks on every possible weakness of computer networks and systems [19]. Vulnerability scanning tools are automated tools that scan applications and networks, trying to identify security vulnerabilities, such as outdated or non-patched software. On the one hand, a systematic vulnerability scanning procedure tends to be more efficient in protecting an organisation [4], be it a manual or an automated test. On the other, a manual security testing requires more resources (human and financial), hampering its adoption in small and medium-sized enterprises (SMEs). As an alternative, open source or free-to-use automated vulnerability scanning tools may be used by organisations to better improve their cybersecurity resilience, even in the case of SMEs.

The detection efficiency of vulnerability scanning tools is heavily dependent on their vulnerability database. A large database will enable a more thorough detection. New vulnerabilities are discovered frequently, which means that these tools

A. Jøsang et al. (Eds.): SEC 2021, IFIP AICT 625, pp. 53–65, 2021.
https://doi.org/10.1007/978-3-030-78120-0_4

are only efficient if they maintain a steady pace of updates to their vulnerability databases. One would expect that, if such a tool is developed by a large company or organisation, its vulnerability database would also be a large one, it would see frequent updates and would be a single tool that would be sufficient for SMEs. We assume that SMEs will have a small in-house support team with only a periodic availability to pursue vulnerability assessments.

More recently, the Tsunami[1] vulnerability scanner was made open source by Google and, despite being clearly marked as a non-official product, it triggered the assessment work described herein. In short, the authors focused on answering the following questions:

- Q1: What is the most efficient, free-to-use, vulnerability scanning tool currently available?
- Q2: How does Tsunami compare to similar tools currently available?
- Q3: Is Tsunami well suited to be used by SMEs?

This paper is organised in sections. Section 2 presents the related work, clarifying the differences from the work herein to theirs. Section 3 presents the technical features of the selected vulnerability scanners and the test-bed designed to compare and assess them. Section 4 compares the selected tools and presents results and our analysis. Finally, Sect. 5 concludes the work.

2 Related Work

There are research works that focus on comparing tools that evaluate a specific type of vulnerability, such as web application scanning tools. Examples being [3,8–10,13,15,16]. Others focus on one specific problem, such as SQL injection attacks [2].

In [20], the authors compare free and commercial off the shelf vulnerability scanning tools. Despite assessing a large set of such tools and considering both functionality and the possibility of correlating the outcomes of the tools with additional information. While valid research at the time, it is now mostly outdated. Some of the referenced tools are no longer available.

In [12], the authors also present a large quantitative comparison of vulnerability scanning tools. Their focus was on the direct output of the tools or, in other words, the number of vulnerabilities these tools identify. They focused on functionality and on accuracy. Our work differs by focusing on tools to be usable by SMEs. Moreover, newer tools were launched, not available at the time, some being open source and free-to-use tools. Additionally, they did not perform resource usage comparison. They conclude that some tools are better at detecting vulnerabilities of Windows systems, others at detecting vulnerabilities of Linux systems.

In a more recent work [14], the authors presented a performance-based comparison between two tools: Nessus and Retina. They selected these two because they considered them as the most used free vulnerability scanning tools. Of the

[1] https://github.com/google/tsunami-security-scanner.

two, Retina has now been discontinued by its developer, which issued a notification stating its end of life by December 31, 2019. We aim to perform a similar work to the one of Kushe but being a more up to date, more complete and more thorough one.

Similar work was presented in [5], where the authors opted for comparing a dedicated, hardware-based commercial tool against a open source, free-to-use, software-based tool. While they conclude that the commercial solution is faster at presenting results, they did not assess the efficiency in their findings. Moreover, it was a small comparison of just two tools.

In [11], the author questioned the performance of vulnerability scanning tools as a method to remedy the security issues these tools identify. He concludes that manual effort will always be needed to reach a complete accuracy. Moreover, the author concludes that the remediation guidelines outputted by the tools is very cumbersome to address.

In our work, we focused on the use of larger spectrum vulnerability scanning tools as these would require less time and resources to implement, while still being able to detect web application vulnerabilities. Moreover, the research works that compare vulnerability scanning tools have not been revisited recently.

3 Experimentation and Setup

A Vulnerability Scanner is a standalone application or program using a Graphical User Interface (GUI) or a Command Line Interface (CLI) with procedures to detect vulnerabilities and exploits in a given machine that is being analysed. These procedures and their effectiveness depends on multiple factors such as Operating System (OS), installed programs, existing services, their versions and configurations. Thus, the scanners rely on signatures of known vulnerabilities and exploits and either maintain them in a local database that maybe updated online or require a set of detection plugins or scripts that must be installed before scanning.

Given the research questions Q1 and Q2, the selected set of tools to analyse was narrowed to the following: OpenVAS [1], Nessus [18], Nexpose [17], and Tsunami [6]. Tsunami has been made available on GitHub in June of 2020. To answer all the three questions, a test-bed was designed and setup and the features of the selected scanners were compared.

3.1 Scanners Technical Features

Table 1 presents the selected vulnerability scanners and their main properties regarding their license, the availability of their source code, their mode of operation and the update process of their vulnerability lists. All selected vulnerability scanners are free-to-use. Nessus Essentials is free for personal use but limits scanning to 16 different IPs. Nexpose offers a 1 year trial, after which turns into a paid tool. Nexpose was included in the current analysis to detect if there is a significant difference between free-to-use and paid scanners. OpenVAS and Tsunami provide their versions as open source. OpenVAS, Nessus and Nexpose use a GUI while Tsunami operates in the CLI environment. Regarding the update process, OpenVAS, Nessus and Nexpose have a local database with the vulnerabilities signatures that are updated online, while Tsunami uses detection plugins.

Table 1. Selected vulnerability scanners

	License and source code			Operation		Vulnerabilities list update		
	Free to use	Trial period	Open source	GUI	CLI	Local DB	Online DB	Plugins or scripts
OpenVAS	x		x	x		x	x	
Nessus	(x)			x		x	x	
Nexpose	(x)	x		x		x	x	
Tsunami	x		x		x			x

The output of the selected vulnerability scanners is a PDF file with a non-standard organisation containing a set of potential vulnerabilities identified by a Common Vulnerabilities and Exposures (CVE) identification. The list of these vulnerabilities and their CVE ID is maintained publicly [7]. Each CVE record comprises the identification number, a description, and at least one public reference. These CVE records are sent to National Vulnerability Database (NVD) that extends their classification with additional information, severity scores and impact ratings. The severity scores are expressed by Common Vulnerability Scoring System (CVSS), an open framework used for communicating the characteristics and severity of vulnerabilities. The score is obtained by using three metric groups: Base, Temporal, and Environmental. The Base metrics produce a score ranging from 0 to 10, which can then be modified by the scoring of the Temporal and of the Environmental metrics.

3.2 Test-Bed Design and Setup

A test-bed was setup in order to test and evaluate all the vulnerability scanning tools identified in Sect. 3.1. The test-bed topology is depicted in Fig. 1 and comprises multiple virtual machines hosted on a laptop with an Intel core i7-4710HQ CPU @ 2.50 GHz processor, 12 GB of RAM, a 256 GB SSD, running the Windows 10 64bit OS. The use of virtualisation was selected in order to produce comparable results. The VirtualBox 6.1 was the adopted virtualisation solution. Five virtual machines were deployed, one to act as the scanner, and the remaining 4 to act as targets. Kali Linux was selected due to the simple installation process of the required tools for scanning. In order to minimise impacts of the installation of the tools, after the initial setup of the Kali Linux OS, a snapshot was taken and all tools were installed over that initial snapshot. This was made to maintain the same exact configuration on the system, prior to each tool installation. While the tests were executed, the target virtual hosts were disconnected from the Internet.

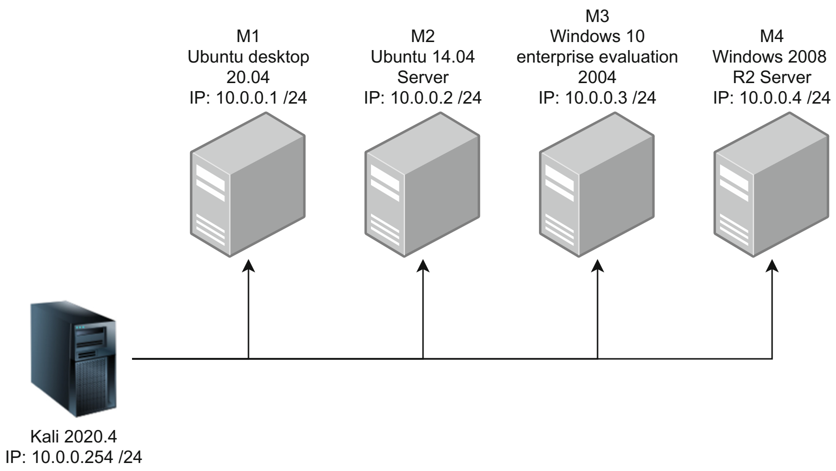


Fig. 1. Test-bed topology

The targets were selected in order to be as diverse as possible. Of the 4 virtual machines, two were Linux-based and two were Windows-based. Each set of two machines per platform were selected to represent a client version and a server version of each platform. Android targets were also considered at the beginning but, because the first tests showed that, due to strict firewall configurations, no result were reported by the selected tools, Android targets were dropped. Thus, the set of target virtual machines comprised:

– a Ubuntu Desktop 20.04 (as M1);
– a Ubuntu Server 14.04 (as M2);
– a Windows 10 Enterprise (as M3);
– a Windows 2008 R2 Server (as M4).

Of note is the fact that the M1 and M3 targets are standard installations of the respective OS, whereas M2 and M4 were deployed using metasploitable, version 3[2]. The metasploitable virtual machines are ones that are specifically setup with older software in order to have a large number of security vulnerabilities. As our focus was also on assessing the detection capabilities of the scanning tools, the later virtual machines were considered as the most relevant to test. These were tested without modifications or configurations, aside from disabling the MySQL server of M2 and enabling ping replies on M4. MySQL was disabled because to the excessive time taken by the Tsunami password brute-forcing with

[2] https://github.com/rapid7/metasploitable3.

Ncrack. Ping was enabled to ease the use of scanning tools that first checked the targets liveliness with a ping request.

A bash script was developed in order to monitor and record the execution time (duration), RAM memory and CPU usage on the Kali virtual machine, in a systematic way. For the network usage monitoring, on the same Kali virtual machine, and whenever a vulnerability scan was started, the **tcpdump** command was executed with arguments to identify the packets sent by the Kali virtual machine to the target machine, i.e. using 10.0.0.154 as the source IP and 10.0.0.X as the destination IP, where the “X” is the IP address of each target presented in Fig. 1.

4 Results and Analysis

Using the topology shown in Fig. 1, four vulnerability scanning tasks were executed per each of the four target machines, and per each of the four vulnerability scanners. In total, 64 vulnerability scans were conducted. Average and standard deviation values were obtained per each scanner.

The preliminary results shown that Tsunami was the fastest, using the least resources. Upon further evaluation, we came to the conclusion that, currently, the Tsunami tool does not contain enough vulnerability detection plugins and because of this, it detects almost no vulnerabilities and requires low resources to do so. This reasoning motivated us to not include Tsunami in the figures of this section.

Figure 2 shows the average duration of the performed scan tasks. Here, one can observe that the standard targets (M1 and M3) are scanned fastest by all tools due to having the least vulnerabilities and the least services available through the network. On the other hand, M2 and M4, being metasploitable-based targets, took the most time to scan. Other observation that can be made is that, of the three shown, Nessus was the fastest one and OpenVAS was the slowest one. OpenVAS took almost 5 times more to scan M4 when compared to the other tools.

Figure 3 a) shows the average network usage in terms of the number of packets, per second, sent by the Kali machine to the target machines. All vulnerability scanning tools report more network usage when scanning the M2 target, which runs a metasploitable Ubuntu Server. This is expected as this target is the one with the most services available through the network. When comparing tools, OpenVAS is the tool that uses more network resources, followed by the Nessus tool.

Figure 3 b) shows the average CPU used by the Kali machine during execution of the different tools. The tool that uses most CPU is the OpenVAS. This was expected as this tool also used more network resources and took the most time to complete. Nonetheless, the overall CPU usage of all tools is very low, maxing below 3,5%. Worthy of note, and because of being so low average values, is the fact that this resulted was the one that has shown the greater standard deviation.

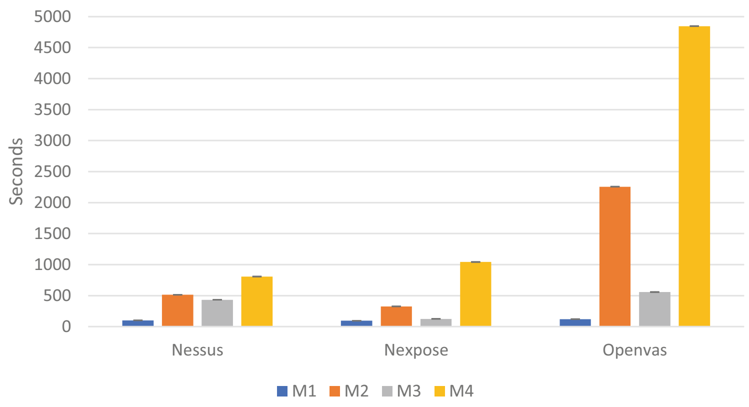


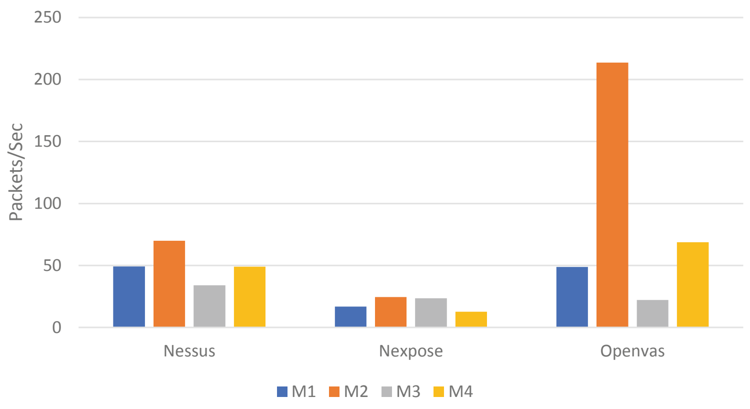
Fig. 2. Scan duration in seconds

Figure 3 c) shows the average memory used by the Kali machine during execution of the different tools. One conclusion that can be made is that all tools use a similar amount of memory independently of the scanned target. The tool that requires the most amount of memory is Nexpose, using almost four times the memory needed by the other two tools.

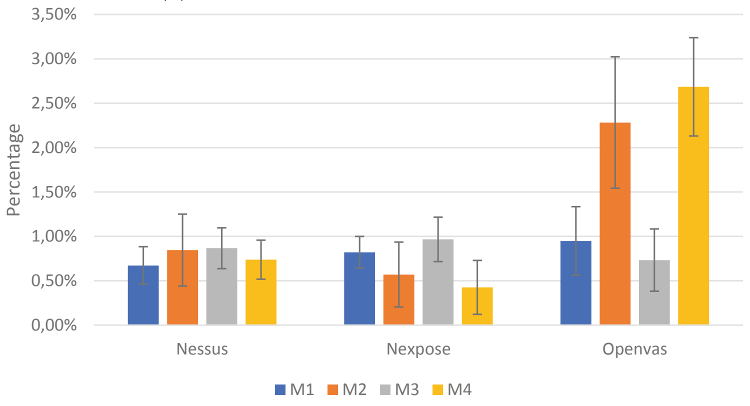
Table 2. Vulnerability identification results for M2

Vulnerability	CVSS	Nessus	Nexpose	OpenVAS
CVE-2010-1574	10 (v2)		FP	
CVE-2015-3306	10 (v2)	TP		TP
CVE-2015-5377	9.8			FP
CVE-2017-3167	9.8	FP	FP	
CVE-2017-3169	9.8		FP	
CVE-2017-7679	9.8		FP	
CVE-2018-1312	9.8		TP	
CVE-2018-5337	9.8			FP
CVE-2018-5341	9.8			FP
CVE-2019-12815	9.8		TP	
CVE-2017-9788	9.1		FP	
CVE-2016-5387	8.1		TP	
CVE-2017-15715	8.1		TP	

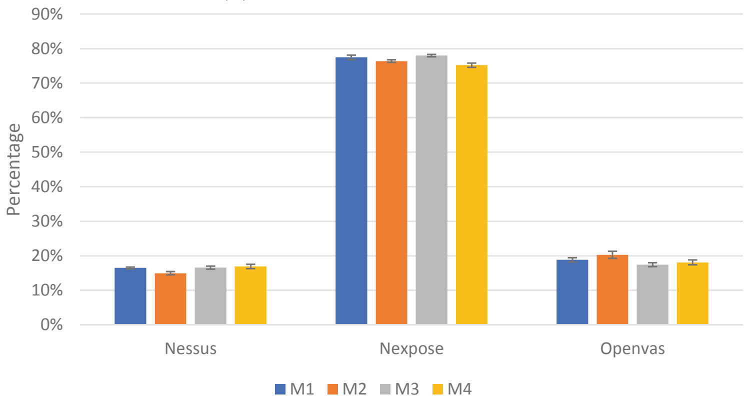
Tables 2 and 3 show the vulnerability identification results achieved by the different tools. In this tables, only vulnerabilities with an assigned CVE identification were considered. Vulnerabilities with an assigned CVE identification are published online and known by all vulnerability scanner tools, thus becoming a ground truth to which results of others tools can be compared. A list comprising all vulnerabilities reported by all tools, separated by target (M2 and M4), was



(a) Network usage in packets per second



(b) CPU usage in percentage



(c) RAM usage in percentage

Fig. 3. Network, CPU and RAM usage

compiled. The real presence of each vulnerability was then manually confirmed. In order to avoid such manual verification to become cumbersome, the full list of vulnerabilities was reduced to the ones that presented a score above 7.5, based on CVSS in version 3, plus the ones that presented a maximum score of 10, independently of the CVSS version.

Table 3. Vulnerability identification results for M4

Vulnerability	CVSS	Nessus	Nexpose	OpenVAS
CVE-2010-0219	10 (v2)			TP
CVE-2010-1574	10 (v2)		FP	
CVE-2012-2688	10 (v2)	TP		
CVE-2015-1635	10 (v2)		TP	TP
CVE-2017-7213	10 (v2)			TP
CVE-2015-5377	9.8			FP
CVE-2015-8249	9.8			TP
CVE-2017-11346	9.8			TP
CVE-2017-3167	9.8	FP	FP	
CVE-2017-3169	9.8	TP		
CVE-2017-7668	9.8	TP		
CVE-2017-7679	9.8	TP		
CVE-2018-5337	9.8			FP
CVE-2018-5338	9.8			TP
CVE-2018-5339	9.8			TP
CVE-2018-5341	9.8			FP
CVE-2020-10189	9.8	TP		
CVE-2017-5648	9.1			TP
CVE-2017-9788	9.1	TP		
CVE-2016-10012	7.8			TP

In order to better evaluate the vulnerability scanning tools, both accuracy and precision of the detected vulnerabilities was analysed. For this specific analysis, only M2 and M4 related results were considered. Reason being that these were the ones that had multiple identifiable vulnerabilities. The reader should recall that M1 and M3 are standard, recent and fully updated installations of Ubuntu and Windows 10, respectively.

Equations 1 and 2 were used to calculate accuracy and precision, respectively. These equations consider the number of True Positives (TP), False Positives (FP) and False Negatives (FN). TP being the number of vulnerabilities identified that are really present in the target. FP being the number of vulnerabilities identified that are not present in the target. FN being the number of vulnerabilities that are present in the target but not identified.

$$\text{Accuracy (\%)} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} (\times 100) \qquad (1)$$

$$\text{Precision (\%)} = \frac{\text{TP}}{\text{TP} + \text{FP}}(\times 100) \tag{2}$$

In a more straightforward way, with Eq. 1 we expect to evaluate if a tool is capable of detecting all available vulnerabilities within a target. i.e. its accuracy. With Eq. 2 we expect to evaluate if a tool only detects existing vulnerabilities and not false ones, i.e. its precision. The results shown in Fig. 4 resulted from calculating these equations from the data of the vulnerability identification results shown in Tables 2 and 3. From these results, the overall obtained accuracy is at most 50% for the case of the M4 scan with OpenVAS, this means that multiple vulnerabilities were not detected by all tools. In terms of accuracy we can see that OpenVAS and Nessus performed better for M4, a Windows machine, while Nexpose was more accurate for M2, a Ubuntu machine. In terms of precision, the overall better performing tools was OpenVAS with 100% accuracy for M2, and almost 80% for M4. The least precise tool was Nexpose, with 50% accuracy for both M2 and M4.

After evaluating the performance, resource usage, accuracy and precision of the selected vulnerability scanning tools, conclusions regarding the three questions listed in Sect. 1 can be drawn.

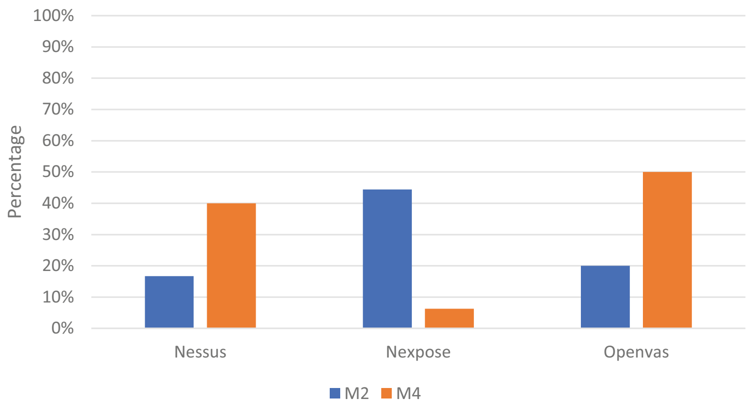
– Q1: What is the most efficient, free-to-use, vulnerability scanning tool currently available?

The correct answer to Q1 seems to be that there is no one tool that can be classified as the best at all evaluated criteria. For instance, while OpenVAS is the tool that uses more CPU and network resources and takes the most time, it is also the one that uses less memory and has a better overall precision. Nonetheless, in terms of accuracy it is better when scanning Windows-based targets, and not so good when scanning Linux-based targets. Nexpose, for instance, has an average precision of 50% for both Linux and Windows-based targets but, when analysing its accuracy, the results show very poor overall results.

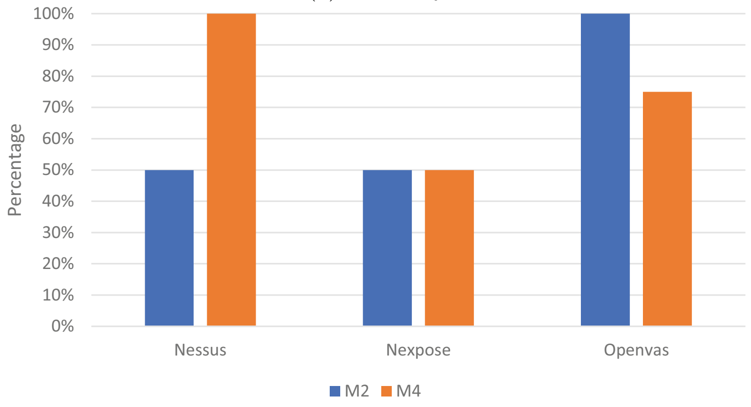
– Q2: How does Tsunami compare to similar tools currently available?

In its present state, Tsunami cannot be considered as a candidate substitute to other tools such as Nessus, Nexpose or OpenVAS. The key reason being the currently lack of openly available detection plugins. Tsunami architecture is plugin oriented, where each plugin will detect the presence of a specific vulnerability. When the authors stated this work, the number of plugins available was almost nonexistent, meaning that Tsunami was unable to detect any vulnerability that were present in the targets. The author believe that, in the future, Tsunami may become a relevant candidate if its authors release a number of detection plugins comparable to the remaining tools.

– Q3: Is Tsunami well suited to be used by SME's?



(a) Accuracy



(b) Precision

Fig. 4. Comparison of detection capabilities

The short answer to this question is no, at the moment it can not. Despite being open source and free-to-use, their current lack of detection plugins plus its mode of operation, makes it unsuitable for use in SMEs that do not have human resources capable of developing their own plugins for Tsunami. Moreover, the way the results are reported by Tsunami (JSON format) make it best suited for use in automatic assessments of a development pipeline in a product development life cycle.

5 Conclusions

Organisations may benefit from a systematic and periodic vulnerability assessment using free-to-use scanning tools. Using automated vulnerability scanning tools also reduces the required human, technical and financial resources when compared to manual penetration testing. With the release of Tsunami, yet another free-to-use vulnerability scanning tool, the authors decided to perform an updated evaluation of the existing similar tools. The evaluation considered both the performance of the tools, but also their accuracy and precision.

The obtained results show that OpenVAS was the tool that achieved the best overall precision and the best accuracy when scanning Windows-based systems. Nexpose was the tool that achieved the best accuracy when scanning Linux-based systems. In terms of CPU, memory and network usage, the results differ greatly from tool to tool but a common trait, of requiring more resources to scan systems with more vulnerabilities, was also identified. The authors also concluded that Tsunami, by having a very small detection capabilities, is still far from the detection capabilities of the other free-to-use tools.

The manual confirmation of vulnerabilities reported by all tools was focused on the critical ones, with a CVSS above 7.5. As future work, the authors will proceed with the manual confirmation of all vulnerabilities reported by all tools to have a better understanding of both the accuracy and the precision of the evaluated tools.

References

1. Aksu, M.U., Altuncu, E., Bicakci, K.: A first look at the usability of openvas vulnerability scanner. In: Workshop on Usable Security (USEC) 2019. NDSS (2019)
2. Ali, A.B.M., Abdullah, M.S., Shakhatreh, A.Y.I., Alostad, J.: SQL-injection vulnerability scanning tool for automatic creation of SQL-injection attacks. Procedia Comput. Sci. **3**, 453–458 (2011)
3. Amankwah, R., Chen, J., Kudjo, P.K., Towey, D.: An empirical comparison of commercial and open-source web vulnerability scanners. Softw. Pract. Exp. **50**(9), 1842–1857 (2020)
4. Austin, A., Williams, L.: One technique is not enough: a comparison of vulnerability discovery techniques. In: 2011 International Symposium on Empirical Software Engineering and Measurement, pp. 97–106. IEEE (2011)
5. Chimmanee, S., Veeraprasit, T., SriphREw, K., Hemanidhi, A.: A performance comparison of vulnerability detection between netclarity auditor and open source nessus. In: Proceeding of the 3rd European Conference of Communications (ECCOM 2012), pp. 280–285 (2012)
6. Cimpanu, C.: Google open sources Tsunami vulnerability scanner. ZDNet, July 2020. https://www.zdnet.com/article/google-open-sources-tsunami-vulnerability-scanner/
7. The MITRE Corporation: Common Vulnerabilities and Exposures (CVE). https://cve.mitre.org/. Accessed 10 Feb 2020
8. Daud, N.I., Bakar, K.A.A., Hasan, M.S.M.: A case study on web application vulnerability scanning tools. In: 2014 Science and Information Conference, pp. 595–600. IEEE (2014)
9. Doupé, A., Cova, M., Vigna, G.: Why Johnny can't Pentest: an analysis of black-box web vulnerability scanners. In: Kreibich, C., Jahnke, M. (eds.) DIMVA 2010. LNCS, vol. 6201, pp. 111–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14215-4_7

10. Fonseca, J., Vieira, M., Madeira, H.: Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks. In: 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), pp. 365–372. IEEE (2007)
11. Holm, H.: Performance of automated network vulnerability scanning at remediating security issues. Comput. Secur. **31**(2), 164–175 (2012)
12. Holm, H., Sommestad, T., Almroth, J., Persson, M.: A quantitative evaluation of vulnerability scanning. Inf. Manag. Comput. Secur. **19**(4), 231–247 (2011)
13. Kals, S., Kirda, E., Kruegel, C., Jovanovic, N.: Secubat: a web vulnerability scanner. In: Proceedings of the 15th International Conference on World Wide Web, pp. 247–256 (2006)
14. Kushe, R.: Comparative study of vulnerability scanning tools: Nessus vs Retina. Secur. Future **1**(2), 69–71 (2017)
15. Mburano, B., Si, W.: Evaluation of web vulnerability scanners based on owasp benchmark. In: 2018 26th International Conference on Systems Engineering (ICSEng), pp. 1–6. IEEE (2018)
16. Qianqian, W., Xiangjun, L.: Research and design on web application vulnerability scanning service. In: 2014 IEEE 5th International Conference on Software Engineering and Service Science, pp. 671–674. IEEE (2014)
17. Rapid7: Free Nexpose Community 1-Year Trial. https://www.rapid7.com/info/nexpose-community
18. Tenable: Nessus Vulnerability Assessment Tool. https://www.tenable.com/products/nessus. Accessed 10 Feb 2020
19. Wang, Y., Yang, J.: Ethical hacking and network defense: choose your best network vulnerability scanning tool. In: 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 110–113 (2017)
20. Welberg, S.: Vulnerability management tools for cots software-a comparison. Hg. v. University of Twente (2008). https://research.utwente.nl/files/5101819/Vulnerability_management_tools_for_COTS_software_-_a_comparison_v2.1.pdf