A Comparative Evaluation of Automated Vulnerability Scans versus Manual Penetration Tests on False-negative Errors

Saed Alavi, Niklas Bessler, Michael Massoth
Department of Computer Science
Hochschule Darmstadt (h_da) — University of Applied Sciences Darmstadt,
and Center for Research in Security and Privacy (CRISP), Darmstadt, Germany
E-mail: {saed.alavi,niklas.bessler}@stud.h-da.de, michael.massoth@h-da.de

***Abstract*—Security analysis can be done through different types of methods, which include manual penetration testing and automated vulnerability scans. These two different approaches are often confused and believed to result in the same value. To evaluate this, we have build a lab with several prepared vulnerabilities to simulate a typical small and medium-sized enterprise. Then, we performed a real penetration test on the lab, and a vulnerability scan as well, and then compared the results. Our conclusion shows, that the results obtained through both types of security analysis are highly distinct. They differ in time expenditure and false-positive rate. Most importantly, we have seen a remarkable higher false-negative rate in the vulnerability scan, which suggests that automated methods cannot replace manual penetration testing. However, the combination of both methods is a conceivable approach.**

Keywords–Security analysis; penetration test; vulnerability scan.

I. INTRODUCTION

Information technology (IT) security has become more and more important, due to the increasing threat of cybercrime. Therefore the demand for security analysis of information technology infrastructure is constantly growing. The purpose of such a security analysis is to identify threats, estimate likelihood and potential consequences, which makes it possible to determine a risk value eventually. Results are achieved through different methods of security testing. However, these security tests can vary significantly in cost, scope, informative value and other characteristics. In this paper, we distinguish between penetration tests (colloquially known as pentest) and vulnerability scans (abbreviated vuln scan).

The goal of this research is to assess which method of security analysis should be considered to be better relating to false-positives errors as well as false-negative errors. In this paper, we focus on the false-negative rate. The reason for this is, that this type of error is more severe. To accomplish this research goal, we strictly separate the work in isolated work packages as summarized in this section. The overall experimental setup is described in detail in Section IV.

First, one of the authors builds a lab environment, which represents a typical and representative IT infrastructure of a small and medium enterprise. In addition, the computer systems are prepared with various vulnerabilities. Following this, a typical penetration test will be performed by another author, who is in the role of a penetration tester. This happens without any knowledge of the previous work package (preparation of the lab environment). Using this approach, we want to ensure that all vulnerabilities are revealed in a proper way through real pentesting and results are not influenced in any way by prior knowledge. Then, we use two popular vulnerability scanners to generate automated vulnerability reports. We make use of the proprietary software Nessus and the free software framework OpenVAS. In a final step, we validate both manually and automatically generated reports and determine error rates, where we finally consider the knowledge, of building a self-made lab environment, which has been totally absent up to this point.

This paper is organized as follows: Section II gives an overview of the relevant terminology. Section III summarize the research achievement of previous scientific publications on this topic. In Section IV we explain all tasks of the particular work packages in their chronological order. This includes how we build the lab environment, our methodology used in the penetration test and how we configured the two vulnerability scanners. Section V provides our results and analysis. Section VI summarizes our conclusion. Finally, Section VII talks about possible future work.

II. BACKGROUND

In this section, we introduce the terminology. We also define several terms, due to lack of a standardized definitions. Furthermore, we describe the two different error types, explain their meaning in the context of a security analysis and point out their relevance.

A. Security analysis

Security analysis refers to the process of identifying security related issues and determining their estimate of risk as well. The process of looking for vulnerabilities can be either in technical manner, including penetration tests and vulnerability scans. Or it is in organizational manner, e.g., business process analysis or enter into a dialog with employees. We only cover the technical part. Furthermore, all findings shall be assessed in their risk value and include recommendations for appropriate measures. As an example periodic security assessments are required in several security standards such as the Payment Card Industry Data Security Standard (PCI DSS) [1]. It requires quarterly external vulnerability scans and external penetration testing at least annually.

B. Risk

A risk is always associated with potential harm to an infrastructure, respectively to an whole organization. It consists of two factors: the potential impact and the probability of occurrence. Both values should be understood as estimates, which are represented in a quantitative (e.g., amount of money) or qualitative (e.g., low, medium, high) form. Figure 1 shows how we categorized the risk values for the penetration test.

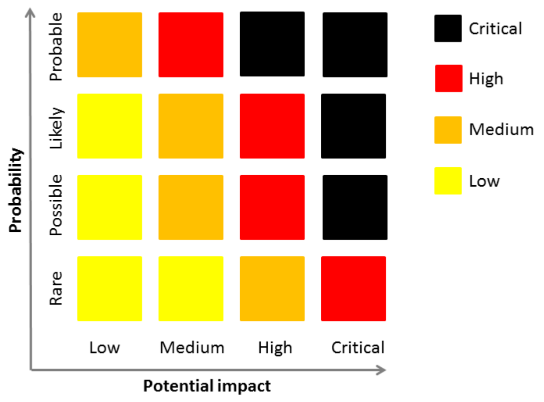

Figure 1. Risk matrix.

C. Penetration test

Penetration testing describes the simulation of a genuine cyber attack. Its goal is to identify vulnerabilities, thereby reducing security risks. This is done using a wide range of attack vectors to cover as much potential vulnerable points as possible. Components of such a test includes especially extensive manual testing and automated tool sets for at least common vulnerabilities. In contrast to a real attack, pentests differentiate in motivation, time expense and legality.

D. Vulnerability scan

A vulnerability scanner is a computer program, which identifies known vulnerabilities in an automated way. Such Vulnerability scans can be performed either unauthenticated or authenticated. Running the scan with corresponding credentials often leads to lower error rates and results of higher quality. Typically a vuln scan is very fast and do not require much technical knowledge, except in interpreting its results. It is a common practice to perform a vulnerability scan and claim it as pentest, although a pentest is actually ordered.

E. Informative base of testing

Black-box testing refers to testing without knowing the internal structure of a system (correlates to the knowledge of external individuals). The opposite is known as White-box test (correlates to the knowledge of (ex-)employees). Something in between we call Grey-box test. In general the informative base determines the knowledge of an attacker, and therefore determine the attack vectors the attacker would be capable of.

F. False-positive error

A false-positive error is a result, that wrongly indicates the presence of a given condition, where the condition is actually absent. It is colloquially known as false alarm. For example, a security report claims a deprecated software version as vulnerable, however a newer version is installed, which is not exploitable anymore. We declare two subtypes of false-positives, which are described more detailed in Section V. This error type is in general not severe, but can slow down the progress of testing.

G. False-negative error

A false-negative error is a result, that wrongly indicates the absence of a given condition, where the condition is actually present. This type of error basically refers to an existing vulnerability, which is not found through testing. False-negative errors are very critical within the context of security analysis, because they result in unreported and thus unfixed vulnerabilities.

III. Related Work

Some research achievement already has been accomplished when it comes to comparing manual and automated methods for security analysis. Therefore, we want to give an overview of the current state of research and show other scientific publications, which address the similar research question.

Austin et al. [2] conducted a case study on two Electronic Health Records (EHR) systems with the objective to compare different approaches of security analysis. In contrast to our research they have chosen two open source EHR systems as subject for study and rather performed a White-box test on computer software, whereas we performed a Grey-box penetration test on a whole IT-infrastructure. In their publication, they distinguish between systematic and exploratory manual penetration testing, static analysis and automated penetration testing. The authors claim, that no single technique discovered every type of vulnerability and almost no individual vulnerabilities were found by more than one type of security analysis. Furthermore, they conclude, that systematic penetration testing was the most effective way. At the same time, static analysis and automated pentesting shouldn't be relied on, because these two methods result in a higher rate of undiscovered vulnerabilities (false-negatives). However, automated penetration testing was the most efficient detection technique, when it comes to found vulnerabilities per hour.

Holm [3] investigated in his research how many vulnerabilities would be remediated if one would follow the recommendations provided by an automated vuln scan. For this purpose both authenticated and unauthenticated scans of seven different network vulnerability scanners were evaluated. The author concludes that "a vulnerability scanner is a usable security assessment tool, given that credentials are available for the systems in the network". However, they do not find all vulnerabilities and likewise do not provide a remediation guideline for every security issue. Furthermore, his research findings suggests that the false-positive rate is relatively low, especially when credentials are given to the scanner. Although false-positives increases with a higher remediation- and detection rate.

Stefinko et al. [4] compared several aspects of manual and automated penetration testing. The comparison was primarily realized in a theoretical way. However, some practical examples are given. The authors come to the conclusion that an automated approach can be less time consuming and highly benefits from reproducibility. Although manual methods are still better, automation can lead to a significant improvement, e.g. by using scripts.

IV. Approach

In this section, we describe the overall experimental setup in detail. Before we assessed any results, we finished the three isolated work packages, which were completed by different authors. The work packages are done in the chronologically order as in this section.

A. Lab environment

First, a typical and representative lab environment was build for the experiment. Its conceptual architecture is shown in Figure 2.

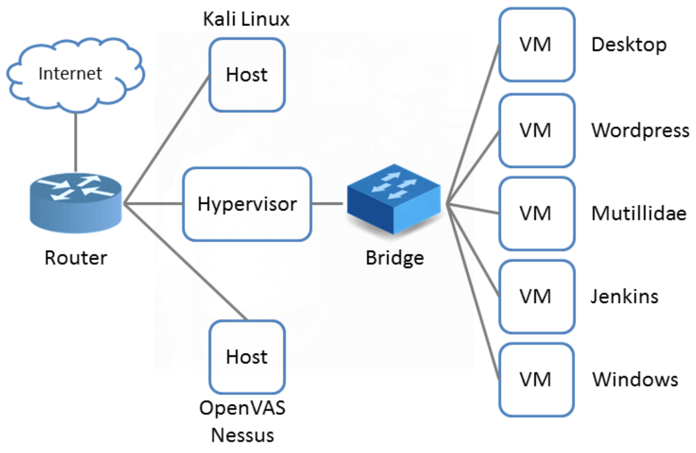

Figure 2. Setup of the Lab Environment.

The lab consists, among others, of two hosts, which main purpose is to perform security analysis. Even in the conceptional architecture of the lab we have already paid attention to the strict separation of the hosts used for penetration testing and automated vuln scans. Therefore, the two techniques are evaluated on independent and isolated hosts. Both hosts have access to the Internet to make research possible on the pentesters side and an easier configuration on the vuln scanners side.

Furthermore we setup one machine as hypervisor, based on VirtualBox. This can be seen as core of the lab running all five guest virtual machines (VM). To ensure none of the virtual machines are able to automatically update themselves, no Internet access is configured to prevent potential auto update mechanisms.

All virtual machines are based on Ubuntu 18.04, except the virtual machine named Desktop, which is running an intendedly outdated version of Ubuntu Linux. One VM is installed with Windows 7. The installed applications are popular and mostly open-source software. They include the automation server Jenkins, the content management system WordPress, the deliberately vulnerable web-application OWASP Mutillidae 2, the web development tool XAMPP and File Transfer Protocol (FTP) services installed on the VM Windows and last but not least a typical linux based desktop computer with lots of outdated components. While Mutillidae is meant to be a vulnerable web application, the other VMs are prepared with either outdated applications, weak credentials or bad misconfiguration. All prepared vulnerabilities are summarized in Table 1. Our selection includes the ten most critical security risks to web applications as postulated in ”OWAS Top 10 - 2017” [5]. Additionally, we covered every item on the check-list proposed in the penetration test guideline ”Ein Praxis-Leitfaden für IS-Penetrationstests” by the German Federal Office for Information Security (in German Bundesamt für Sicherheit in der Informationstechnik) [6].

The number of detected security issues, which are mentioned in Table 1, will be later considered as criterion for false-negatives. The reason for this is that the amount of false-negatives is potentially uncountable, so we decided to use this scale as an objective measurable criterion. Deeper explanations follows in the result section.

B. Penetration test of the lab

The second work package includes comprehensive penetration testing of the lab. It is designed as it would be a genuine pentest in the real world. In order to do this, one of the authors, who has not any knowledge of the lab environment, obtains an Internet Protocol address (IP address) of the bridged network to access the virtual machines. From this point he is permitted to perform pentests on the lab.

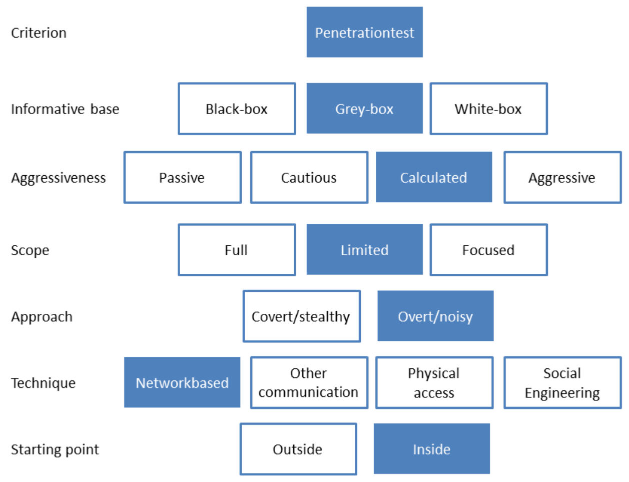

Figure 3. Classification of the penetration test. Criteria based on [7].

Before testing starts, we define the scope of the penetration test. As it is common practice we also clarified underlying conditions. So the penetration tester has permission to test the whole infrastructure for five work days which is the equivalent of 40 hours. Aggressive methods like distributed denial-of-service (DDoS) attacks are forbidden and no value is put to a stealthy approach. Furthermore we let the pentester know that no IPv6 addresses have to be checked, because it is impossible to scan an IPv6 network in an appropriate time. Furthermore we communicated that no User Datagram Protocol (UDP)-based network ports are open, because scanning such ports is very time consuming. Figure 3 illustrates how we would classify the penetration test. To guarantee the highest possible reproducibility the author in the role of the penetration tester strictly followed the methodology proposed by the Federal Office for Information Security (BSI) [7]. By following this guide every penetration tester should find at least the same number of vulnerabilities.

C. Vulnerability scan of the lab

Last but not least, we performed the vulnerability scans. In order to do that, we decided to use two popular scanners - one proprietary (Nessus) and one open source tool (OpenVAS). According to the developer Nessus is the most used network assessment tool. It has over 100,000 plugins that contain vulnerability information, a set of remediation actions and algorithms to prove the presence of security issues. OpenVAS is a free software framework of services and tools recommended by the BSI. It offers vulnerability- scanning, assessment and management. It contains over 50,000 Network Vulnerability Tests (NVTs) that are equivalent to the Nessus Plugins.

We performed Address Resolution Protocol (ARP)-, Transmission Control Protocol (TCP)- and Internet Control Message Protocol (ICMP) tests on the default port range of Nessus, that includes 4,790 commonly used ports, including all open ports on our target network. Additionally the following settings were enabled: Probe all ports to find services, perform thorough tests, web applications - and the ”enable safe checks” setting was turned off. All available Nessus Plugins were activated.

The OpenVAS default port range includes 4,481 ports and was performed with ICMP-, TCP-ACK Service- & ARP Ping alive tests. The full and very deep scan setting was used, while the Safe Checks” setting was disabled. No UDP ports were scanned, because they are very time consuming and there are no open UDP ports in the test environment as communicated before. Both scanners allow to use credentials for authenticated scans. All virtual machines except of the Mutillidae host were scanned two times, authenticated and unauthenticated.

V. RESULTS

In this section, we want to display our results. We listed the prepared vulnerabilities of the lab and the findings of the pentest and vuln scanners. Furthermore we describe the meaningfulness of the different security analysis methods.

Figure 4 shows the concrete number of findings revealed by the pentest and the vuln scanners. The penetration test includes a total of 73 findings in this work, while we ignored findings with the lowest criticality ”Info/Log” in general and also ignored findings by OpenVAS with a quality of detection less than 70%, which is the recommended threshold value. The automated analysis lists 1.5 to 3-times more findings than the manual test. Actually, concluding the automated methods have more findings is a fallacy. The explanation for this is that the pentest report is a sum up of all exploited findings, e.g. it is grouped in categories. The vuln scanners list every single vulnerability, that matches a database entry for the software version or configuration. Those findings can include false-positive errors, which are explained in more detail in Subsection B.

A. Report

The result of every security analysis usually ends up with a structured report. The main part of such a write-up includes the revealed vulnerabilities, their criticality, a proof of concept and recommendations for remedial actions. In this section, we will look at the differences of the reports. A penetration test report includes a comprehensible proof for every listed vulnerability. That could be either a few lines of code, a screenshot or for example a console output, that proofs the existence of a vulnerability, which makes it possible to reproduce the test. OpenVAS and Nessus both have their own way of output format for listed vulnerabilities, denoted as Vulnerability Detection Result (OpenVAS) and Plugin Output (Nessus). The major problem with the outputs of the vuln scanners is, that they are not reproducible, i.e. the output is just a version number of a running service. It is not distinguishable whether the vuln scanners actually exploited the vulnerability or not. Only for web services the scanners reported a comprehensibly and reproducible proof.

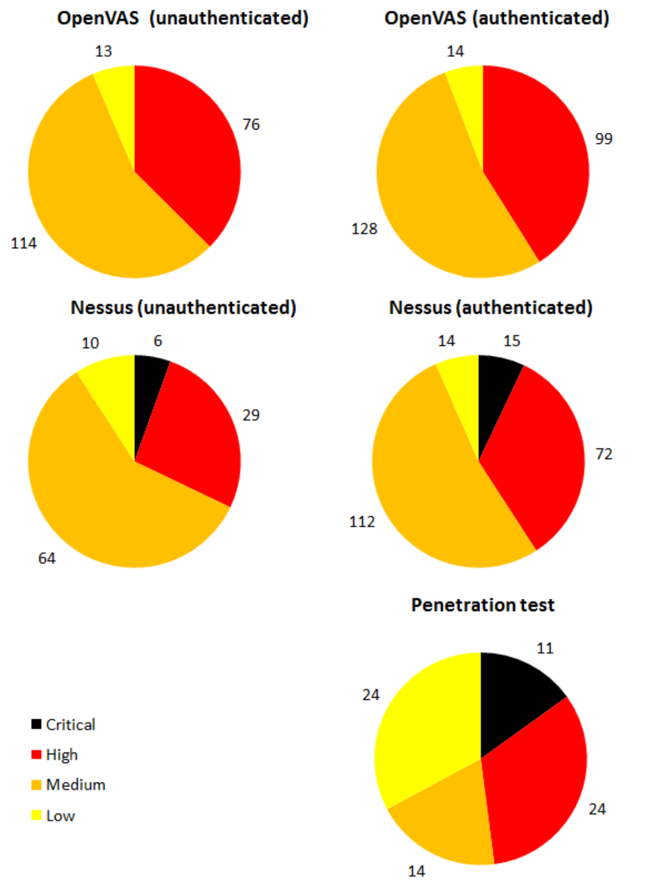

Figure 4. Overall vulnerabilities detected via different methods of security analysis.

A set of vulnerabilities may help the system administrator/network operator to get an overall overview of the tested systems. But it is also important to get recommendations for remedial actions, to fix detected security issues. The manual generated report as well as the automated generated report have recommendations for more than 99% of the findings. Since the pentest report lists a lower number of findings, it also has fewer recommendation treatments. Nevertheless, all the findings of the automated scans are also handled in the manual report.

To get an order for closing weak points, it is recommended to treat the most dangerous vulnerabilities as first. All security analysis evaluate the listings, denoted as criticality. Nessus and our pentest have four evaluation stages between low and critical, meanwhile OpenVAS has three, from low to high. The ISACA Implementation Guideline ISO/IEC 27001:2013 recommends an even number of criticality levels to prevent a more frequently ”landing in the middle” for decisions [8].

B. False-positives

As explained in Section II, false-positive errors indicate the presence of given conditions, that are not the case. For vuln scans there are two major kinds of false-positive errors. The first type is a displayed vulnerability that doesn't exist, actually. To classify such errors as true-positives, it is enough to exploit that vulnerability. The inversion in that case doesn't apply. If one cannot exploit that vulnerability it doesn't mean that it's non-existent. Usually it is associated with lots of efforts to check if the reported vulnerabilities are perhaps false alarms. To perform such a classification, you have to examine the affected lines of code to check if an exploiting is impossible, what requires access to the code (White-box testing).

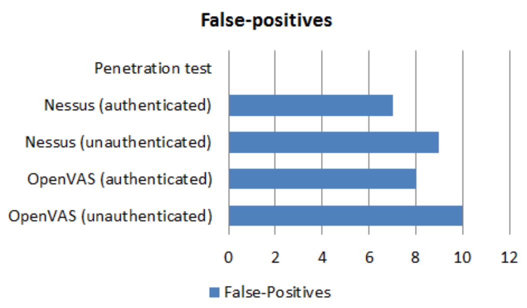

Figure 5. Number of Type II false-positive errors (less is better). Based on the totals findings cf. Figure 4.

The second type, we have identified, is a wrong information about the target systems. For example the report shows misconfiguration of a service, when the configuration is actually fine. To classify such vulnerabilities as true or false positives, you can easily check the affected configuration files. Penetrationtesters usually prove the existence of a security issue and therefore shouldn't be prone to any false positives. In this research, we considered only the second type of false-positive errors, that include incorrect information about the target system. Figure 5 shows the number of false-positives, the scanners have listed. Compared to the total number of findings, false-postive rate in this cases is less than 1%. The authenticated scan has a lower number of false-positives for both scanners, than without providing credentials. The reason is that the vulnerability scanners can access more detailed information. In comparison of other works the false-positive rate is unexpectedly low.

C. Time required

For security analysis the required time is always one major factor to choose between an automated or a manual security assessment. Vulnerability scanners should always have the same scanning time, if utilization factors like network, memory and central processing unit (CPU) are the same. Only the time for setting up varies, depending on the experience of the user. In our case it took approximately 30 minutes for configuring Nessus and approx. 45 minutes for OpenVAS as shown in Figure 6.

Usually, for a penetration test a fixed period of time is scheduled. As in Section II described the penetration test in this case was performed in 40 hours. This time includes preparation, analysis, exploitation and writing the final penetration test report. One huge advantage of the vulnerability scanners is the detection speed and that it provide other useful information about the target systems. Figure 6 shows, that the vuln scanners need only approximately 10% of the time a pentester needs for the full security analysis.

TABLE I. PREPARED VULNERABILITIES.
✓: TRUE-POSITIVES, ✗: FALSE-NEGATIVES

VM	*Vulnerability*	*Criticality*	*Nessus*	*OpenVAS*	*Pentest*
Jenkins	missing authentication for web application	high	✓|✓	✗|✗	✓
Jenkins	CMDi-Vulnerability	high	✗|✗	✓|✓	✓
Jenkins	file-permission misconfiguration	critical	✗|✗	✗|✗	✓
Jenkins	user-permission misconfiguration	high	✗|✗	✗|✗	✓
Desktop	outdated linux kernel	critical	✓|✓	✓|✓	✓
Desktop	weak user password	high	✗|✗	✗|✗	✓
Desktop	weak openSSH configuration	high	✗|✗	✗|✗	✓
Windows	outdated XAMPP	critical	✗|✗	✓|✓	✓
Windows	vulnerable free-sshd	critical	✗|✗	✗|✗	✓
Windows	phpMyAdmin missing authentication	high	✓|✓	✗|✗	✓
Windows	FTP-Anonymous user enabled	medium	✗|✗	✓|✓	✓
Windows	FTP-unencrypted data transfer	medium	✗|✗	✗|✗	✓
Wordpress	XSS vulnerable activity-log Plugin	high	✗|✗	✗|✗	✓
Wordpress	CMDi-Vulnerability	high	✗|✗	✗|✗	✓
Wordpress	missing HTTP-Only and Secure Flag	medium	✗|✗	✗|✗	✓
Wordpress	Wordpress-Admn with weak credentials	high	✗|✗	✗|✗	✓
Wordpress	misconfigured ssh private-key	critical	✗|✗	✗|✗	✓
Wordpress	sudo commands without password	critical	✗|✗	✗|✗	✓
Mutillidae	SQL-Injection/XSS Vulnerabilities	high	✓	✗	✓
Mutillidae	CMDi-Vulnerability	critical	✓	✗	✓
Mutillidae	missing HTTP-Only and Secure Flag	medium	✗	✗	✓
Mutillidae	bypass authentication via authentication token	high	✗	✗	✓
Mutillidae	unvalidated redirects and forwards	medium	✗	✗	✓
Mutillidae	CSRF-vulnerability	high	✓	✗	✓
Mutillidae	Insecure Direct Object References	medium	✓	✓	✓
General	missing banner	low	✗|✗	✗|✗	✓
General	exploit error routine for information leaks	medium	✗|✗	✗|✗	✓
General	no HTTPS	medium	✓|✓	✓|✓	✓
General	outdated Software	low to critical	✓|✓	✓|✓	✓

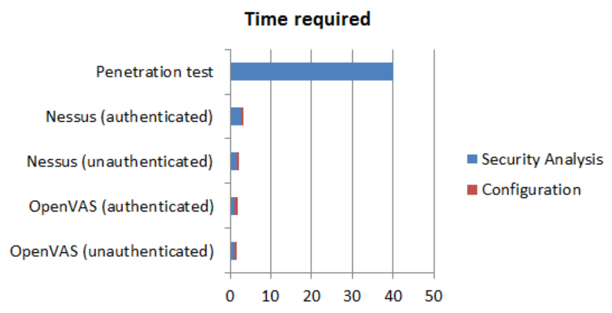

Figure 6. Required time in hours to perform a security analysis.

D. False-negatives

The most important criterion of a security analysis is to reveal vulnerabilities. Therefore every false-negative can lead to a very serious security problem. Nevertheless it is not possible to find all vulnerabilities in a system. To get a measurable rate of false-negative errors, we created the lab with prepared vulnerabilities as listed in Table 1. The table only shows the deliberately implemented vulnerabilities. The list was not completed by found true-positives that was not noted before the tests. The columns of the vulnerability scanners Nessus and OpenVAS are separated in authenticated (left of the pipe) and not authenticated (right of the pipe) results.

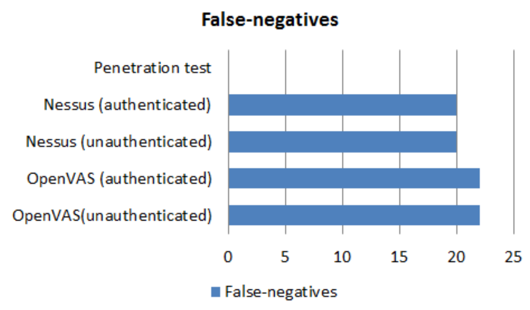

Figure 7. Number of false-negative errors (less is better). Based on the prepared vulnerabilities cf. Table 1.

The author who performed the penetration test found all of those vulnerabilities and had no false-negatives, using the OWASP-Testing guide and the methodology of the Federal Office for Information Security (BSI). Figure 7 shows that Nessus has 20 false-negative errors, that is equivalent to a rate of 69% and OpenVAS 22 errros, that matches 76%. Surprisingly, there is no difference whether the scans were given system credentials or not. The high false-negative shows, that if all vulnerabilities found by the scanners were fixed, the systems would still be prone for security issues.

VI. Conclusion

In general, automated vulnerability scanners detect more security issues than manual penentration testing viewed in a quantitative perspective. However, the results of automated methods significantly lack of quality and have less useful findings. Furthermore, vulnerability scanners do not always prove their concept how a security hole can be exploited. Fortunately, both methods show appropriate remediation recommendation in the reports. When it comes to false-positive errors, we only find a few errors, although we were not able to consider all potential false-positives, as described in the results section and future works. The most important finding of this research is the significant higher false-negative rate of the vulnerability scanners, i.e. many security holes were not detected. It is worth mentioning that automated methods are much faster in performing the security analysis. To sum up, both manual and automated methods can complement each other, but an automated scan cannot replace manual penetration testing these days. A conceivable approach could also be in combining both methods, which can get the best ratio of effort and results.

VII. Future Work

One aspect that would benefit from further research is a better approach to analyze false-positive errors. In our publication we only determined one specific type of this error, although we are fully aware that the false-positive rate is potentially much higher. The problem at this point is that investigation of such errors is not possible in an appropriate time or sometimes completely impossible. If one of the findings is exploitable we have a clear true-positive result, but this doesn't apply vice versa. If a reported security issue is not exploitable this does not prove the presence of a false-positive error. We have found no scientific publication that deals with this potential errors in a smart way.

Acknowledgment

The authors would like to thank the Center for Research in Security and Privacy (CRISP) in Darmstadt for their support and our good friends from the binsec GmbH for sharing best practices in security analysis.

References

[1] PCI Security Standards Council, LLC., "Payment Card Industry (PCI) Data Security Standard, v3.2.1," May 2018.

[2] A. Austin and L. Williams, "One Technique is Not Enough A Comparison of Vulnerability Discovery Techniques," 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM), Sep. 2011, doi: 10.1109/ESEM.2011.18.

[3] H. Holm, "Performance of automated network vulnerability scanning at remediating security issues," Computers & Security, vol. 31, Mar. 2012, pp. 164–175, doi: 10.1016/j.cose.2011.12.014.

[4] Y. Stefinko, A. Piskozub, and R. Banakh, "Manual and automated penetration testing. Benefits and drawbacks. Modern tendency," 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), Feb. 2016, doi: 10.1109/TCSET.2016.7452095.

[5] Open Web Application Security Project, "OWASP Top 10 2017," 2017, URL: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_(en).pdf.pdf 2018-06-15.

[6] Federal Office for Information Security (BSI), "Ein Praxis-Leitfaden für IS-Penetrationstests [A guideline for information system penetration tests]," 2016.

[7] Federal Office for Information Security (BSI) , "Study: A Penetration Testing Model," 2003.

[8] ISACA Germany Chapter e.V., "Implementation Guideline ISO/IEC 27001:2013," Apr. 2017, URL: https://www.isaca.de/de/Implementation Guideline ISO/IEC 27001%3A2013 2018-06-16.