International Journal of Innovative Solutions in Engineering
2025, Vol. 1 No. 1 (pages 1–9)
DOI: 10.47960/3029-3200.2025.1.1.1

Research Article

OPEN ACCESS

Achieving Successful Software Penetration Testing

Doris Musa[1], Ivan Markić[1]

[1]Faculty of Mechanical Engineering, Computing and Electrical Engineering University of Mostar, Bosnia and Herzegovina

Abstract

In today's digital environment, where society faces cyber threats, security is a key component of any organization. Penetration testing is crucial in detecting and eliminating security weaknesses before real attackers can use them. In reality, software penetration testing mimics real-world attacks on software applications to detect vulnerabilities, i.e., security vulnerabilities that real attackers could exploit. The testing requires careful planning, precise performance, and a thorough analysis of the results. This paper deals with successful penetration testing, starting with the basic steps of planning and defining the scope of the test itself. An overview of all tools and techniques and the importance of choosing the right tools for penetration testing are presented. The paper provides a comparative analysis of key tools for penetration testing, including practical guidelines for choosing appropriate methodologies and frameworks. Penetration testing requires a continuous approach as threats and vulnerabilities evolve with technology development. Through regular testing and updates of security policies, organizations can ensure the security of their own software systems.

Article history

Received: 20. 12. 2024.
Revised: 14. 01. 2025.
Accepted: 16. 01. 2025.

Keywords

Software Security,
Risk Assessment,
Penetration Testing,
Abuse Cases.

1 Introduction

Penetration testing is a technique for assessing the security of computer systems, networks, or applications based on simulating actual attacks. Penetration testing aims to detect vulnerabilities that attackers could exploit and provide the organization with insight into weaknesses that need to be eliminated before a real threat occurs. At the beginning of the testing process, it is necessary to clearly define the testing scope and goal, including identifying the components to be tested. Once the scope is set, the next step is to perform a test that includes different testing methods. The final penetration testing phase is the results' analysis phase, which consists of a detailed review and analysis of the collected data to identify potential security flaws. Reports that include recommendations for resolving discovered vulnerabilities are produced, which document the vulnerabilities found and provide guidance for improving security measures. Key stages of the testing process, including preparation, vulnerability identification, exploitation, and final reporting, were also discussed to ensure a thorough and systematic approach that allows for a detailed analysis of security vulnerabilities, as well as providing practical recommendations for improving the protection of the systems themselves [1], [3], [4], [5].

1.1 Vulnerabilities in the software

A vulnerability is a bug that an attacker can exploit. There are a multitude of vulnerabilities, and in the field of computer security, their classifications have been created. In computer systems, vulnerabilities range from local implementation errors through procedural errors to more serious ones at the design level. Vulnerabilities are often divided into two categories: implementation-level errors and design-level flaws. Malicious attackers don't care if a bug or a flaw causes a flaw in the software, although bugs can be exploited more easily. The most challenging thing

Contact Doris Musa, doris.musa@fsre.sum.ba, Faculty of Mechanical Engineering, Computing and Electrical Engineering University of Mostar

is handling vulnerabilities at the design level, which are the most common and critical. Determining whether a program has a vulnerability at the design level requires much expertise, making finding these vulnerabilities particularly difficult to automate. Some design-level issues include error handling in object-oriented systems, unsecured data channels, and lack of monitoring/monitoring, which almost always leads to a security risk [1].

1.2 Security testing and risk management

Software security professionals perform a variety of tasks to manage software security risks, including creating malicious use cases, listing normative security requirements, conducting an architectural risk analysis, creating risk-based security test plans, using static analysis tools, performing security and later penetration tests, as well as cleaning up after security breaches. The tasks of architectural risk analysis, risk-based test plan development, and security testing are closely related, as a key aspect of security testing involves the research of security risks [1].

The well-known saying "software security is not security software" emphasizes the importance of security testing. Specific security features such as cryptography and authentication play a key role, but security is still a feature of the entire system, not just security mechanisms [1].

For this reason, security testing should include testing of security mechanisms to verify their correctness and risk-based testing, which simulates an attacker's approach [1].

Risk analysis is a tool to support business decisions, i.e., it is a way of gathering the necessary data to make the right decision based on knowledge of vulnerabilities, threats, impacts, and probabilities [2].

All defined risk analysis methodologies have advantages and disadvantages but share modern software design's fundamental principles and limitations. Risk analysis should be included in the entire software development life cycle [2].

Figure 1 shows the software development life cycle and the specific parts of the cycle where risk analysis is examined [2]. Traditional risk analysis methodologies are divided into two categories, commercial and standardized, and each includes key concepts, namely:

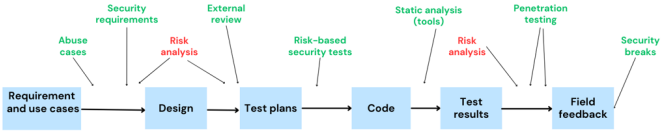

Figure 1 Software development life cycle with an emphasis on risk analysis [2]

- Property constituting an object of protection;
- Risk is the probability of an adverse event affecting an asset;
- The threat is a source of danger;
- A vulnerability is a weakness in a system that an attacker can exploit;
- The impact of risk can be financial, reputational or legal;
- Probability is the probability that an event will occur, expressed as a percentage [2].

Many risk assessment methods calculate the nominal value of an information resource and try to determine the risk as a function of loss and probability of events [2].

The classic method of risk analysis expresses risk as financial loss, i.e., annual expected loss (ALE-Annualized Loss Expectancy*)* according to the following equation:

$$ALE = SLE\ x\ ARO \qquad \text{Eq (1)}$$

SLE (Single Loss Expectancy) is the single expected loss, and ARO (Anual Rate of Occurrence) is the annual rate of incidence [2].

For example, a stock trading app over the internet contains vulnerabilities allowing unauthorized access. If the risk analysis determines that the office procedures will prevent a malicious transaction, the loss would only be the cost of withdrawing transactions, which amounts to $150 per event. If 100 such events occur annually, the total loss (ALE) will be $15.000 [2].

This amount provides a baseline estimate for deciding whether to invest in remediating vulnerabilities. However, an annual loss of $ 15.000

may not be significant for a stock trading company in terms of other market risks [2].

1.3 The Basics of Penetration Testing

Penetration testing is a method of assessing the security of computer systems or networks based on the simulation of actual attacks. During this process, an authorized tester uses the same methods and techniques that a potential attacker would use to identify vulnerabilities that could allow unauthorized access. Although penetration testing effectively detects potential security weaknesses, its role is often overestimated. This method cannot provide absolute certainty or prove that a system has no weaknesses. Still, it can estimate the knowledge and effort an attacker requires to compromise a system. As a result, safety cannot be guaranteed in general circumstances, only in specific circumstances [3].

In the initial stages of penetration testing development, it was believed that all vulnerabilities could be detected through recognizable patterns, which influenced the development of automated tools to find them. However, due to the diversity of computer and operating systems and programming languages, these tools have become impractical and quite complicated. In contrast, adapting existing tools for new environments is expensive and complex [3].

Today's modern systems are focused on a limited number of operating systems and a few programming languages, which leads to the assumption that future tools for identifying vulnerabilities could be much more effective. The new tools are specialized and focus on certain security aspects [3].

Vulnerability scanning uses automated processes for the initial security assessment, while penetration testing confirms or refutes these vulnerabilities. Based on this, a reduced and precise list of vulnerabilities that can be determined with high certainty to exist in the system is created [3].

Testing the security of computer systems, networks and applications through penetration testing has been around for decades but became popular in the early 1990s after the release of the Internet Security Scanner tool from the Georgia Institute of Technology. As the software became more complex, penetration testing was expanded to include application software, which brought new technical challenges. Also, advances in reporting have made it easier to analyze the results, including reports at the organizational level. In addition, it allowed testing professionals to test remotely just before implementation, and it included testing from the outside, simulating an attacker without insight into the system's design. However, this approach has limitations as testers lack a deeper understanding of the application. Although penetration testing has progressed, it is still not fully integrated into the development of software systems since most of the advances are focused on tools rather than on the development of security in the software development process itself [4].

2 General methodology

The process of penetration testing takes place through seven phases, namely:

1. Pre-engagement phase;
2. Reconnaissance (active and passive);
3. Discovery;
4. Vulnerability analysis;
5. Exploitation and post-exploitation;
6. Maintaining access and hiding traces;
7. Reporting.

2.1 Pre-engagement phase

In the first phase, the security specialist analyzes the test logistics and the engagement rules. Testing objectives are defined, which should be clear and aligned with the specific security requirements of the organization [5].

At this stage, it is essential to determine the scope of testing by defining the systems, networks, and applications involved in the testing, as well as to identify prohibited areas or systems to know what can be expected from the testing process [5].

The duration of the testing is also determined with start and end dates, and at the end of this phase, the testing provider is provided with all the access necessary for penetration testing [6].

2.2 Reconnaissance

A penetration test begins with a reconnaissance of the target operating systems by gathering the necessary information about the system, organization, or person. Collecting information is crucial when conducting any test on an information system and provides the prerequisites required to continue testing [3].

This phase includes two ways of collecting information, namely passive and active.

Passive reconnaissance collects information about the target system without direct contact with it. The usefulness of the information collected depends on the type of pentest. For example, the network pentest may reveal additional domain names and IP addresses to be included in the scope [6].

Active reconnaissance sends network requests to test the target system. A testing specialist analyzes the software vulnerabilities published in the discovered software to find further potential vulnerabilities and inbound exact attacks such as authentication forms [6].

2.3 Discovery

This phase of testing is present only in network penetration tests in such a way as to identify as many open ports as possible. The scan is performed using automated tools and applications [6].

The network scanning process involves several key activities. The first step is to identify the active computers within the network, after which their ports and services are scanned to gather information about the available resources. The next step involves determining the outer boundaries of the network, which consists in identifying the routers and firewalls that protect the network. It is also essential to identify critical services that are of great importance for network functioning and data protection. One of the parts of the scanning phase is the collection of information about the operating system based on the specific characteristics of the system, and the MIB (Management Information Base) database is used to identify network routes, which allows an understanding of the flow of data through the network [3].

The final step involves analyzing the active services on the network to gather more detailed information about the types and versions of services running on devices within the network [3].

2.4 Vulnerability analysis

During the secure scan, various sources of threats are detected, which pentesters analyze to identify hidden vulnerabilities and prioritize them according to the risk they pose to the system. An inventory of discovered vulnerabilities is also performed, as well as an assessment of the expected impact and the identification of attack paths and scenarios for exploitation [3].

Following a procedure that assesses severity and risk consistently is necessary to examine vulnerabilities properly. One of the tools used to determine the severity of vulnerabilities is the CVSS (Common Vulnerability Scoring System). CVSS is a widely used tool that assigns a numerical rating to each vulnerability based on its severity. Based on the score obtained, a priority is defined to address the vulnerability [5].

2.5 Exploitation and post-exploitation phase

This phase aims to establish access to the system in a simulated attack through identified vulnerabilities. A testing specialist finds an entry point and searches for available resources. At this stage, testers must be very careful with the functions of the test object so as not to damage the work process. After the pentester, i.e., an expert who conducts security testing of computer systems, networks or applications through attack simulation, exploited vulnerabilities and finds the entrance to the system, the next step is to consider questions such as the amount of access provided by the entry point, how easy it is to maintain access and how long it can take before a breach is noticed, and what is the degree of damage that the vulnerability can cause [5].

When penetration is successful, but access to lower-level systems is obtained, it is possible to increase privileges further. It is necessary to perform mapping of local vulnerabilities, which is the opposite of network-based vulnerabilities, exploiting or developing a proof of concept that has been tested in

an isolated case and applied to the compromised system. The main goal in this mode is to gain administrator privileges based on previously acquired rights of minor importance, which are limited by the applied fixes and tools for checking the system's integrity [3].

If the attempt to achieve access has failed, some other options are available to gain access. Some ways are to detect usernames/passwords using dictionary or brute force attacks, to detect blank or default passwords in system accounts, and to detect public services that allow certain operations, such as writing/creating/reading files. If the attempt to bypass the passwords yields an appropriate result, the compromise of the target and intermediate systems, such as routers, firewalls, etc., follows. [3].

2.6 Maintaining access and hiding traces

Maintaining access and hiding traces are key aspects of the pentest, which allow the tester to remain present in the compromised system without the risk of detection [3].

In this phase, various tools and techniques are applied, such as the use of hidden communication tools, the installation of a retractable door (e.g., Backdoor), using rootkit tools, hiding files, cleaning logs, unchecking integrity checks as well as deactivating antivirus channels. Standard penetration testing does not use techniques such as channel obfuscation, backdoor installation, and rootkits because of the risk of leaving these tools active after testing, allowing potential attackers to access the system in an actual attack [3].

2.7 Reporting

All previous steps of penetration testing contribute to this phase, which includes creating a VAPT and sharing it with the client [5]. VAPT (Vulnerability Assessment and Penetration Testing) is a methodological approach to improving an organization's security posture by identifying, prioritizing, and mitigating vulnerabilities in its infrastructure [7].

The penetration testing report covers the agreed testing parameters and the known limitations of the testing performed. It also features a high-level discussion of the penetration test results, business impact assessment, and recommendations. In addition, the report provides detailed technical information on each vulnerability identified and sensitive data exploited, including a description of the vulnerability, a severity assessment, reproduction steps, evidence, and practical recommendations for remediation [6].

3 Access to penetration testing

Any type of testing, including software security testing, requires determining who should do it and what actions need to be taken [1].

Using a traditional approach, standard testing organizations can perform functional safety testing. One of the primary tasks of functional testing is to ensure that access control mechanisms work as expected. On the other hand, traditional quality assurance teams find it more challenging to conduct risk-based testing due to expertise [1].

Security tests, especially those that lead to full exploitation, are challenging to design because the designer needs to think like an attacker. Another problem is that security tests don't always produce security exploits, which causes visibility issues. A safety test can lead to an unexpected outcome that requires further sophisticated analysis [1].

The white box and black box methods try to understand the software using different approaches depending on whether the testers can access the source code. White box analysis involves analyzing and understanding the source code and design and effectively finding bugs. This approach also has a drawback: it can sometimes report a vulnerability where it does not exist [1].

Black Box testing is based on analyzing the program in operation by sending it various inputs. This type of testing requires an operation program and does not use source code analysis. Malicious inputs are added to the program to try to exploit it. If the program stops working during the test, a security problem can be detected [1].

Each of the testing methods can reveal possible software risks and potential exploits. However, one of

the drawbacks is that most organizations focus on features by devoting little time to understanding or examining risk dysfunction [1].

3.1 Penetration Testing Tools and Frameworks

Testers use various techniques, tools, and frameworks when conducting penetration testing. The most well-known penetration testing tools include Kali Linux, Metasploit, Burp Suite, and Nmap.

- Kali Linux is an open-source Linux distribution based on Debian. It is designed for the needs of digital forensics and security breach testing and comes pre-installed with various security tools. Key features are adaptability to suit the specific needs of users, community support that contributes to its development by providing support through forums and documentation, and portability since it can be run on a virtual machine or installed on a hard disk [8].
- The Metasploit Framework is a project that contains information about security vulnerabilities and helps with penetration testing. The tool is designed for anti-forensic techniques and evading detections [9]. This tool comes pre-installed in penetration testing operating systems such as Kali Linux. It contains a collection of tools that hackers use to check for security vulnerabilities, perform attacks, and evade detections [10].
- Burp Suite is a graphical tool for testing the security of web applications. The tool is developed in Java by PortSwigger Web Security. It encompasses a collection of tools combined into a single package for web application security. Burp Suite can be used as a basic HTTP proxy to intercept traffic for analysis and playback, a web application security scanner, and a tool to perform automated attacks on web applications and inspect the entire website to identify attacks [11].
- Nmap is an open-source network discovery and security auditing tool. Many system and network administrators find it helpful to organize service upgrade schedules and keep track of host or service uptime. Nmap uses a more advanced technique of sending raw IP packets to identify which hosts are available on the network, what services they provide, as well as which operating systems are installed. The Nmap package includes an advanced GUI and result viewer, flexible data transfer, redirection and debugging tool, and a package generation and analysis tool [12].

In penetration testing, in addition to tools, frameworks that help testers in the penetration testing phases are also important: OWASP, PTES, ISSAF, OSSTMM, and NIST.

- OWASP (Open Web Application Security Project) is a non-profit organization that provides various security testing guides, tools, and methods with open licenses, including OTG (OWASP Test Guide). The three main components are web application development with the OWASP Testing Framework, a testing methodology for web applications, and reporting. This framework allows developers to build security-oriented web applications through penetration testing and security recommendations, instruments, and standards for coding and testing web and mobile applications [13].
- The Penetration Testing Execution Standard (PTES) is a framework developed in 2009 by Nickerson et al. This framework includes pre-engagement interaction, information gathering, threat modeling, vulnerability analysis, exploitation, post-exploitation, and reporting. It can also be combined with other frameworks like OWASP for application testing. This framework aims to create guidelines for penetration testing where security professionals can use basic guidelines for the expected security requirements of the pentest [14].
- The Information Systems Security Assessment Framework (ISSAF) is an open license framework developed by the Open Information Systems Security Group (OISSG). This framework seeks to cover all aspects of penetration testing, and one of the advantages is to show the different

relationships between the tasks and the associated tools for each task [15], [16], [17].

- OSSTMM (Open-Source Security Testing Methodology Manual) was created in 2000 as a free framework from v.3, while v.4 required a subscription. The framework is not as comprehensive as ISSAF but is the basic methodological framework for audits [18].
- The National Institute of Standards and Technology (NIST) presents a cybersecurity framework that is designed to identify and mitigate cyber risks in the industry. NIST lists tasks that must be completed during the pentest without explaining how to inventory physical devices, systems, or software applications and platforms [19], [20].

Figure 2 provides an overview of testing tools based on 27 selected papers that specifically study frameworks widely used with corresponding tools [14].

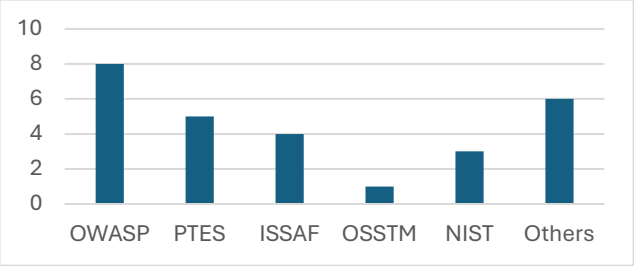

Figure 2 Penetration testing frameworks [14]

Based on the analysis and research question (What frameworks are widely used in penetration testing security assessments?), it was found that there is a vast demand for OWASP, followed by PTES and ISAF, OSSTM, and NIST, with several frameworks already popular and named after development teams [14].

4 Use Case: Penetration testing for software security in financial institutions

Financial institutions are frequently the targets of cyberattacks due to the high volume of financial transactions they process and the sensitive data they manage. Identifying security vulnerabilities is essential to prevent potential attacks, as their software systems are complex and rely on web applications. This case study examined the penetration testing of a financial institution's software system, focusing on the security of web applications and critical system components.

4.1 Methodology

A structured approach was used across several important stages of the testing methodology. Information on the system was gathered in the first phase from publicly accessible sources, such as DNS services and WHOIS databases, which were used to identify domains and IP addresses. We carried out passive research by identifying exposed devices and services with Shodan and identifying security flaws in publicly accessible components with Censys. Network scanning and port analysis were active techniques used to find open services and potential application entry points [2], [5], [14], [15].

A combination of automatic and manual techniques was used to identify vulnerabilities. In order to find known vulnerabilities, we used security scanners. Simultaneously, manual testing made it possible to precisely analyze particular application functionalities like API protection, session security, and input validation. Targeted attacks like SQL injection and XSS were simulated in order to assess the probability of unauthorized data access and system compromise. The final step involved creating reports that detailed vulnerabilities, their consequences, and recommendations for improving security [2], [5], [14], [15].

4.2 Results

The test results shown in Table 1 indicated the presence of significant security vulnerabilities. A total of 20 vulnerabilities were identified, of which 7 (35%) were classified as critical due to the high risk to the system and user data. Among them were SQL Injection, which allowed unauthorized access to the database, and XSS vulnerabilities, which compromised user sessions. Analysis of session mechanisms revealed a weakness in cookie settings, while vulnerabilities were noted in API components due to inadequate encryption and authorization rules.

Table 1 Display of penetration testing results

Vulnerability	Description	Number of cases	Percentage of total vulnerabilities	Classification
SQL Injection	Enables unauthorized access to the database through manipulation of SQL queries	3	15%	Critical
Cross-Site Scripting (XSS)	Allows insertion of malicious code that can compromise user sessions	4	20%	High Priority
Weak Authentication Mechanisms	Lack of two-factor authentication and weak password protection	5	25%	Critical
Inadequate API Protection	Missing encryption of data in transit and weaknesses in authorization rules	3	15%	High Priority
Weak Session Management	Poorly configured cookies allow potential session hijacking.	5	25%	Critical

The implementation of a Content Security Policy (CSP) to prevent XSS, input checking to prevent SQL Injection, and enhanced authentication procedures and encryption to fortify API security were among the suggested actions.

5 Conclusion

Penetration testing is a method that allows organizations and individuals to identify vulnerabilities through a security assessment. The effectiveness of penetration testing requires a holistic approach that includes a transparent methodology, the use of specific tools, and the selection of appropriate testing frameworks. Frameworks such as OWASP, PTES, and NIST provide structured guidance through the penetration testing phases, from planning and information gathering to exploitation and reporting. Tools such as Nmap for scanning networks or Metasploit for vulnerability testing are crucial for identifying and exploiting security weaknesses in the system. It is essential to align the methodology and tools with the characteristics of the target system, whether it is web or mobile applications, networks, and more, to ensure a complete insight into threats. Effective penetration testing is the basis for maintaining high system security and protection against potential threats. Penetration testing greatly decreased the danger of cyberattacks by enabling the identification of critical vulnerabilities and the proposal of focused countermeasures. Financial institutions can improve system security and boost user confidence in data protection by using this strategy.

Conflicts of Interest: The authors report there are no competing interests to declare;

References

[1] B. Potter and G. McGraw, "Software security testing", *IEEE Security & Privacy*, vol. 2, no. 5, pp. 81–85, Sep. 2004, doi: 10.1109/MSP.2004.84.

[2] D. Verdon and G. McGraw, "Risk analysis in software design", *IEEE Secur. Privacy*, vol. 2, no. 4, pp. 79–84, Jul. 2004, doi: 10.1109/MSP.2004.55.

[3] CARNet CERT and LS&S, "Metodologija penetracijskog testiranja," 2008. [Online]. Available: https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2008-02-219.pdf

[4] K. Van Wyk, "Adapting penetration testing for software development purposes," Jan. 2007. [Online]. Available: https://apps.dtic.mil/sti/pdfs/AD1180049.pdf

[5] H. M. Z. A. Shebli and B. D. Beheshti, "A study on penetration testing process and tools", in *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, May 2018, pp. 1–7. doi: 10.1109/LISAT.2018.8378035.

[6] "Penetration Testing Phases". Accessed: Jan. 16, 2025. [Online]. Available: https://amatas.com/blog/penetration-testing-phases/

[7] J. N. Goel and B. M. Mehtre, "Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology", *Procedia Computer Science*, vol. 57, pp. 710–715, 2015, doi: 10.1016/j.procs.2015.07.458.

[8] G. Whittaker, "Hacking Made Easy: A Beginner's Guide to Penetration Testing with Kali Linux | Linux Journal". Accessed: Jan. 16, 2025. [Online]. Available: https://www.linuxjournal.com/content/hacking-made-easy-beginners-guide-penetration-testing-kali-linux

[9] Metasploit, "Metasploit Framework User Guide," *Amyotroph*. lateral Scler. Off. Publ. World Fed. Neurol. Res. Gr. Mot. Neuron Dis., vol. 11, no. 1–2, pp. 38–45, 2010.

[10] S. Raj and N. K. Walia, "A study on Metasploit Framework: a Pen-Testing tool," *2021 International Conference on Computational Performance Evaluation (ComPE)*, pp. 296–302, Jul. 2020, doi: 10.1109/compe49325.2020.9200028.

[11] P. Kumawat, "Introduction to Burp Suite – Guide for Burp Suite," *Security Cipher*, Nov. 15, 2023. https://securitycipher.com/2020/06/07/introduction-to-burp-suite-guide-for-burp-suite/

[12] "Nmap: the Network Mapper - Free Security Scanner." https://nmap.org/

[13] The Open Web Application Security Project, "Testing guide," book. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf

[14] H. M. Adam, Widyawan, and G. D. Putra, "A Review of Penetration Testing Frameworks, Tools, and Application Areas", in *2023 IEEE 7th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, Purwokerto, Indonesia: IEEE, Nov. 2023, pp. 319–324. doi: 10.1109/ICITISEE58992.2023.10404397.

[15] J. A. Pratama, A. Almaarif, and A. Budiono, "Vulnerability Analysis of Wireless LAN Networks using ISSAF WLAN Security Assessment Methodology: A Case Study of Restaurant in East Jakarta," *2021 4th International Conference of Computer and Informatics Engineering (IC2IE)*, pp. 435–440, Sep. 2021, doi: 10.1109/ic2ie53219.2021.9649360.

[16] I. G. A. S. Sanjaya, G. M. A. Sasmita, and D. M. S. Arsa, "Information Technology Risk management using ISO 31000 based on ISSAF Framework Penetration Testing (Case Study: Election Commission of X City)," *International Journal of Computer Network and Information Security*, vol. 12, no. 4, pp. 30–40, Aug. 2020, doi: 10.5815/ijcnis.2020.04.03.

[17] F. Abu-Dabaseh and E. Alshammari, "Automated Penetration Testing : An Overview", in *Computer Science & Information Technology*, Academy & Industry Research Collaboration Center (AIRCC), Apr. 2018, pp. 121–129. doi: 10.5121/csit.2018.80610.

[18] A. Giuseppi, A. Tortorelli, R. Germana, F. Liberati, and A. Fiaschetti, "Securing Cyber-Physical Systems: An Optimization Framework based on OSSTMM and Genetic Algorithms," *2022 30th Mediterranean Conference on Control and Automation (MED)*, pp. 50–56, Jul. 2019, doi: 10.1109/med.2019.8798506.

[19] N. M. Karie, N. M. Sahri, W. Yang, C. Valli, and V. R. Kebande, "A review of security Standards and Frameworks for IoT-Based Smart Environments," *IEEE Access*, vol. 9, pp. 121975–121995, Jan. 2021, doi: 10.1109/access.2021.3109886.

[20] B. A. B. Arfaj, S. Mishra, and M. AlShehri, "Efficacy of unconventional penetration testing practices," *Intelligent Automation & Soft Computing*, vol. 31, no. 1, pp. 223–239, Sep. 2021, doi: 10.32604/iasc.2022.019485.