INSTITUTO
UNIVERSITÁRIO
DE LISBOA

Web Application Penetration Test: Proposal for a generic Web Application Testing Methodology

Hugo Pinto João

Master Degree in Computer Science and Business Management,

Supervisor:

Phd Carlos Jorge Corredoura Serrão, Associate Professor

ISCTE - IUL

November/2021

TECNOLOGIAS
E ARQUITETURA

Web Application Penetration Test: Proposal for a generic Web Application Testing Methodology

Hugo Pinto João

Master Degree in Computer Science and Business Management,

Supervisor:

Phd Carlos Jorge Corredoura Serrão, Associate Professor

ISCTE - IUL

November/2021

Acknowledgments

Sincere thanks to the supervisor of this thesis, Dr. Carlos Serrão, for providing guidance and unrestful dedication throughout the development of this dissertation. I am profoundly grateful for the critical spirit shown during the whole process, the follow-up and initiatives to share his knowledge and his time to give me the motivation to complete this stage of my life. The meetings and conversations were vital to form a comprehensive and objective critique mindset. It goes without saying that this thesis would not have been possible without you.

I would also like to thank my co-workers at VTXRM – Software Factory, particularly Fernando Moura, Luís Cravo, João Santos, Guilherme Lopes, and Filipe Ferreira, for their participation, availability, and aid for any existing doubt or technical problem, as well as the support received during the whole development.

To the entire board of directors at VTXRM – Software Factory for the allowance of the development of this project, making it possible to test their software. Thanks for all the trust and support.

To my father, sister, brother-in-law, and all of my friends thank you for your constant support and belief in me, thank you for the motivation and sermons that made me a more diligent person.

To all individuals that generously shared their tools and expertise with the open-source community, thank you.

Resumo

Atualmente, a Gestão de Segurança da Informação começa a tornar-se uma prioridade para a maioria das Empresas, com o principal objetivo de impedir que identidades não autorizadas acedam a informações confidenciais e as utilizem contra a organização. Uma das melhores formas de mitigar os possíveis ataques é aprender com as metodologias dos atacantes. Existem inúmeras formas de o fazer, mas a mais comum baseia-se na realização de Testes de Intrusão, uma simulação de um ataque para verificar a segurança de um sistema ou ambiente a ser analisado. Este teste pode ser realizado através de meios físicos utilizando hardware, através de engenharia social e através de vulnerabilidades do ambiente. O objetivo deste teste é examinar, em circunstâncias extremas, o comportamento de sistemas, redes, ou dispositivos pessoais, para identificar as suas fraquezas e vulnerabilidades.

Nesta dissertação será apresentada uma análise ao estado da arte relacionada com testes de penetração, as ferramentas e metodologias mais utilizadas, uma comparação entre elas, serão também explicadas algumas das vulnerabilidades mais críticas em aplicações web. O objetivo é o desenvolvimento de uma metodologia genérica de testes de intrusão, ambicionando a sua aplicabilidade e genericidade em aplicações web, sendo esta aplicada e descrita num teste de intrusão real à aplicação web desenvolvida pela VTXRM – Software Factory (Accipiens), aplicando passo a passo métodos e softwares Open-Source com o objetivo de analisar a segurança dos diferentes componentes do sistema no qual o Accipiens está instalado. No final serão apresentados os resultados do mesmo e a sua análise.

Palavras-chave: Testes de Intrusão, Cibersegurança, Segurança, Aplicações Web, Auditoria de Segurança TI, Riscos de TI e OWASP.

Abstract

Nowadays, Security Management is beginning to become a priority for most companies. The primary aim is to prevent unauthorized identities from accessing classified information and using it against the organization. The best way to mitigate hacker attacks is to learn their methodologies. There are numerous ways to do it, but the most common is based on Penetration Tests, a simulation of an attack to verify the security of a system or environment to be analyzed. This test can be performed through physical means utilizing hardware or through social engineering. The objective of this test is to examine, under extreme circumstances, the behavior of systems, networks, or personnel devices, to identify their weaknesses and vulnerabilities.

This dissertation will present an analysis of the State of the Art related to penetration testing, the most used tools and methodologies, its comparison, and the most critical web application vulnerabilities. With the goal of developing a generic security testing methodology applicable to any Web application, an actual penetration test to the web application developed by VTXRM – Software Factory (Accipiens) will be described, applying methods and Open-Source software step by step to assess the security of the different components of the system that hosts Accipiens. At the end of the dissertation, the results will be exposed and analyzed.

Keywords: Penetration tests, Cyber Security, Security, Web Application, IT Security Audit, IT Risks and OWASP.

Index

Table Index

Figure Index

Glossary

IC3 – Internet Core Competency Certification

SQL – Structured Query Language

Pentest – Penetration Test

CSS – Cross-Site Scripting

OWASP – Open Web Application Security Project

Webapp – Web Application

PTES – Penetration Testing Methodologies and Standards

DSR – Design Science Research

DNS – Domain name servers

CSRF – Cross-Site Request Forgery

CORS – Cross-Origin Resource Sharing

XSS – Cross-Site Scripting

XML – Extensible Markup Language

HTTP – Hypertext Transfer Protocol

WASC – Web Application Security Consortium

HSTS – HTTP Strict Transport Security

RFC – Request for Comments

CSP – Content Security Policy

CVSS – Common Vulnerability Scoring System

NDA – Non-Disclosure Agreement

Chapter 1 - Introduction

Nowadays, Information Security Management is beginning to turn into a priority for most companies. The leading aim is to prevent unauthorized identities from accessing classified information and using it against the organization (Alzahrani, 2018). Successful cyber-attacks can lead to significant losses regarding classified information, direct economic losses, and reputational damage. The need for vulnerability prioritization in organizations is widely recognized and systematically increasing; therefore, new ways of reporting and assessing these vulnerabilities have emerged within the last years.

1.1 Problem statement and context

Our society is more technologically dependent than ever before, but cybercrime is also on the rise and becoming more sophisticated. According to Cybersecurity Ventures (Morgan, 2020), the average cost of cybercrime to companies has increased, reaching a total value of six trillion dollars in total per year. Data breaches constantly increase, giving attackers access to sensitive data, personally identifiable information (PII), protected health information (PHI), personal information, intellectual property, data, and governmental and industry information systems.

Unfortunately, the up-to-date security approach may not be enough because security flaws may result from misconfigured settings, network infrastructure design flaws, poorly implemented software, and many others. If an organization truly wants to avoid risks, it should adopt a proactive approach. To do it, it needs to seek out all types of vulnerabilities by systematically and actively testing the security for vulnerabilities (Bechtsoudis & Sklavos, 2012). This can be done through different methodologies, like penetration testing or vulnerabilitiy analysis. In this thesis the main focus will be penetration testing which is an approach that simulates attacks to verify the security of a system or environment to analyze its flaws (Denis et al., 2016).

These tests are made by targeting different entry points for the system since it can be done through Database Injections (Abdul Raman, 2019), Network Man-in-the-Middle (Vondráček et al., 2018), Denial of Service, Packet Sniffing, and many other possible attack vectors (Cangea, 2018; Wang et al., 2016). In this dissertation, we will focus on Penetration Tests for Web Applications, how to deal with flaws related to its runtime environment, some tools and methods used to test these flaws, and how to audit the security of Web Applications.

Web Application security is more critical now than it used to be. With the rising technological adoption (Skare & Riberio Soriano, 2021) and daily use of web applications for accessing data, tools, social media, and even work collaboration directly over the internet, web applications security was never so important. As it is possible to verify on figure 1, complaints regarding cyber crimes have been

increasing from 2016 to 2020. In addition to direct financial and data theft, web application threats can destroy assets, customer goodwill, and business reputations, leading to huge losses.

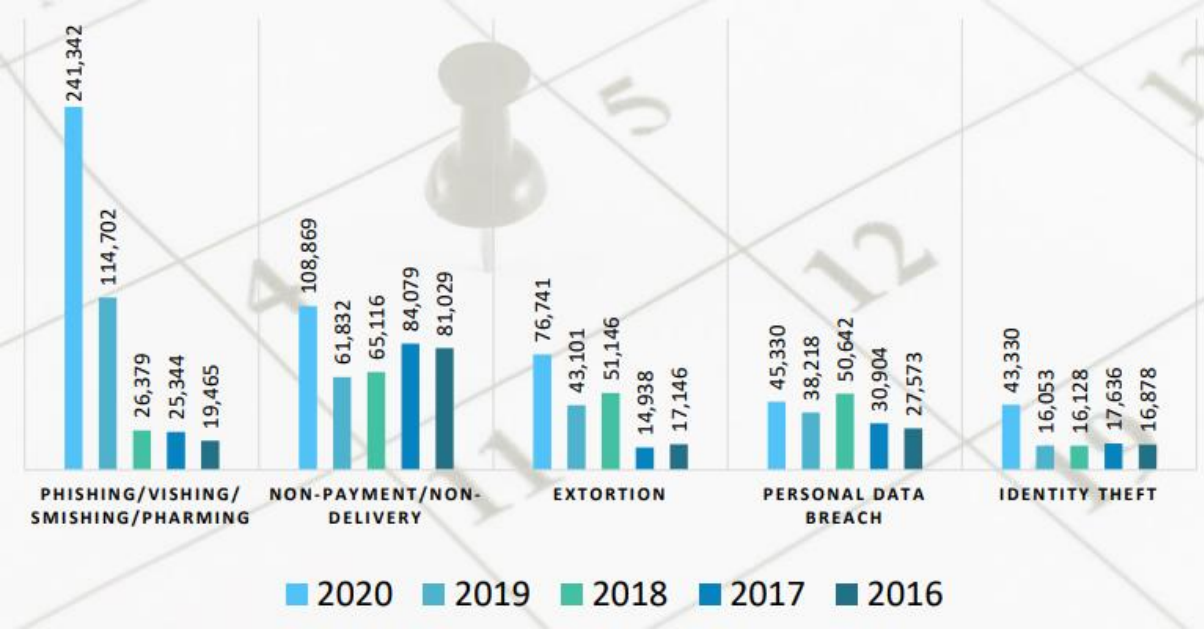

Figure 1: Complaint Statistics on top 5 Crime Type Comparison of the last five years(Federal Bureau of Investigation - Internet Crime Complaint Center, 2020).

Web application security deals with security attacks against the application layer. Therefore, the concern is to protect Web application servers against different security threats that exploit applications' vulnerabilities (Jain & Jain, 2019). Combining a set of tools and the proper methodology makes it possible to cover a broader range of security issues in a web application.

The penetration testing process can be divided into five generic stages: planning and reconnaissance, where the scope and goal of the tests are defined and the intelligence needed to understand how a target works and its potential weaknesses are gathered; the scanning stage, where the pentester will understand how the target application can respond to different intrusions by scanning the network through static or dynamic analysis (Poltavtseva & Pechenkin, 2017); the third stage is exploitation, as it is implicit, in this stage the pen testers use tools like SQL injection, cross-site scripting and so on to exploit vulnerabilities in the target in order to gain access to its data (Cangea, 2018; EC-Council, 2020). After these three initial stages, it comes to maintaining access and analyzing the results, assuring that the flaws found are real vulnerabilities and not false positives, to report them (Liu et al., 2017). The report will contain important and detailed data that includes the sensitive data accessed, the specific vulnerabilities exploited, the time the pentester remained in the system undetected, and other information required.

1.2 Research Questions

This dissertation intends to answer the following research question: Is it possible to specify and design a generic web application penetration testing methodology applicable to most web applications?

1.3 Research Objectives

The objectives of this research are:

- Research on the most common methods, best practices, and tools used for web application penetration testing
- Design a generic methodology applicable to conduct the assessment of any web application;
- Test the designed methodology on a real case scenario, using a production web application
- Evaluate the applicability of the methodology.

1.4 Research Methodology

The development of this dissertation will be conducted using the Design Science Research Methodology allied with a Systematic Literature Review to research insight on the subject and then design a solution based on the findings. Under the Design Science Research (DSR) methodology, it was intended to perform a Systematic Literature Review to research insight on the subject. In DSR, the provided guidelines assist researchers in conducting research based on defined principles, procedures, and practices. DSR defines an iterative process divided into six stages (see Figure 2), problem identification and motivation, the definition of the objective, the design and development, the demonstration, the evaluation, and the communication (Peffers et al., 2020). It has also been structured in three phases where the six stages are inserted, being this the exploration, induction, and deduction of the problem through the context and the activities which leads to setting hypothesis, and then we have the second phase where the solution is designed and tested to verify the hypothesis and the last phase where the research is validated and generalized to other applications (Horvath, 2007). The main goal of the methodology is to achieve knowledge and understand a specific problem domain by building and applying a designed artifact (Hevner & Chatterjee, 2010).

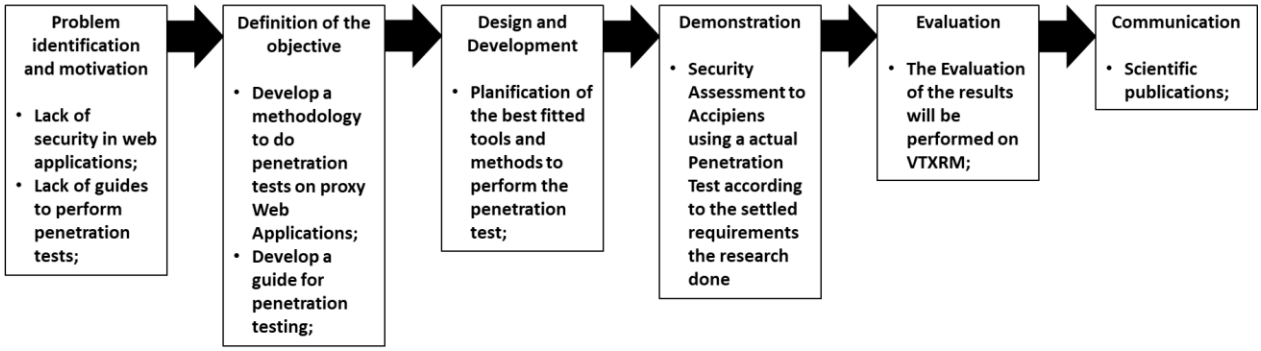

Figure 2: Design Scientific Research Stages

1.4.1 Problem identification and motivation

Security Management is beginning to turn into a priority for most companies. Taking that into consideration, one of the best ways to evaluate the security of web applications is through a penetration test. Considering that there are almost no guides on how to perform a penetration test on web applications justifying the tools and methods used, this dissertation will present the beginning of the development of a methodology to test applications that have similar characteristics to the target of the pentest.

1.4.2 Definition of the objective

The objective of this dissertation is to present a step-by-step demonstration and development of a methodology to assess the security of the web application Accipiens (developed by VTXRM - Software Factory) through a penetration test, including all the tools and methods used, the reason for its appliance and the analyses of the results.

1.4.3 Design and development

The design and development of this methodology is done on the Chapter 3, presenting all of the stages and processes to perform a pentest, the input, tools and techniques and output of every process as well as the representation of all processes.

1.4.4 Demonstration

The demonstration will be conducted through a pentest on Chapter 4 where it is reported the environment setup, the eligibility of Accipiens, the tools used, as well as the entire penetration test process, using the defined methodology and tools that most fit the application, the penetration test requirements and goals by adopting the defined guidelines to assess Accipiens security.

1.4.5 Evaluation

The evaluation was conducted in both VTXRM and this dissertation by analyzing the results, evaluating the whole process, and confirming the vulnerabilities and entry points.

1.4.6 Communication

This stage refers to the communication of the dissertation in scientific publications (in this case, the master's dissertation).

1.5 Document structure

This work is organized by chapters as follows: In Chapter 2, it is presented the state of the art and related work; Chapter 3 will present the design and the development of the proposed generic web application methodology; in Chapter 4, Accipiens will be explained, as well as the environment setup, the tools used, and the demonstration of the applicability of the developed methodology to the Accipiens use-case; Chapter 5 will present the conclusions of this dissertation and provide information for future research and why it may be essential to continue developing this work.

Chapter 2 – State of the Art

At this point, this research will contextualize the subject's scope by extracting conclusions about the methods and tools used by Penetration Testers. First, it will describe what Cybersecurity is as well as Penetration Tests. The related work will then contextualize with more information on Web Application Vulnerabilities, in-depth knowledge on Penetration Tests, the most common exploitations made on Web Applications, and the most common tools used.

2.1 Cybersecurity

This era is thought to be the most secure, with everyone leaning towards security compared to the ease of interoperability (Mukhopadhyay et al., 2019) which may lead to more security flaws due to its unrestricted sharing possibilities. Cybersecurity is the practice of defending computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It is also known as information technology security or electronic information security. The term applies in various contexts, from business to mobile computing, and can be divided into a few common categories. Many organizations understood that applying the same techniques, procedures, and tools used by attackers would be an efficient form of testing and assessing risks and threats to their systems (Gondim et al., 2016).

These problems may be due to network security, application security, information security, operational security, or end-user education. To effectively deal with these concerns, forming a security policy according to its requirements and objectives would be a proactive approach to provide sufficient defense layers against various attacks (Bechtsoudis & Sklavos, 2012).

An attack is some hostile, aggressive, or malicious action executed using its infrastructure or services flaws against itself, and an attacker is some entity that executes an attack.

There are two types of cyber-attacks: passive and active. Passive attacks are challenging to detect. They only monitor and scan the traffic between computers. Active attacks are attempts to make unauthorized modifications to the system. They are easy to detect but usually produce considerable damages, such as alterations of transmitted and stored data or new data flows created to access computers.

2.2 Context of Penetration Testing

Penetration Testing is a simulation of an attack to verify the security of a system or environment to be analyzed. The objective of this test is to examine, under extreme circumstances, the behavior of systems, networks, or personnel devices, to identify their weaknesses and vulnerabilities (Alzahrani, 2018). There are five types of penetration tests, external testing, internal testing, blind testing, double-blind testing, and targeted testing. In external tests, the attacker targets a company's externally visible servers

or devices, such as domain name servers (DNS), e-mail servers, Web servers, or firewalls, to determine whether an outside attacker can gain illegitimate access and which level of access he can obtain. The internal tests simulate an inside attack behind the firewall by an authorized user with standard access privileges. A blind test simulates the actions and procedures of an actual attacker by strictly limiting the information given to the person or team that is performing the test beforehand. Double-blind testing takes the blind test even further by only warning a few individuals within the organization that a test is being conducted (Denis et al., 2016). Targeted testing on the otherhand requires the tester and security personnel to work together and keep each other updated on every movement made during the tests, providing real-time feedback from both parties. When conceiving a Penetration test, it must be decided which approach should the tester take, Black-Box, where the tester either considers some information or no information at all about a specified target (IP address, source code, or any additional information), White-Box, where all the information regarding the target is shared or Gray-Box, where only part of the information is shared, but everything else is hidden (Cangea, 2018).

There is an enormous variety of tools, mostly suitable for automatic testing, from open-source tools to commercial tools, and lately, more online tools are also being developed with no cost associated with the tester (Kalirathinam, 2019).

2.3 Related Work

To acquire more in-depth knowledge about the subject, a Systematic Literature Review was performed (Kitchenham et al., 2009). The steps taken on this review are visible in Table 1.

Table 1 – Systematic Literature Review Stages (Pereira & Serrano, 2020)

Outlining Systematic Literature Review	Conducting Systematic Literature Review	Reporting the Review
Identification of the need for a review ● Lack of security and the increased need for data protection	Applying filters and getting final articles ● 28 Articles	Report the findings ● Discussion about data and draw conclusions
The objective of the review ● Perform research on the primary method and tools to conduct a Penetration Test on Web Applications	Perform Data extraction and analysis of the sample ● Extract information about methods, tools, and methodologies on Penetration Tests	

Review Protocol ● Use a search string, filters, repositories, and defining inclusion criteria.	● Analysis of the sample characteristics	

2.3.1 Outlining Systematic Literature Review

The main objective of this research is the review of the main methods and tools to conduct Web Applications Penetration Testing. However, since the initial review process, it is clear that there are numerous ways to perform Penetration Tests and report the results.

In order to obtain more information about this subject, seven online repositories were selected:

- Scopus;
- Taylor & Francis;
- Web of Science;
- EBSCOHost;
- Wiley Online Library;
- AIS eLibrary;
- Google Scholar;

From the review, the selection involved only articles in English, published in Journals or Scientific Magazines and Conferences Proceedings released between 2015 and 2020, with some articles developed prior to 2015 because of the relevance for this research.

Since all the Online Repositories use different search options, a keyword adaptation for each repository was made, being the initial search made. The results were selected according to four filters and then added to the Mendeley software to store all the articles and their information.

The first filter applies the keywords to the article title or the abstract or the author keywords, on the second filter were the inclusion criteria of articles in English, published between 2015-2020 and peer-review articles, the articles that did not meet these requirements were rejected (later, some articles prior to 2015 were used for their relevance on this subject). On the third filter, was applied the quality criteria, where only articles from Journals and Conferences were selected. The fourth and last filter used was the Manual Selection of articles, where duplicated articles were removed and then, by assessing articles introduction and abstract, there was a selection of the most relevant to the subject review as well as some complementary ones needed to understand better the context of some security issues.

The keywords for the search were used in all repositories with operators AND and OR, being “Penetration Tests” the main keyword.

Keywords: (“Penetration tests”) AND ((“Cyber Security” OR “Security”) OR (“Web Application Security” OR “WebApp Security”) OR (“IT Security Audit” OR “Security Audition” OR “IT Risks”) OR (“OWASP”))

The main focus of this search was indeed Penetration Tests, but the inclusion of related subjects such as Web Applications and IT Security created a more specific group of articles. The term OWASP means Open Web Application Security Project. This not-for-profit foundation works to improve software security, one of the leading entities about the subject that, through community-led open-source software projects, has its influence in software security worldwide.

2.3.2 Conducting a Systematic Literature Review

As mentioned before, each platform has different search options. Due to those differences, Taylor & Francis and Web of Science were used without the first filter.

The first filter (F1) was to select only articles related to the subject on the three domains included (article title, abstract, and author keywords). These domains were selected because they summarise the article’s matter, therefore being the main parts.

The second filter (F2) was the inclusion/exclusion criteria of the articles where it excluded all before 2015, not in English and white papers.

On the third filter (F3), the quality criteria were used to select only articles from journals and conferences, turning 1465 articles into a selection of only 70 articles. Consequently, 70 article introductions and abstracts were read to select the most relevant for the subject and its context, and it was also made a verification for duplicates (F4), ending with a total of 28 articles.

Table 2 – Filtration Process

	F0	F1	F2	F3	F4
Scopus	540	205	109	18	6
Taylor & Francis	74	-	24	11	5
Web of Science	54	-	28	7	3
EBSCO Host	1082	74	16	9	4
Wiley Online Library	297,250	40,632	1,269	15	4

AIS eLibrary	2,684	408	9	6	3
Google Scholar	7,070	12	10	4	3
Total	308,754	41,321	1,465	70	28

2.3.3 Information Extraction Process

After the previous selection of articles, they were analyzed. For each one, the methodologies, methods, year of publication, domain, and other characteristics were extracted to organize the review and its context.

By examining Figure 3, between 2015-2020 (the focus of the research previously filtered), it is noticeable that the distribution over the years tends to be increasing, possibly reflecting the widening need for more secure software and new methods, tools, and methodologies being developed on this subject.

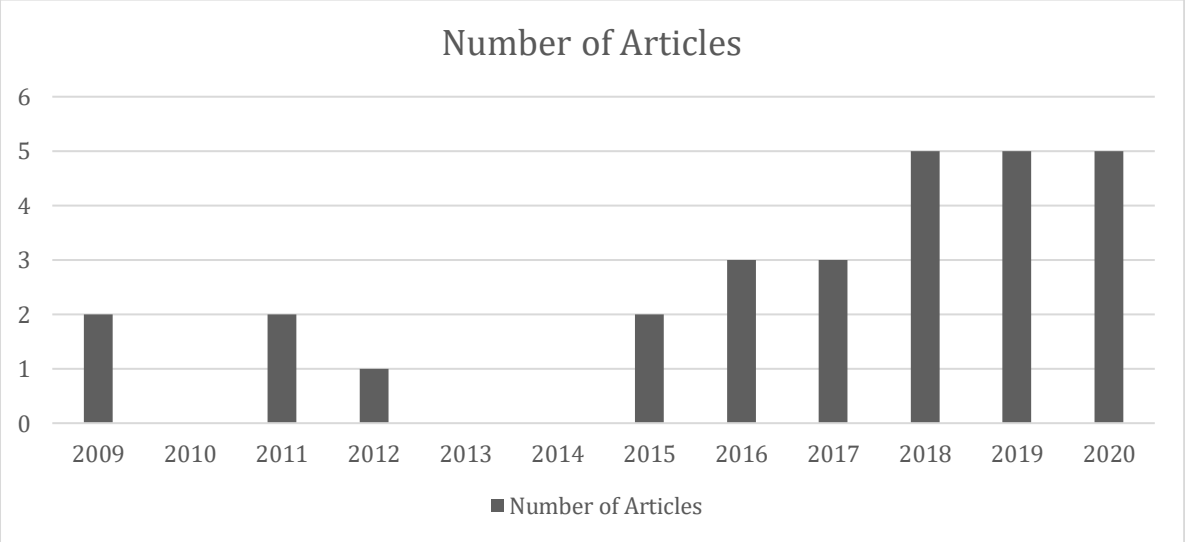

Figure 3: Distribution of papers over the years

With Figure 4, we can observe that 14 articles belong to journals, representing 50% of the sample collected, with 13 articles from Conference Proceedings and one from books.

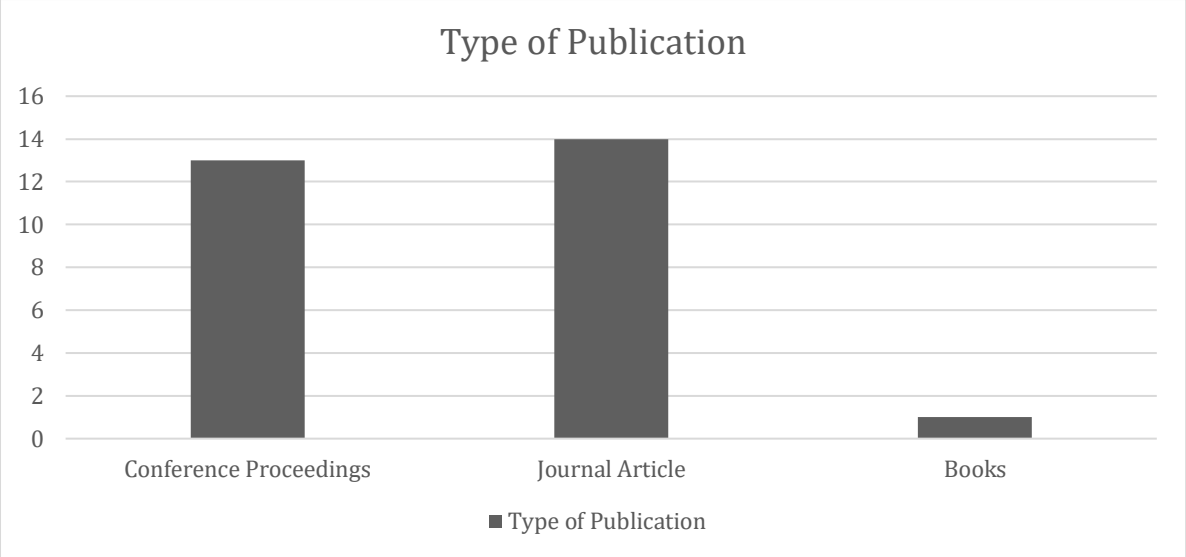

Figure 4: Distribution of papers by type of publication

2.4 Web Application Vulnerabilities

In terms of Web Application vulnerabilities, many can be found during penetration tests (see Table 3) and many risks can be associated with them. However, the most common ones are categorized by OWASP or WASC (Web Application Security Consortium), which are top entities in this field of research. From OWASP, Web Applications have ten standard security risks associated with it, while WASC divides them into more specific risks, creating a top 50 (Kim et al., 2009). These vulnerabilities are susceptible to two different types of attack, Client-Side Attacks, and Server-Side Attacks. The Client-Side attacks include various attacks (such as Cross-Site Request Forgery (CSRF), Cross-Origin Resource Sharing (CORS), Cross-Site Scripting (XSS), Clickjacking, HTML Injection), to steal client's data from websites. In contrast, Server-Side Attacks are deployed against the webserver by targeting vulnerable endpoints and sending malicious payloads to the servers. After the successful execution of the payload, it responds to the attacker with the requested confidential data. This confidential data includes server information, server services and version-related info, user information, passwords, and others. (Nagendran et al., 2019).

Table 3 – Vulnerabilities found with Penetration Tests

Vulnerabilities	Reference
Clickjacking	(Jain & Jain, 2019; Nagpure & Kurkure, 2017; Singh et al., 2020)
SQL Injection	(Abdul Raman, 2019; Farah et al., 2015; Kalirathinam, 2019; Liu et al., 2017; Nagendran et al., 2019; Nagpure & Kurkure, 2017; Palma Salas & Martins, 2015)
Cross-site Scripting (XSS)	(Kalirathinam, 2019; Nagendran et al., 2019; Nagpure & Kurkure, 2017; Palma Salas & Martins, 2015; Vijayalakshmi & Syed Mohamed, 2020)
Malformed XML	(Jain & Jain, 2019; Palma Salas & Martins, 2015)
XML Bomb	(Palma Salas & Martins, 2015)

Other XSS Vulnerabilities	(Nagpure & Kurkure, 2017)
Cross-site Request Forgery	(Jain & Jain, 2019; Kalirathinam, 2019; Nagendran et al., 2019; Nagpure & Kurkure, 2017)
File Upload Vulnerabilities	(Kalirathinam, 2019; Nagpure & Kurkure, 2017; Singh et al., 2020)
Privilege Escalation	(Nagendran et al., 2019; Nagpure & Kurkure, 2017)
Xpath Injection	(Kalirathinam, 2019; Palma Salas & Martins, 2015)
HTTP response splitting	(Kalirathinam, 2019)
Session Exploitation	(Kalirathinam, 2019; Nagpure & Kurkure, 2017)
Browser Cache Weakness	(Nagpure & Kurkure, 2017)
DoS (denial of service)	(Gondim et al., 2016)
Bypass Authentication	(Cangea, 2018; Kalirathinam, 2019; Nagpure & Kurkure, 2017; Wang et al., 2016)
PHP Vulnerable Code	(Mudiyanselage & Pan, 2020; Nagendran et al., 2019)
Man-in-the-Middle	(Bechtsoudis & Sklavos, 2012; Cangea, 2018; Kalirathinam, 2019; Mukhopadhyay et al., 2019; Vondráček et al., 2018; Wang et al., 2016)
Spoofing	(Bechtsoudis & Sklavos, 2012; Cangea, 2018; Singh et al., 2020; Wang et al., 2016)
Sniffing	(Alzahrani, 2018; Bechtsoudis & Sklavos, 2012; Cangea, 2018; Kalirathinam, 2019; Wang et al., 2016)

2.5 Penetration Tests

When it is proposed to perform a penetration test, there are three possible approaches for the test: a black-box approach, a white-box approach, or a grey-box approach. A black-box approach is the closest to a real-life situation, where the attacker does not know all the ins and outs of the infrastructure that it is targeting, being needed to use brute force attacks against it to try and find vulnerabilities or weaknesses to exploit by trial and error. Since there is no information given on the source code or software architecture, it takes longer to complete the test, resulting in the need for automated processes to find vulnerabilities (Kalirathinam, 2019; Palma Salas & Martins, 2015). On the other hand, White-box is the complete opposite, as the tester has full knowledge and access to the source code and its architecture, giving the tester a shorter time frame than with a black-box approach; it is also a more direct and complete test to the weaknesses. However, with this, it can be challenging to decide the focus of the test, especially regarding system and component testing and analysis. Also, special tools are needed to analyze and debug the software code. Grey-box combines the black-box approach and the white-box approach, which means that the tester has partial knowledge of the targeted environment, which in most cases is the software code and the system architecture diagrams. In a grey-box test, usually manual and automated testing processes are used, allowing the tester to focus on main areas of the target or specific vulnerabilities, which it knows about, to exploit and attack, making it easier to discover alternative security flaws (Cangea, 2018; Kalirathinam, 2019).

The penetration testing process can be divided into five stages, Planning and Reconnaissance, Scanning, Exploitation, Maintaining Access, and the Analysis of the Results (Bechtsoudis & Sklavos, 2012).

In the Reconnaissance stage, the attacker scouts for necessary information about the target. With this, the attacker gains a foothold on the technologies used in the application and other relevant information that will help him identify some security vulnerabilities. The attacker can even compromise the hosts on which the target relies and then pivot into the target. Thereby, it does not need direct access to its target to exploit it.

In the Scanning stage, the attacker uses the information gathered from recon done previously and with host discovery, content discovery, scanning ports and services, and discovered vulnerabilities. It is possible to select the right endpoint to begin carrying out the exploitation phase. The vulnerabilities found can be considered Positive (if it turns out to be a fundamental flaw), Negative (if it is not an actual vulnerability), False Positive (is when a scanner indicates that there is a vulnerability, but it is not), and False Negative (vulnerabilities not found while doing Recon or Scanning but that exist, it just was not found on the previous stages) (Wang et al., 2016).

In the Exploitation stage, the attacker will try to gain access to its target information and data. This can be done with automated tools, as well as the previous stages. However, manual exploitation techniques are the most appropriate way since a vulnerability can be exploited in a thousand different ways, but each Pen tester has its methods and tools to do it. The vulnerabilities targeted by the attackers can be represented under different terms depending upon the organization. As stated before, OWASP summarises the vulnerabilities upon ten different types, while other platforms rate each vulnerability differently; nevertheless, the vulnerability rating taxonomy is consistently rated by type, severity, and impact (Nagendran et al., 2019).

After the Exploitation stage, the attacker must Maintain Access and its Privilege, with the intent of escalating the privileges he gained access to, to stay in control of the data breach, which can result in access to other user's data, confidential data, and even find out other vulnerabilities for future exploitation. All the incoming requests to a web server will be saved in a log file. Suppose the attacker attains superuser permissions in the web server. In that case, he can delete the log file leaving no trace for him. However, since it is not that easy to attain superuser permissions, it is better to use proxy mechanisms to exploit vulnerabilities (in a Web context).

In the analysis of the results (the final stage), all the relevant data will be analyzed in other to confirm or exclude the Positive/False Positive results. It will then be written a detailed report which includes the name of the vulnerability, the vulnerable endpoint, the technical description of the vulnerability, possible business impact and severity, as well as other additional information that may be asked to the Pen Tester (Mudiyanselage & Pan, 2020).

Figure 5: Penetration Test Stages

While performing penetrations tests, it is possible to choose automated or manual testing methods, with each method having its pros and cons. On one side, automatic testing methods are faster at scanning their targets, enabling the tester to save its workforce and time on the second stage, since there are many locations to scan for, providing a continuous evaluation of data with little effort, on the other side, there are several unique vulnerabilities that cannot be detected by automatic scanners and it is more susceptible to find false negatives and false positives (Singh et al., 2020). As for manual testing methods, it is observed that automated scanners miss exceptional vulnerabilities that testers with experience can locate, as well as the fact that they can find alternative security technics used by developers to reduce the detection of vulnerabilities that prove to be false positives, even though it takes longer to test applications thoroughly, it takes a need of cascaded intelligence to test specific vulnerabilities.

Given the differences among websites and applications systems, the testers may need different tools to perform these tests. However, it was observed that manual vulnerability assessment and penetration testing are more accurate than automated vulnerability assessment and penetration testing, which do not provide 100% accurate results (Nagpure & Kurkure, 2017; Singh et al., 2020).

In order to help the auditor with the analysis of the conducted tests, various intelligent data analysis methods are being developed to simplify and automate the process. Only actions and information that expand the testers' knowledge are included to expose vulnerable objects for these methods. The collected data is then reduced by applying thresholds to the actions and information, eliminating the most inefficient action. The threshold applying can be calculated so that all information is considered. However, this means more data volume for the tester to handle. Based on the data collected, it is possible to compare properties and similarities to estimate relevant assessments to properties by types of attacks and types of vulnerabilities, ranking them by individual properties, vulnerability levels estimate, and similarities. In the end, the analyst should have enough knowledge about the tested system to report it to the analysts(Poltavtseva & Pechenkin, 2017).

2.6 Exploitation on Web Applications

Web application security deals with security against attacks on the application layer. Therefore, the concern is to protect Web Application servers against different security threats that exploit applications' vulnerabilities (Jain & Jain, 2019). The five stages previously referred should be taken into account in order to perform a penetration test. This way, the penetration tester will facilitate exploitation by analyzing the vulnerabilities and weaknesses of the system. The most common vulnerabilities that are widely exploited are SQL Injections, Cross-Site Scripting, and XML injections (Gupta et al., 2020; Nagendran et al., 2019).

2.6.1 SQL Injections

SQL injections are malicious queries that are injected via data input fields in order to access unauthorized data. Since most web applications use SQL Database, it is most common to test injections. Suppose an attacker executes a malicious query and the server performs the attacker requested action due to improper query validation. In that case, the attacker may leverage this to attain administrative rights, letting him have complete access to the database. SQL injections are classified according to 4 different types, Error-based, Union-based, Boolean-based Blind SQL injections, and Stacked Queries (Abdul Raman, 2019).

Error-based SQL injection is a technique that relies on error messages thrown by the database server to receive information about data stored inside the database and the structure of the database. This technique works when the target web application has been configured to disclose error messages a response to errors.

Union-based is an in-band SQL injection technique that leverages the UNION operator to combine the results of two or more SELECT statements into a single result. The combined results are then returned as part of the HTTP response, which could be displayed on the web page.

Boolean-based SQL Injection is a “blind" or inferential SQL Injection technique that relies on sending an SQL query to the database and "forcing" the application to return a different result. Depending on the result, the content within the HTTP response will change or remain the same (Nagendran et al., 2019).

In stacked queries, for each given parameter, a semicolon (;) appended at the end of a SQL statement indicates the end of the statement. SQL scripts that come after the semicolon are parsed as a new SQL statement; this allows the attacker to execute other SQL statements. Stacked queries can be used to execute any SQL statement or stored procedure, in contrast to UNION attacks limited to SELECT and UNION statements (Abdul Raman, 2019).

2.6.2 Cross-Site Scripting

On Cross-site scripting (XSS), the attacker injects a malicious script on the target website. This allows the attacker to execute undesired functions on other users that visit the website. XSS attacks can have severe impacts such as account takeover, credential stealing, data exfiltration, crypto mining, keylogging, fingerprinting, tab-napping, screenshot capture, and others. XSS can be combined with several other vulnerabilities to increase the impact level. The XSS attacks can be classified into three main types, Reflected XSS, Stored XSS, and mXSS. Reflected XSS is a method where the attacker tests various inputs upon the HTML tags to break them and execute his input. Stored XSS is a type of XSS where the attacker supplies input to the Web Application, and after storing this input, if it is not encoded with HTML, it will be served to all the users that visit the application. mXSS refers to Mutated XSS,

which abuses the incorrect reading of inner HTML by the application (Vijayalakshmi & Syed Mohamed, 2020).

2.6.3 XML Injections

Given the fact that most XML processors allow the specification of an external entity such as a Uniform Resource Indicator (URI) that is de-referenced and evaluated during the processing of an XML document, it can be uploaded malicious payload to extract data, scan server's network or cause Denial of Service (DoS) on the servers. (Gupta et al., 2020)

Various technics and algorithms are available, which can provide invaders the opportunity to execute code remotely on the website server, access local resources and confidential files, insertion of malicious content in messages or documents, and even execution of code on the website users system.

2.7 Most Common Tools for Penetration Tests

Nowadays, plenty of penetration testing software are available, some being open-source software and some with costs associated with it, and even some online tools, but from the research done, most testers seem to prefer open-source tools. Some of the tools used in penetration tests are as follows:

- Nmap ("Network Mapper"), an open-source tool for port scanning and OS fingerprinting. Nmap has a scripting engine called Nmap Scripting Engine (NSE), allowing users to write their scripts and automate their tasks. Nmap supports different types of scans to detect and evade various types of IDS and Firewalls (Nagendran et al., 2019; Nmap, 2021).
- Nikto, an open-source web vulnerability scanner, is used to scan for server misconfigurations and insecure files (Jain & Jain, 2019; Sullo, 2021).
- W3af, an automated open-source scanner that uses Python to test more than 200 different vulnerabilities (W. Org., 2021).
- KNOXSS is an online XSS discovery tool based on its own server that is widely used to bypass the Web Application Firewall using custom payloads directly (Logic, 2021).
- Acunetix, a web vulnerability scanner that tests for more than 4500 vulnerabilities (Liu et al., 2017). It is one of the most used testing tools due to the number of vulnerabilities that it can detect, being most of them varieties of SQL injections and cross-site scripting (Invicti, 2021a; Nagpure & Kurkure, 2017).
- Wireshark, which is a protocol and traffic analyzer(*Wireshark*, 2021).
- sqlmap, an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over database servers (Bernardo Damele A. G. & Miroslav Stampar, 2021).

- Kali Linux is an open-source Linux-based operating system with many integrated tools used for penetration tests (EC-Council, 2020; OffSec, 2021).
- Metasploit, a Ruby-based framework used for penetration tests with which it is possible to test security vulnerabilities, execute attacks, enumerate networks. It is also possible to write, test, and execute code for exploitations (EC-Council, 2020; Rapid7, 2021).
- OWASP Zed Attack Proxy, an exploitation and proxy tool that provides support to test the steadiness and security of web applications and websites. By setting up the browser's proxy (manually), it is possible to attack a website just by clicking a button, triggering an active attack containing brute force scanners, proxy interception, active and passive scanners, port scanners, and through web sockets (Nagpure & Kurkure, 2017; OWASP, 2021a).

Due to the number of vulnerabilities that must be tested, most testers use automatic scanners to detect commonly occurring vulnerabilities, being more efficient given the amount of workforce and time needed to detect all flaws and because automated tools are better for scanning a large number of locations where sensitive data may be present (Singh et al., 2020).

Table 4 - Tools used for Penetration Tests

Application Name	References
Nmap	(Cangea, 2018; Denis et al., 2016; Kalirathinam, 2019; Mukhopadhyay et al., 2019; Nagendran et al., 2019)
Nikto	(Nagendran et al., 2019; Teodoro & Serrao, 2011b)
W3af	(Denis et al., 2016; Kalirathinam, 2019; Nagendran et al., 2019; Teodoro & Serrao, 2011a, 2011b; Vijayalakshmi & Syed Mohamed, 2020)
KNOXSS	(Nagendran et al., 2019)
Acunetix	(Abdul Raman, 2019; Kalirathinam, 2019; Krasniqi & Bejtullahu, 2018; Liu et al., 2017; Nagendran et al., 2019; Nagpure & Kurkure, 2017; Teodoro & Serrao, 2011b; Vijayalakshmi & Syed Mohamed, 2020)
Wireshark	(Denis et al., 2016; Gondim et al., 2016; Mukhopadhyay et al., 2019; Vondráček et al., 2018; Wang et al., 2016)
sqlmap	(Abdul Raman, 2019; Cangea, 2018; Kalirathinam, 2019; Liu et al., 2017; Tetskyi et al., 2018)
Kali Linux	(Denis et al., 2016; Mudiyanselage & Pan, 2020; Mukhopadhyay et al., 2019; Tetskyi et al., 2018)
Metasploit	(Alzahrani, 2018; Cangea, 2018; Denis et al., 2016; Kalirathinam, 2019)
OWASP Zed Attack Proxy	(Kalirathinam, 2019; Nagpure & Kurkure, 2017; Singh et al., 2020; Tetskyi et al., 2018; Vijayalakshmi & Syed Mohamed, 2020)
Burp Suite	(Cangea, 2018; Kalirathinam, 2019; Nagpure & Kurkure, 2017)

Nessus	(Kalirathinam, 2019)
Vega	(Kalirathinam, 2019; Vijayalakshmi & Syed Mohamed, 2020)
pentest-tools	(Kalirathinam, 2019)
Google Dorking	(Farah et al., 2015)
Websecurify	(Teodoro & Serrao, 2011a)
NetSparker	(Nagpure & Kurkure, 2017; Teodoro & Serrao, 2011b)
Rips Analyzer	(Mudiyanselage & Pan, 2020)
Pixy	(Mudiyanselage & Pan, 2020)
wpscan	(Tetskyi et al., 2018)
OWASP Live CD Project	(Teodoro & Serrao, 2011b)
WSAttacker	(Palma Salas & Martins, 2015)

2.8 Penetration Testing Methodologies

Methodologies are designed to assure consistent outcomes; this is achieved by a clear, methodical, and systematic approach to testing that ensures a reliable, consistent, and accurate outcome. On the other hand, if a tester follows a very vague or poorly setup methodology, it may leave out essential security flaws that an attacker may find, which would mean a failure to the pentest. The more comprehensive a methodology is, the more comprehensive the outcome will be.

There are three widely used Open-source methodologies used for testing, Open-Source Security Testing Methodology Manual (OSSTMM), Open Web Application Security Project (OWASP) Testing Methodology, and Penetration Testing Execution Standard (PTES).

OSSTMM was developed by the Institute for Security and Open Methodologies (ISECOM) and offers a detailed testing plan, metrics (for assessing security level), and recommendations for the final report. In this methodology, there are five main topics to explore, Human Security (aspects that deal with direct interaction between humans), Physical Security (any physical element of security that is operated mechanically or physically), Wireless Communication, Telecommunications, and Data Networks (which means the security of corporate networks and Internet connections) (ISECOM, 2021). OSSTMM is mostly a methodology used for tests that focus on a company's entire telecommunication and network infrastructure.

OWASP Testing Methodology, on the other hand, is much more focused on Web Applications and services testing, focusing more on the core testing phases of web applications security testing instead of complete coverage for a pentest. This methodology uses mostly black-box methods for testing and

is divided into four main phases, Information Gathering (covering exposure assessments and deployment fingerprinting), Configuration and deployment management testing (for the evaluation of the server security configurations), Web application security testing (with steps for testing different and specific web apps vulnerabilities) and Reporting (which is the final phase of any penetration test) (OWASP, 2021b).

The Penetration Testing Methodologies and Standards developed by PTES is a standard consistently developed by information security experts from various industries that provide a minimum baseline on the requirements of a pentest and how to conduct it. It consists of seven main sections that cover from the initial communication and reasoning for the pentest to the final report, defining a guide on the whole process which provides the most value for its users. The main sections are the Pre-engagement interactions, Intelligence Gathering, Threat Modelling, Vulnerability Analysis, Exploitation, Post Exploitation, and Reporting. In the Pre-engagement section, it is defined the scope of engagement and the tools required. After the initial section, it comes to the Intelligence Gathering where the tested organization provides general information about in-scope targets, and the tester gathers all the available information from publicly accessible resources, which leads to the Threat Modelling section, on which the tester will look to find vulnerabilities in the system, either through manual or automatic tools. Vulnerability Analysis consists of identifying, validating, and evaluating the security risks associated with the previously found vulnerabilities to find flaws in the system that could be exploited in the fifth phase, the Exploitation to attempt to breach the system and its security. After the exploitation, there is the Post Exploitation phase where must be considered the value of the compromised system and its usefulness and scalability, finishing off with the Reporting phase in which an executive and technical reports are issued and delivered, covering what was tested, how it was tested the vulnerabilities found, and how were they found, guiding the organization to better their security practices (PTES, 2014).

2.9 Conclusion

The work conducted in this chapter aimed to identify the existing tools and methodologies to test the security of web applications (through the usage of penetration testing methodologies and techniques). A consensus was found among most studies in terms of methodology stages since most researchers recommend using the five-stage method. On the other hand, it was also explicit from the literature that most web applications penetration testers mostly use automated tools to scan and recognize vulnerabilities. However, since there are many tools, it was not possible to find unanimity about the most recommended tools for these tasks. As for the exploitation stage, most testers use manual methods to achieve their goals. As it was verified, the number of false positives was lower than with automated methods, resulting in more concise results.

Chapter 3 – Design and Development: Generic Web Application Penetration Testing Methodology

In this chapter, it will be presented the developed methodology. The design of this methodology was influenced by the research made on state of the art. This methodology aims to give a complete guide for any tester to apply during the assessment of the security of any web application through a penetration test, leading to consistent results while taking a generic approach to the application. The methodology is divided into seven main stages, Planning stage, Reconnaissance stage, Scanning stage, Vulnerability Analysis stage, Exploitation stage, Analysis of Results stage, and Reporting stage. Each stage is divided into sub processes which should be followed according to the flow present in the following figure.

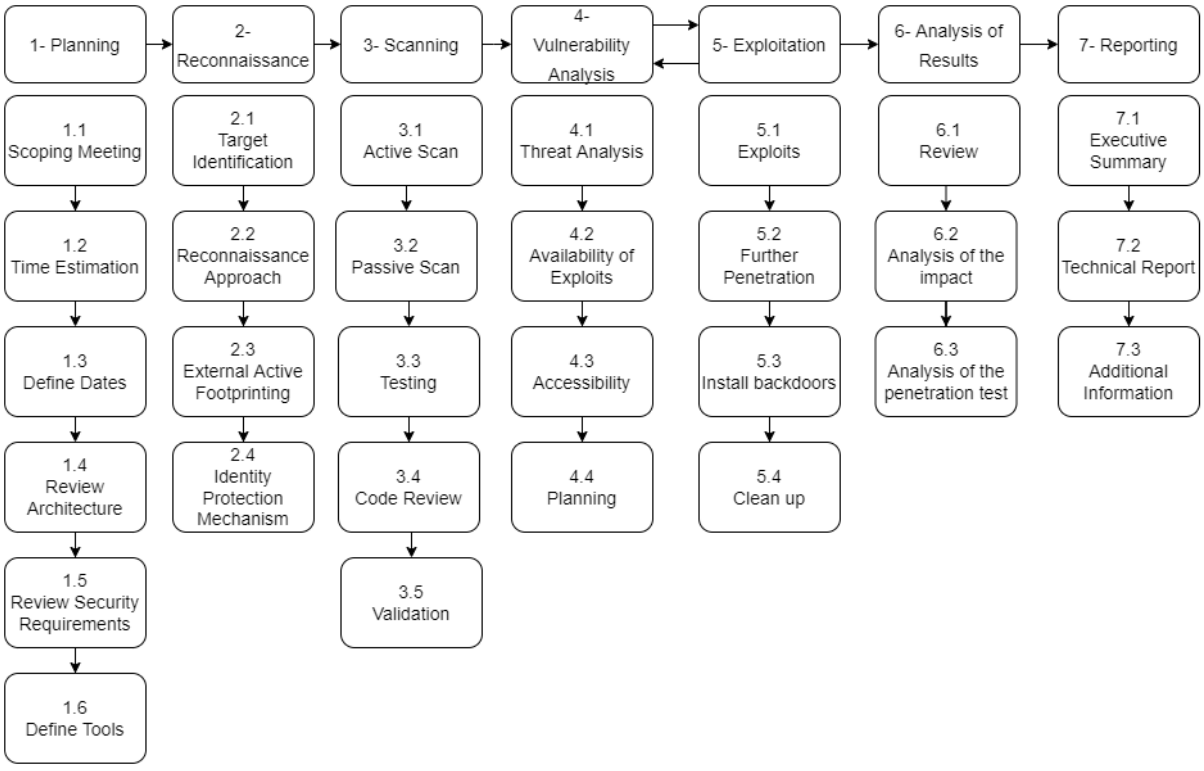

Figure 6: Main Stages of the Methodology

This methodology covers every stage that a web application penetration test is supposed to have, from the initial scoping meeting that will lead to the beginning of the test through the reconnaissance and scanning stage on which the application interaction starts by gathering information to better understand the target's infrastructure, through the vulnerability analysis and exploitation stage where the tester will

try to explore entry points and security flaws to exploit and gain access of the target, finishing the process with the analysis of results and reporting stage, where the tester reviews all the project, procedures, and artifacts to develop a final report on which the entire process is described in a manner that makes sense to the executive management and technical staff.

Following are the main stages and sub-processes defined by this methodology, the Inputs, Tools, Techniques, and Outputs represented according to the PMP ITTO(OSP International LLC, 2021) (Project Management Professional – Inputs, Tools, Techniques, and Outputs), as well as the Process Interaction within each stage. The inputs represent any item (internal or external to the project) required by a process before it proceeds, the tools represent tangible artifacts like software programs or templates, the techniques represent systematic procedures employed by human resources, and the outputs represent a result generated by the process.

3.1 Planning

The first stage of this methodology is the Planning stage. This stage aims to define and plan the penetration test by defining the scope, understanding the rules of engagement, getting to know more about the requirement for the test, and defining the tools needed accordingly to the goals and dates (see figure 7).

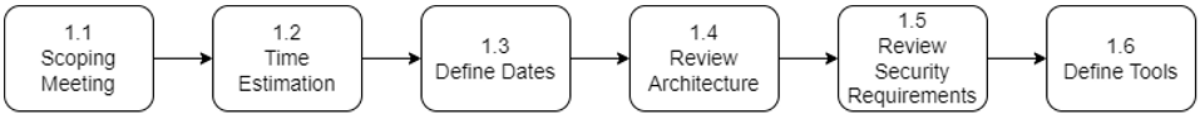

Figure 7: Process 1 - Main Procedures of the Planning Stage

3.1.1 Scoping Meeting

A scoping meeting is a meeting made between the testers and the customer on which the scope of the test should be defined, and additional information about the system and internal procedures should be presented (see figure 8).

Figure 8: Process 1.1 - Scoping Meeting (ITTO)

During a scoping meeting, the following steps should be taken (see figure 9):

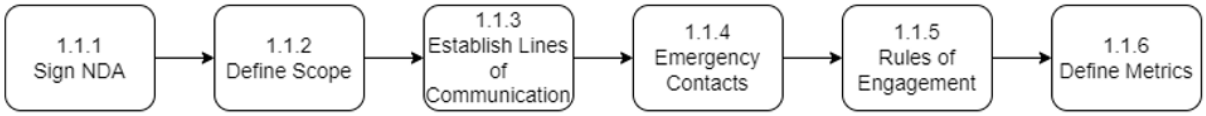

Figure 9: Process 1.1 - Sub-processes of the Scoping Meeting

3.1.1.1 Sign NDA

An NDA is both parties signed document on which the tester compromises himself into not sharing any private or sensitive information in any form (see figure 10). This document is usually signed before any scoping discussion occurs, mainly to protect the company's privacy.

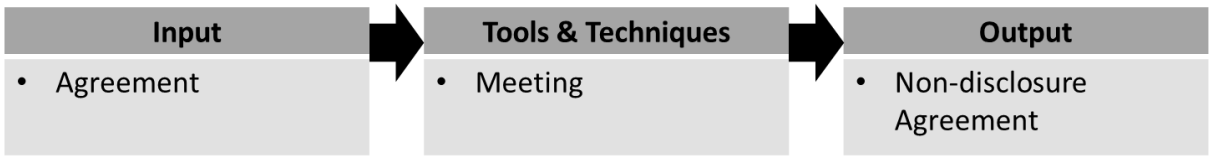

Figure 10: Process 1.1.1 - Sign NDA (ITTO)

3.1.1.2 Define Scope

This is one the most critical parts of the planning stage as it is when it is defined on what is to be tested (see figure 11). During this part of the scoping meeting, a tester should try to be as exhaustive as possible to explicitly define the scope to avoid scope creep (how a project's requirements tend to increase during the project development). The goals for the pentest must be specified, as well as ranges and domains, targets information or technical details (if allowed), and additional information, such as limitations and requirements. The tester should consider the complexity of the established requirements for the test while defining the scope. The amount of information disclosed is directly connected to the application's tester approach (black-box, grey-box, or white-box).

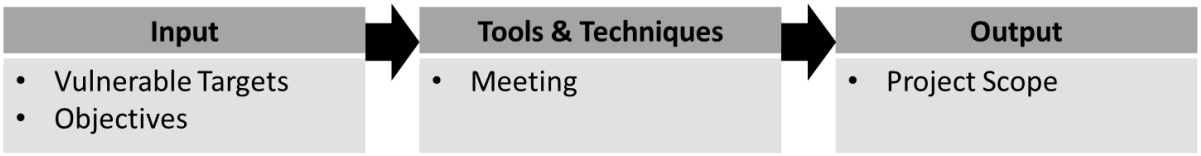

Figure 11: Process 1.1.2 - Define Scope (ITTO)

3.1.1.3 Establish Lines of Communication

In order to keep the customer informed, lines of communication should be established, as well as the regularity of the meetings (see figure 12).

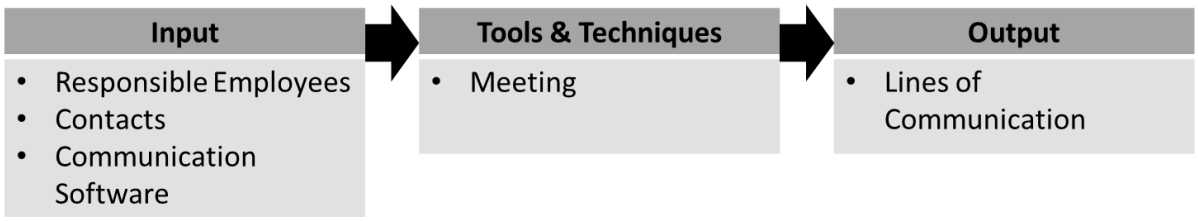

Figure 12: Process 1.1.3 - Establish Lines of Communication (ITTO)

3.1.1.4 Emergency Contacts

It is essential to establish emergency contacts as errors happen, and it is better to be safe than sorry. For emergency contacts, a tester should prepare a list with the contacts, schedules of availability, and forms to secure data transfer in case of need (see figure 13).

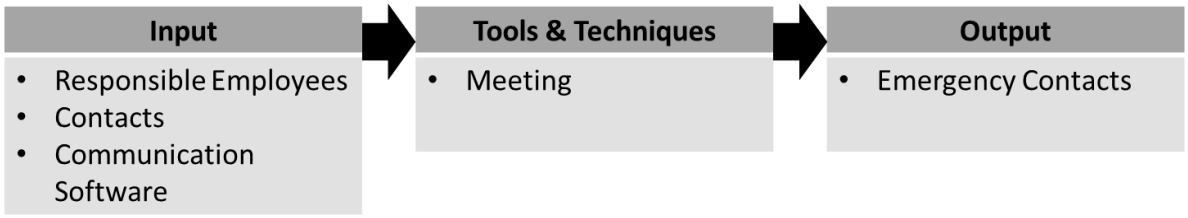

Figure 13: Process 1.1.4 - Emergency Contacts (ITTO)

3.1.1.5 Rules of Engagement

On the scope, it is defined what is going to be tested, and, in this part, it is defined on how the testing should occur. For this, both parties should decide on the procedure for evidence handling, timeframe for target tests, permissions to test, and approach limitations (specific methods that should not be applied) (see figure 14). This set of rules ensures that the client's system is not subjected to unnecessary risks by the tester's actions and ensures that the exploitation and compromise of the application is conducted accordingly to the customer's needs.

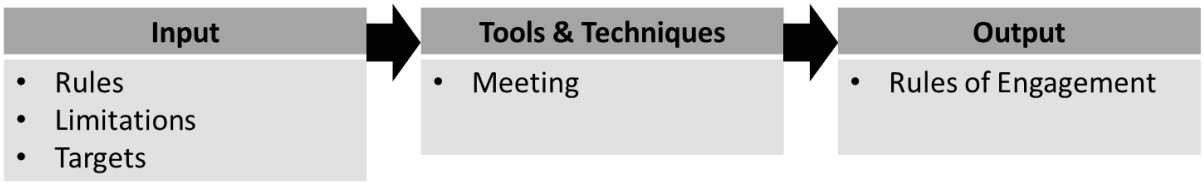

Figure 14: Process 1.1.5 - Rules of Engagement (ITTO)

3.1.1.6 Define Metrics

As Peter Drucker famously says, "if you cannot measure it, you cannot manage it". Measuring results and procedures is crucial for the results of the penetration test. Measures will help the tester, and the

customer determine and understand how much progress is being made during the process. With the defined measures, a tester can later use that information and generate trend data that may help him better estimate timeframes for tests. Usually, metrics define the percentage for coverage of the targets, percentage of false-positive vulnerabilities, vulnerabilities per target, time spent per target, the severity of vulnerabilities, and as much as needed (see figure 15).

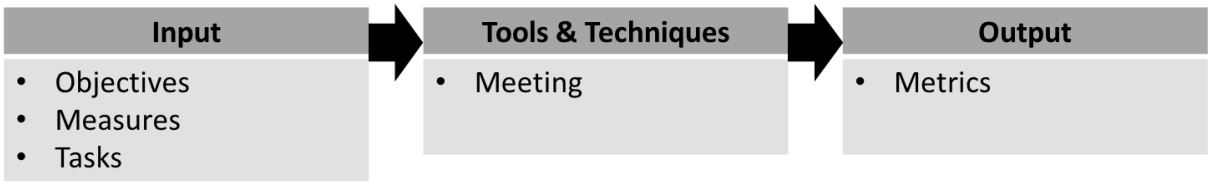

Figure 15: Process 1.1.6 - Define Metrics (ITTO)

3.1.2 Time Estimation

After the scoping meeting, the tester must consider all the information to decide on a time estimation for each task (see figure 16). The time estimation is directly related to a tester's experience and knowledge since the experience will allow the tester to take less time with specific tasks that he may be more comfortable with. While defining the time estimation, a tester should be prudent and count with more time than he expects; if there is any mishap during the tests, he will still have a margin for the extra time.

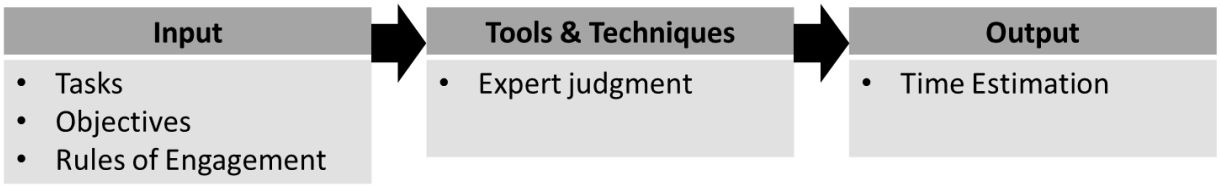

Figure 16: Process 1.2 - Time Estimation (ITTO)

3.1.3 Define Dates

Define a start and an end date explicitly (see figure 17). It not only protects the tester from scope creep, but it also protects the client from delays on the results. The defined dates must consider the time estimations previously established.

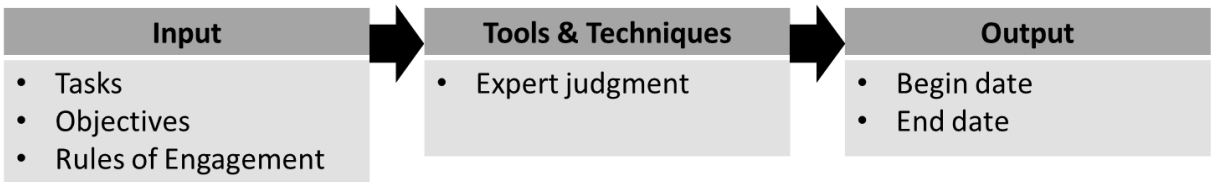

Figure 17: Process 1.3 - Define Dates (ITTO)

3.1.4 Review Architecture

Suppose any information on the target architecture is provided. In that case, it should be reviewed to have a better understanding of the infrastructure and some of the possible entry points (see figure 18).

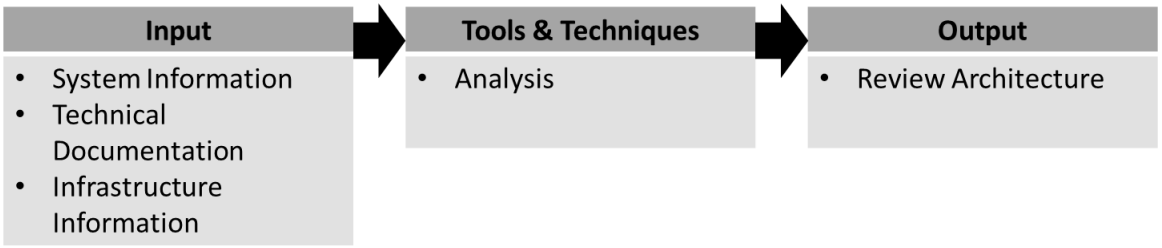

Figure 18: Process 1.4 - Review Architecture (ITTO)

3.1.5 Review Security Requirements

The security requirements dictate how an application works from a security perspective., this means understanding user management, authentication procedures, authorization procedures, data confidentiality, and session management (see figure 19). If provided, this information will ease the process of access; if not, a tester should find a way to discover security flaws in the security mechanisms.

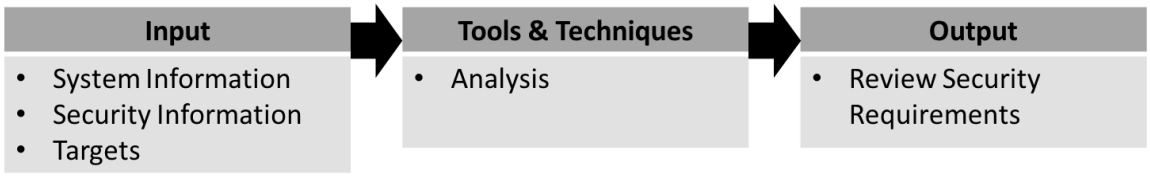

Figure 19: Process 1.5 - Review Security Requirements (ITTO)

3.1.6 Define Tools

For the decisions regarding the tools that the tester will use, he must consider the scope, the rules of engagement, his limitations, and user experience. Specific tasks may require specific tools, so for this reason, a tester should always be able to go for automatic scanners and exploits or go for manual inspection and customized exploits. If there are no limitations from the customer, this part of the planning stage is totally in charge of the tester (see figure 20).

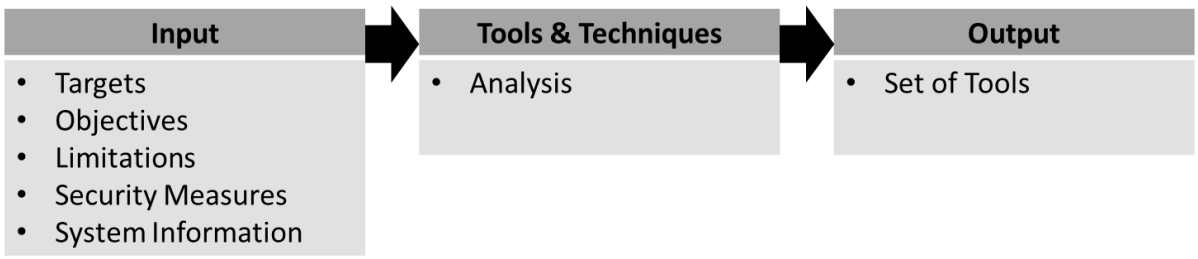

Figure 20: Process 1.6 - Define Tools (ITTO)

3.2 Reconnaissance

The second stage of this methodology is the Reconnaissance stage. In this stage, the tester should consider the goal and the rules of engagement and gather basic but specific information on the targets (see figure 21). For this, a tester can use any type of reconnaissance approach towards the targets to identify additional information about them and the protection mechanisms that may be affiliated with them.

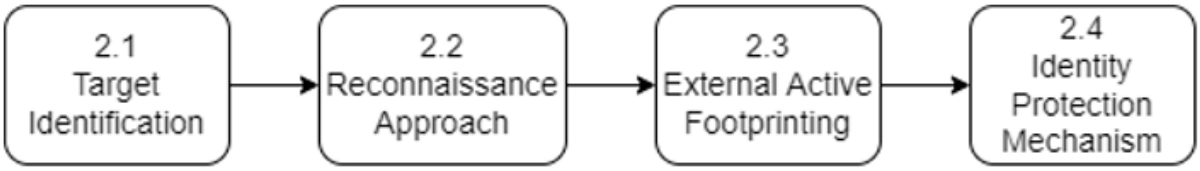

Figure 21: Process 2 - Main Procedures of the Reconnaissance Stage

3.2.1 Target Identification

Even though most information should be disclosed during the scoping meeting, a tester should always identify his targets (see figure 22). This may help identify targets that were not part of the initial scope or even find additional information about services or protocols used. In order to identify his targets, a tester should consider the system architecture and test his connection to them.

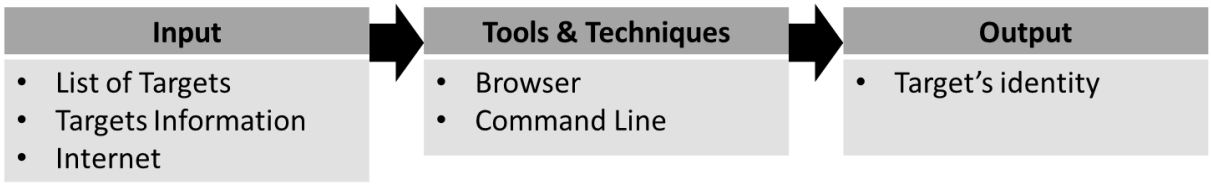

Figure 22: Process 2.1 - Target Identification (ITTO)

3.2.2 Reconnaissance Approach

While performing the reconnaissance on the targets, a tester can have three approaches: a passive, semi-passive, and active approach (see figure 23).

Figure 23: Process 2.2 - Reconnaissance Approach (ITTO)

The three types of approach can and should be used while performing reconnaissance on the system (see figure 24).

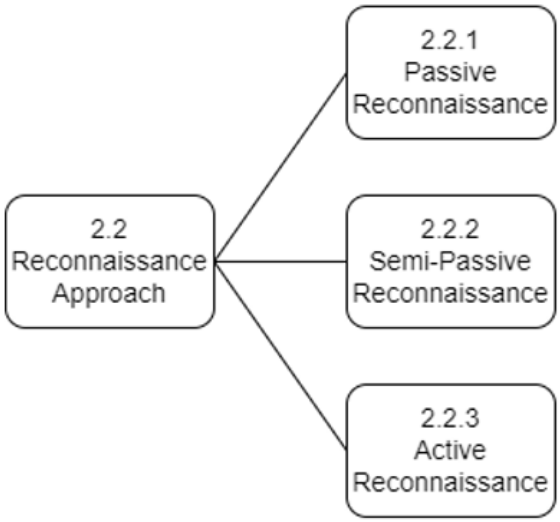

Figure 24: Process 2.2 - Sub-processes of the Reconnaissance Approach

3.2.2.1 Passive Reconnaissance

In a passive approach towards reconnaissance, a tester can only gather and use archived or stored information (see figure 25). No traffic should be sent to the target as it is supposed to gather information without being detected. This should not be a general procedure, but specific tasks or customers may require this approach.

Input	Tools & Techniques	Output
• Archived information • Stored information	• File Explorer	• Passive Reconnaissance

Figure 25: Process 2.2.1 - Passive Reconnaissance (ITTO)

3.2.2.2 Semi-Passive Reconnaissance

While performing reconnaissance with a semi-passive approach, a tester's goal is to gather information to profile the target without compromising himself, sending what would appear to be regular traffic or behavior (see figure 26). A tester should not actively seek confidential information but published details that may help profile the targets.

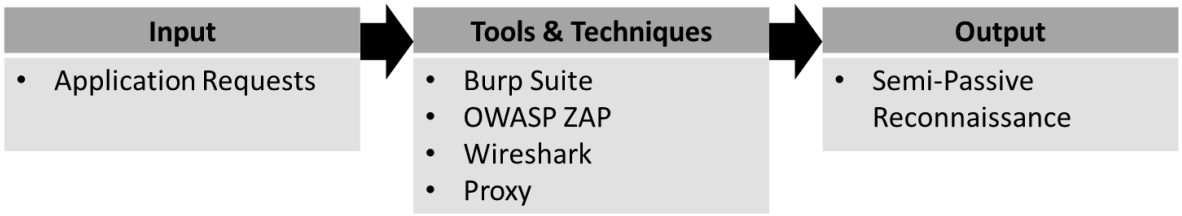

Figure 26: Process 2.2.1 - Semi-Passive Reconnaissance (ITTO)

3.2.2.3 Active Reconnaissance

The target should detect active reconnaissance as a tester is actively mapping the network structure, scanning for ports, open services, unpublished directories, files, and servers, and if possible, gathering information about database systems and services running in the background. The goal of this approach is to enumerate and map the targets as much as possible (see figure 27).

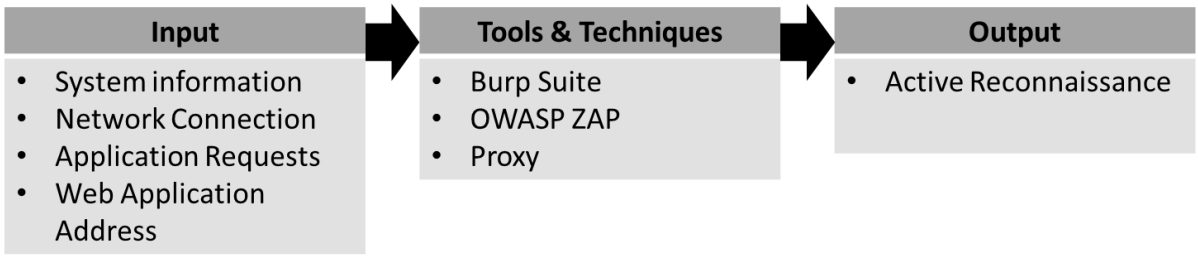

Figure 27: Process 2.2.2 - Semi-Passive Reconnaissance (ITTO)

3.2.3 External Active Footprinting

External Active Footprinting is similar to active reconnaissance but from an external perspective. The reconnaissance should be done as if the tester was outside the organization, interacting with the targets to gain information (see figure 28). This can be achieved through different methods such as port scanning, DNS discovery, forward or reverse DNS, DNS Bruteforce, web application discovery, and host detection and enumeration. All these methods will help the tester better prioritize the targets according to the information gathered about them.

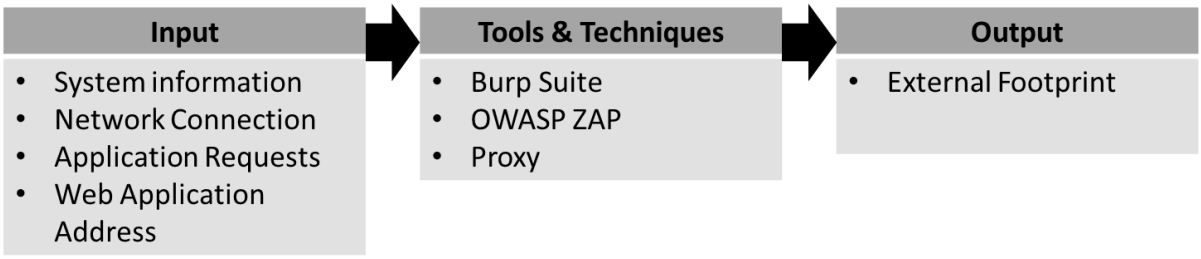

Figure 28: Process 2.3 - Semi-Passive Reconnaissance (ITTO)

3.2.4 Identity Protection Mechanism

Accordingly to the target, the protection mechanisms should be identified and mapped to maximize the efficiency of the tests and minimize the detection ratio (see figure 29). These protection mechanisms can be network-based, host-based, at the application level, or even mechanisms to protect users and data storage.

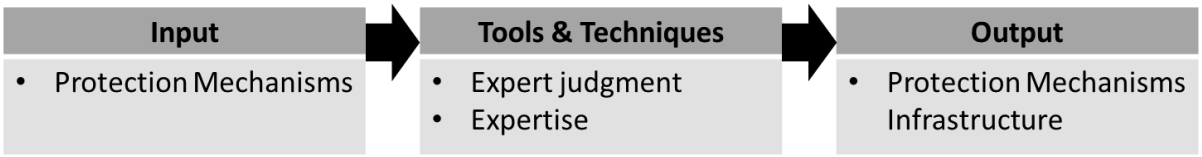

Figure 29: Process 2.4 - Semi-Passive Reconnaissance (ITTO)

3.3 Scanning

The third stage of the methodology is the scanning stage. In this stage, the tester should take a more active role in the information gathering process, taking into account previous targets disclosed information to start searching for new vulnerabilities. A tester should scan the application for entry points and flaws during this process, then test and validate the findings (see figure 30). In this stage, the tester starts to have a more active role in testing, dealing directly with the application and its services.

Figure 30: Process 3 - Main Procedures of the Scanning Stage

3.3.1 Active Scan

A tester should take a more active role in scanning during the scanning stage, interacting directly with target components. This can be done with automated or manual tools to identify and evaluate the security regarding possible vulnerabilities (see figure 31 and figure 32).

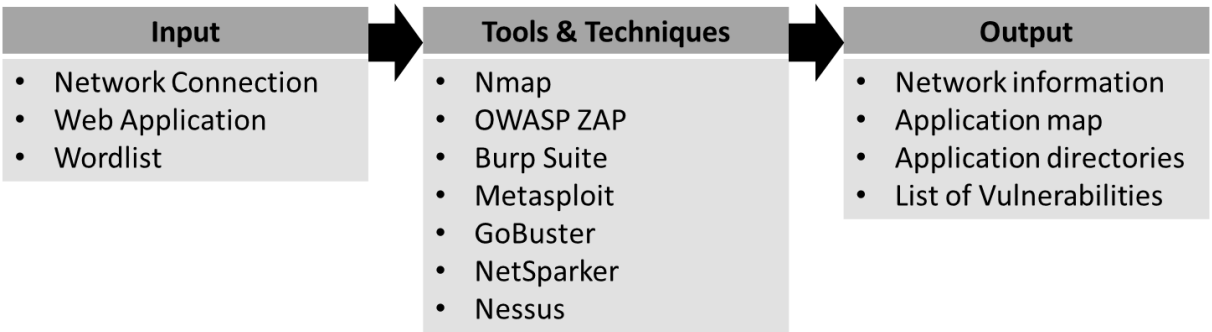

Figure 31: Process 3.1 – Active Scan (ITTO)

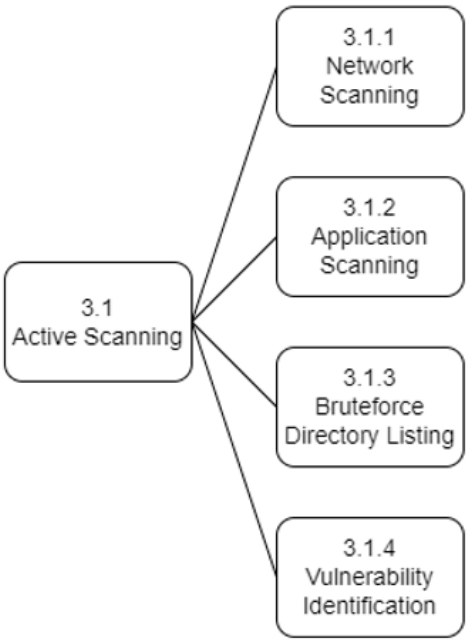

Figure 32: Process 3.1 - Sub-processes of the Active Scan

3.3.1.1 Network Scanning

Network scanning is mainly done through the appliance of automated tools to help the tester obtain a basic overview of what may be available on the target network (see figure 33). This step towards the scan of the application usually starts on the reconnaissance stage, while performing the port scan; however, in this stage, it is intended to do a more in-depth analysis of the network, searching for misconfigurations on the network, vulnerabilities in Voice over IP technologies, test for vulnerabilities on the services and protocols used.

Input	Tools & Techniques	Output
• Network • Network Access	• Nmap	• Network information

Figure 33: Process 3.1.1 – Network Scanning (ITTO)

3.3.1.2 General Application Scanning

General application scanning is a method used to find general flaws in the application. Either through manual or automatic crawling, it is possible to find flaws in the application (see figure 34). While performing the scan, a tester may find form fields to attempt SQL injections or XSS. While crawling the application, it is also possible to find sensitive information in error details or on the Viewstate. The most common tools used for this are Burp Suite (PortSwigger, 2021b), Nessus (Tenable, 2021b), and

OWASP ZAP (OWASP, 2021a), since they all have built-in crawlers that scan the application for vulnerabilities.

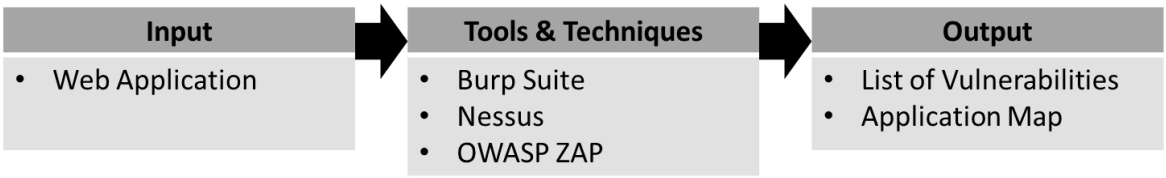

Figure 34: Process 3.1.2 – General Application Scanning (ITTO)

3.3.1.3 Bruteforce Directory Listing

Bruteforce directory listing is a method used to find directories that were not found on previous steps of the penetration test. A scanner will search for common and reachable directories. However, with brute force directory listing, a tester might use standard wordlists or even customize wordlists regarding terms used in the application (found during previous scans and reconnaissance) to find administrative or sensitive directories, extending the engagement attack field (see figure 35). Sometimes this procedure may cause a crash on the application or inundate the webserver with requests, causing a DDOS. Some of the most common tools used for this are Nmap (Nmap, 2021), NetSparker (Invicti, 2021b), and GoBuster (Christian Mehlmauer, 2021).

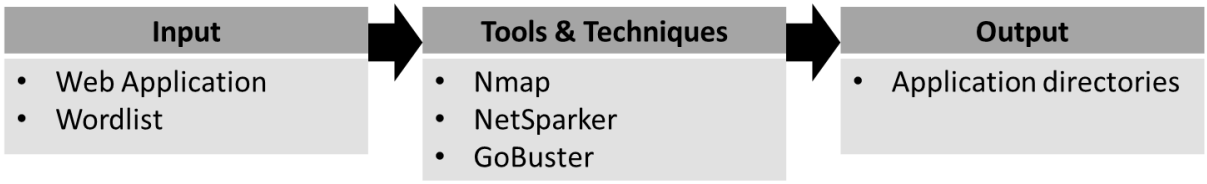

Figure 35: Process 3.1.3 – Bruteforce Directory Listing (ITTO)

3.3.1.4 Vulnerability Identification

While performing the previous procedures for the active scan of the application, a tester might come across new vulnerabilities within the scope. An example of this situation is when the tester performs a general application scan and finds form fields vulnerable to SQL injections. All the vulnerabilities found must be analyzed and identified in order to exploit them in later stages (see figure 36). For this, a tester should understand how the vulnerability was reached and found, how the vulnerability works, and what kind of impact its exploit may have.

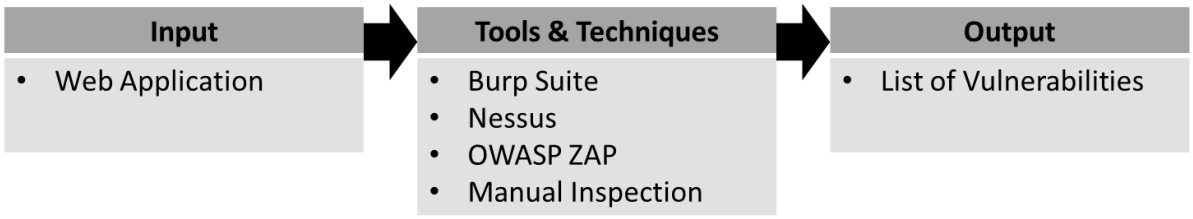

Figure 36: Process 3.1.4 – Vulnerability Identification (ITTO)

3.3.2 Passive Scan

A passive scan is when a tester scans the application taking a passive approach towards it (see figure 37).

Figure 37: Process 3.2 – Passive Scan (ITTO)

It is mainly done using a combination of automatic and manual tools as the automatic tools seek to find and gather information, and the manual tools will help the tester analyze and assess the outputs (see figure 38).

Figure 38: Process 3.2 - Sub-processes of the Active Scan

3.3.2.1 Traffic Monitoring

For passive scanning of the application, a tester can monitor the traffic, which could help determine the specifics of an operating system or device. A tester may find misconfigurations, unsecured data transfer, or even capture sensitive information during this process (see figure 39). The most common tools used

for this are Wireshark (*Wireshark*, 2021), Tcpdump (The Tcpdump Group, 2021), and WinDump (Riverbed Technology, 2021).

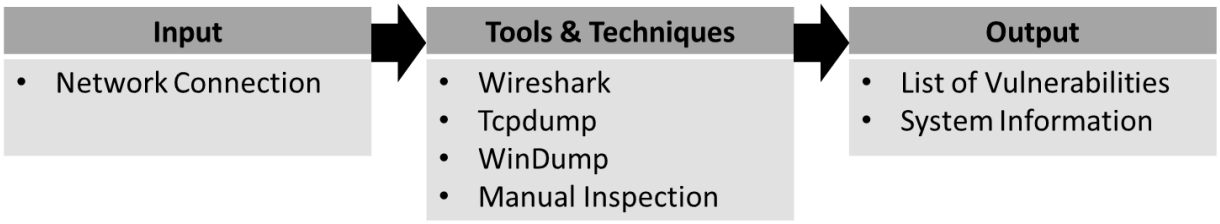

Figure 39: Process 3.2.1 – Traffic Monitoring (ITTO)

3.3.2.2 Metadata Analysis

Metadata analysis consists of analyzing and evaluating the data that describe data instead of the data itself. For example, when examining a file, the metadata information may contain details such as the document author, when the document was created, and other information that can include custom metadata. Within this metadata, it is possible to find internal addresses and paths to the server, IP addresses, and other information, facilitating testers to gain additional access or information to discover new entry points (see figure 40). This is mainly done manually, but tools such as FOCA (Josep, 2021) or MetaCrawler (Metacrawler, 2021) are accessible for testers to use.

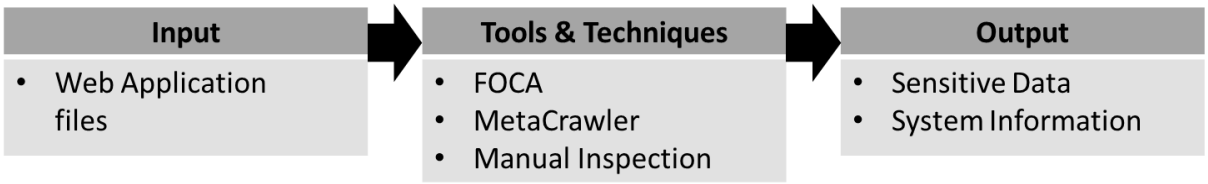

Figure 40: Process 3.2.2 – Traffic Monitoring (ITTO)

3.3.3 Testing

Vulnerability testing is the process of discovering vulnerabilities and security flaws associated with the application. This can be done while performing an active scan of the application and must always be tailored to the test requirements, the rules of engagement, and the end goal. For this procedure, a tester will look into vulnerabilities within the established depth and breadth of the requirements, ensuring that the assessment results meet the expectations regarding it (see figure 41). This should be done using active methods and tools, helping the tester verify the authenticity of the findings. It is common to use tools such as Nessus, OWASP ZAP, or Burp Suite for this procedure.

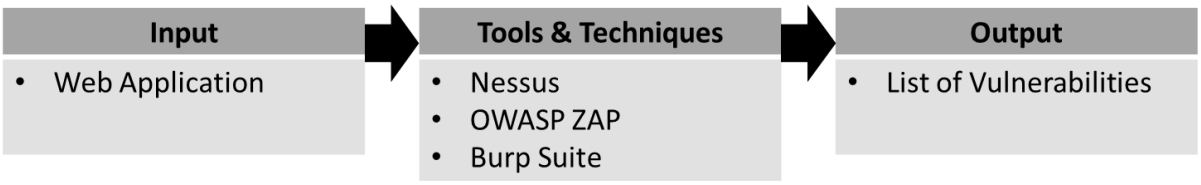

Figure 41: Process 3.3 – Traffic Monitoring (ITTO)

3.3.4 Code Review

For the scanning stage, it is also essential to manually inspect and review the code associated with the application, the source code, or the analysis of requests made to the application (see figure 42).

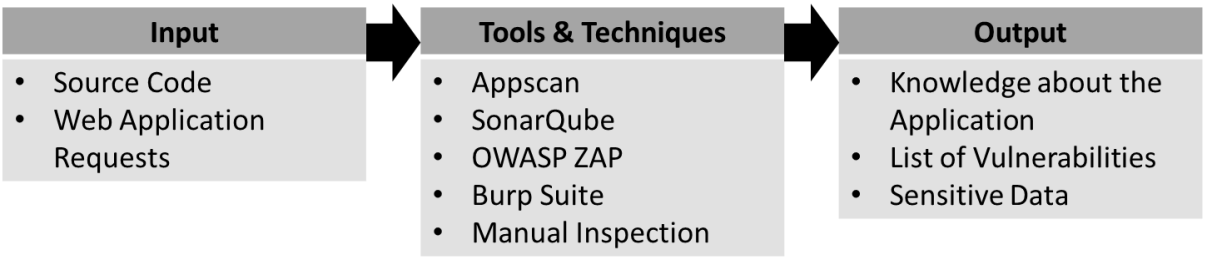

Figure 42: Process 3.4 – Traffic Monitoring (ITTO)

Depending on the approach towards the pentest, a tester may be authorized to look into the application's source code (in case of white-box testing), which, even though it may be an extensive procedure, will help the tester to understand the workflow of the application better (see figure 43). It is also possible to review with automatic tools, but most testers would agree that there is no substitute for manual inspection.

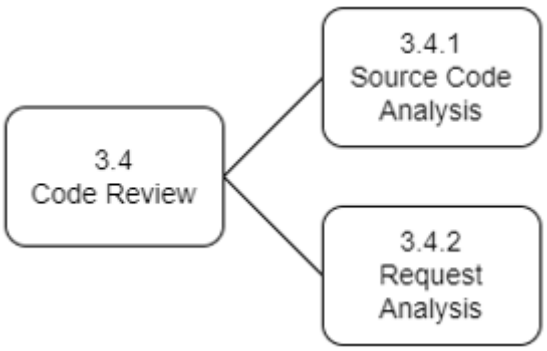

Figure 43: Process 3.4 - Sub-processes of the Code Review

3.3.4.1 Source Code Analysis

Source code analysis is the process by which the tester will check the application's source code in search of security issues as many security flaws may be undetected by vulnerability scanners. Any information associated with security flaws is always in the source code. With access to the source code, a tester can

accurately determine all the procedures within the application and remove the guesswork around the security tests. Findings are variable and may be related to flawed business logic, concurrency problems, easter eggs, or even cryptography, being these the most harmful vulnerabilities in web applications (see figure 44).

The analysis of the source code might be an extensive procedure since some of it should be done through manual inspection. However, with the help of tools like SonarQube (SonarSource, 2021) or Appscan (HCL Software, 2021), this process can be facilitated as these tools analyze and review the code, searching for coding errors, security vulnerabilities, and design flaws, offering remediation measures to assure stable and secure code.

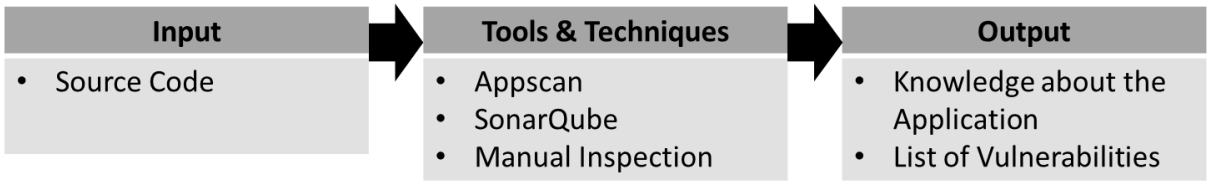

Figure 44: Process 3.4.1 – Souce Code Analysis (ITTO)

3.3.4.2 Request Analysis

Another way of finding sensitive information or vulnerabilities is observing, analyzing, and manipulating requests made between the application and the browser. Requests such as GET HTTP requests may often disclose sensitive data that may be (or not) cryptographed, allowing the tester to unveil new gateways to the application (see figure 45). This analysis can be done using Burp Suite (using the Intruder) or with OWASP ZAP, as both software have a configurable proxy to intercept requests, as well as plugins to help the tester transform encoded or raw data into usable information.

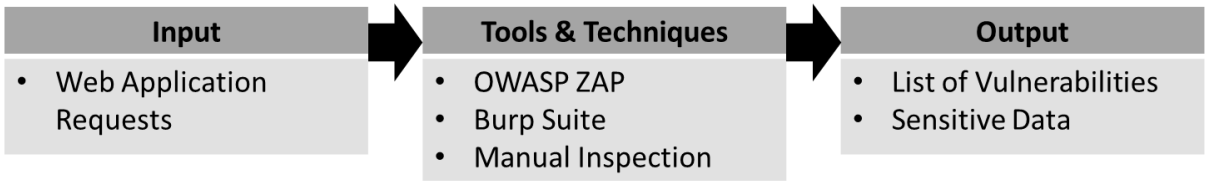

Figure 45: Process 3.4.2 – Request Analysis (ITTO)

3.3.5 Validation

The validation of the findings will allow the tester to do a better correlation between the chosen tools and their limitations. This step should be considered during most penetration tests since it is critical for the results to validate every input and output of the performed tasks and their results. This procedure is crucial to help the tester reduce the number of identified vulnerabilities to only those that are valid (see figure 46). This can be done either by testing the vulnerabilities or by inspection of the related flaw.

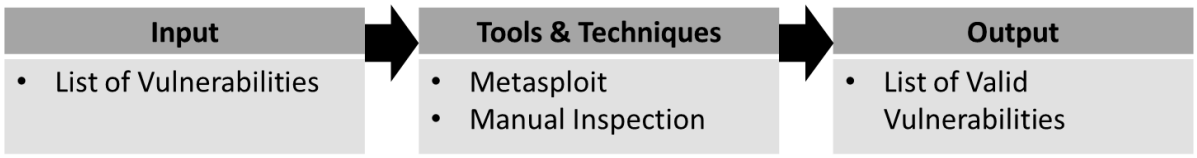

Figure 46: Process 3.5 – Validation (ITTO)

3.4 Vulnerability Analysis

The fourth stage of the methodology is the vulnerability analysis stage. In this stage, the tester will focus his effort on identifying and evaluating the findings and establishing a path of execution for the exploitation and testing of the vulnerabilities found (see figure 47). There are no software requirements for this methodology stage since it should be mainly done through process reviews, manual inspection, and public research.

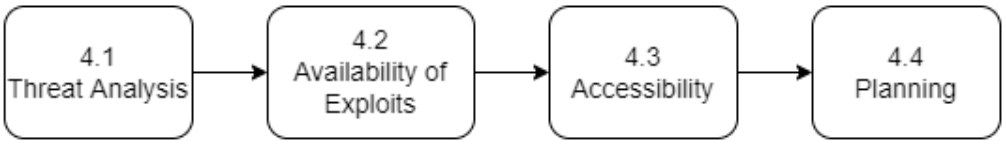

Figure 47: Process 4 - Main Procedures of the Vulnerability Analysis Stage

3.4.1 Threat Analysis

After performing all the previous scanning procedures, a tester should do his research on all findings. By doing this, the tester will have a better insight into how to conduct the exploitations, the impact they may have, vulnerabilities associated with the findings, and the limitations within the rules of engagement and the defined scope. In most cases, vulnerabilities can be found while doing public research on the issues. However, sometimes, a tester needs to set up a replicated environment to make a more in-depth analysis of the situation. The goal of threat analysis is to allow the tester to identify potential vectors of attack regarding the final goal of the tests and consider its impact. Online platforms such as NVD (National Vulnerability Database), OSVDB (Open Source Vulnerability Database), Security Advisories, and Issue Trackers are great vulnerability databases that allow testers to look and find more information regarding the analyzed vulnerabilities. CVE (Common Vulnerabilities and Exposures) is also a great source of information regarding vulnerabilities found according to system components, types of vulnerabilities, and the CVE numbers that make research more accurate since this represents the identifier for specific vulnerabilities (see figure 48).

Figure 48: Process 4.1 – Threat Analysis (ITTO)

3.4.2 Availability of Exploit

While performing a vulnerability analysis, a tester should also consider his capability to either obtain or develop exploits or payloads needed to test the environment. Not only by analyzing the availability and accessibility of such exploits or payload but also by considering the usage of third parties and the customization of specific methods to assess the veracity of the findings. This step is crucial for planning the exploitation stage as this will help the tester confirm his findings and better understand how to take advantage of certain design flaws (see figure 49). In terms of custom exploits, a widely used software is Metasploit which contains a collection with a constantly increasing number of exploits available for any tester to use.

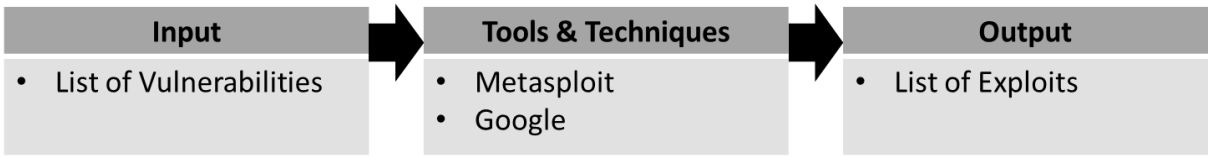

Figure 49: Process 4.2 – Availability of Exploit (ITTO)

3.4.3 Accessibility

The accessibility analysis consists of accessing specific vulnerabilities or entry points to establish a precise scenario and execution path for the exploitation (see figure 50). Considering the protection mechanisms and the workflow of the targets, a tester should define all the requirements and needs to access the entry point.

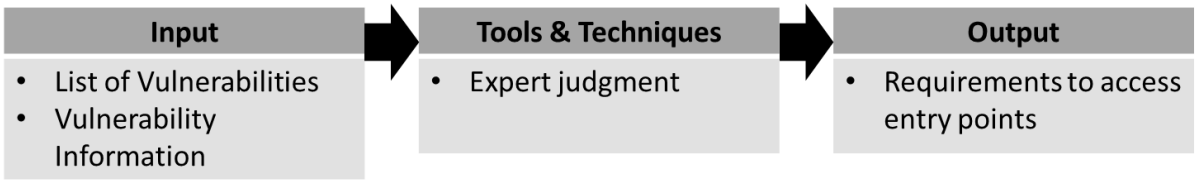

Figure 50: Process 4.3 – Accessibility (ITTO)

3.4.4 Planning

Taking all of the previous tasks into consideration, the planning of the following steps should have a clear and concise output. Considering the scope and the rules of engagement, as well as all the findings, allow the tester to determine a path of execution for his exploits with as much information as possible, detailing the goals of exploitation, all of the entry points, how to access them, what payloads or exploits to use, the tools and methods to use and how to configure the tools given the established goals (see figure 51).

Figure 51: Process 4.4 – Planning (ITTO)

3.5 Exploitation

The fifth stage of the methodology is the exploitation stage. This stage's primary goal is to establish access to the targets by bypassing security restrictions. Suppose all of the previous stages were performed correctly. In that case, this stage should be pretty straightforward since most of the entry points are established, the approach and engagement are planned, and all additional information was considered for the appliance of payloads and exploits. If the attack vector is well established, there will be a reasonable probability of success according to the defined goals. This stage may lead to a loop between the exploitation stage and the vulnerability analysis stage since new vulnerabilities can be found while performing exploitation on previous findings or while performing further penetration and escalation of permissions (see figure 52).

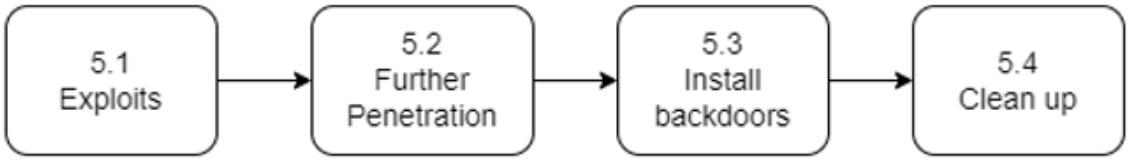

Figure 52: Process 5 - Main Procedures of the Exploitation Stage

3.5.1 Exploits

While performing exploitation, the attack vectors should be precise and evasive. This means that since the whole process aims to simulate an actual attack, the exploitation stage is the appliance of all of the

accumulated research done on the target, combined with evasive measures so a tester can execute precise attacks to the target without being detected during the penetration test (see figure 53). Sometimes is not possible to take evasive measures in exploits of brute force or flooding, but extra cautions need to be taken to try and hide the testers' identity.

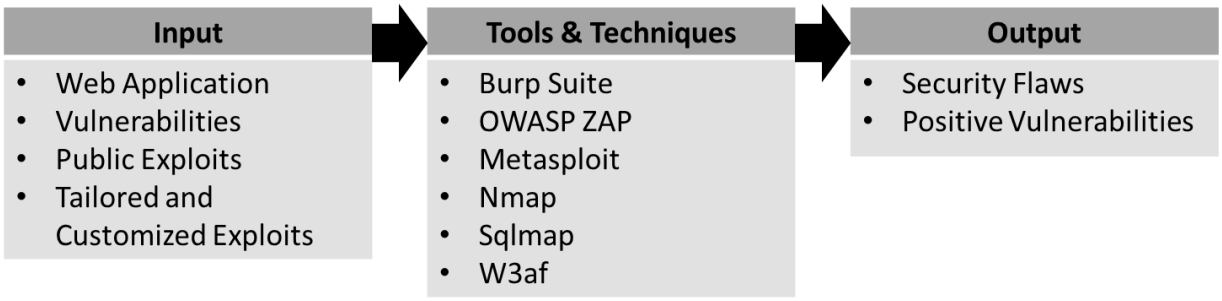

Figure 53: Process 5.1 – Exploits (ITTO)

There are mainly two types of exploits, the public, and the tailored and customized exploits; both have their applicability and should be used according to the needs of the tester (see figure 54). There are numerous tools for the execution of exploits, being the most popular, Burp Suite, OWASP ZAP, Metasploit, Nmap, Sqlmap (Bernardo Damele A. G. & Miroslav Stampar, 2021), and w3af (W. Org., 2021).

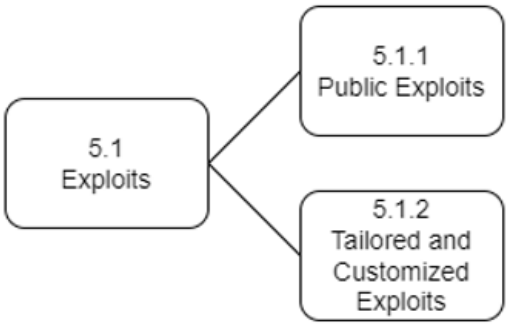

Figure 54: Process 5.1 - Sub-processes of the Exploits

3.5.1.1 Public Exploits

In order to maximize the chances of successfully exploiting a target's vulnerability, a tester will want to access as many resources as possible. With all of the information gathering done in previous steps, a tester can search for public exploits available on the internet or in other types of documentation. These exploits are developed according to specific vulnerabilities in specific hardware or software configurations, so if all requirements are met, they can be applied by the tester. In this step, a tester should take caution in terms of the liability and the source of the vulnerability since there are several fake exploits designed to harm or destroy their user's computer. Some exploit databases considered trustworthy are Exploit-DB (Security, 2021), Searchsploit (K. Org., 2021), Metasploit, and even Google (*Google*, 2021) (see figure 55).

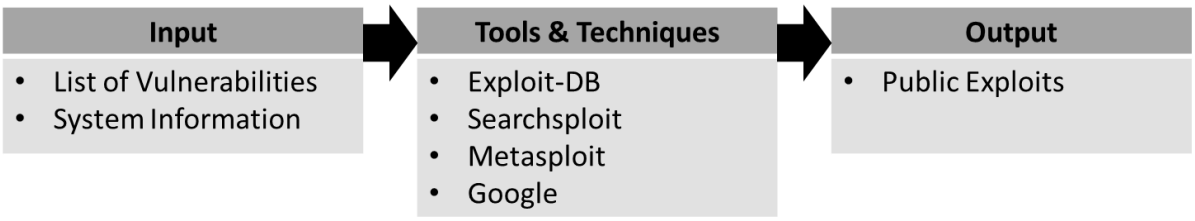

Figure 55: Process 5.1.1 – Public Exploits (ITTO)

3.5.1.2 Tailored and Customized Exploits

Since every attack will tend to be different in how the exploitation occurs, to achieve successful results, the tester may need to tailor and customize the exploitation to fit the testing scenario. By clearly understanding the testing environment and the applicability of an exploit, a tester will increase his chances of achieving a successful attack. Since public exploits may be too specific to certain versions of operating systems or applications, there is the need to change and customize exploits for their execution. This process may require an environment simulation to test the changes and guarantee its results. Even if a tester has all the information gathered about the system in his hands, having a working infrastructure and system to test the exploits will make the exploitation process easier. This is justified due to memory address changes based on service packs or new version releases. For this task, a tester will need to know how to read code developed in different programming languages and understand how the payloads work for each exploitation to adapt them according to his needs (see figure 56).

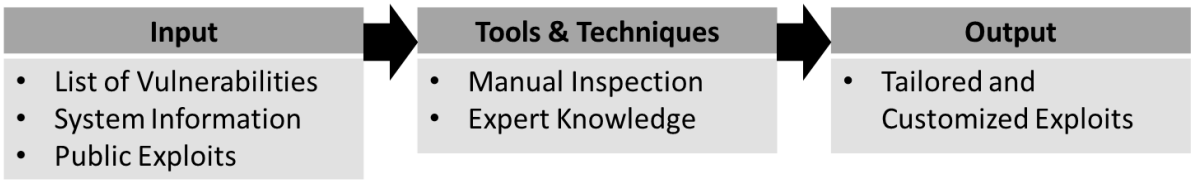

Figure 56: Process 5.1.2 – Tailored and Customized Exploits (ITTO)

3.5.2 Further Penetration

After executing successful exploits and depending on the rules of engagement and the scope of the test, a tester may further enumerate and gain access to other systems on the client's infrastructure. Using the access that was granted during the exploitation of vulnerabilities, a tester may be able to execute actions within the compromised system, allowing him to upload tools into the system, enumerate DNS of the internal network, execute brute force attacks, execute remote exploits, and abuse of compromised credentials. A tester may also use the compromised system to proxy to an internal network, configure port forwarding, access restrict information, and even manipulate authentication. This is mainly done using shell commands within the compromised system (see figure 57).

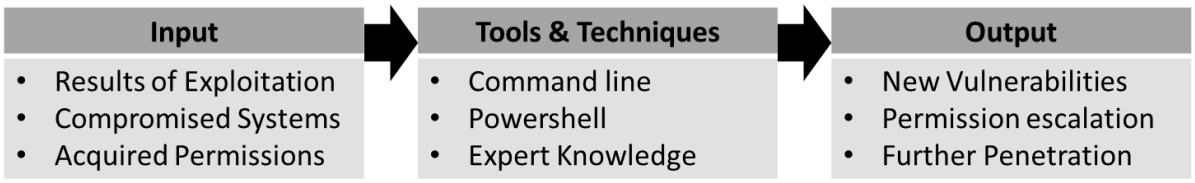

Figure 57: Process 5.2 – Further Penetration (ITTO)

3.5.2.1 New Vulnerabilities

While exploiting vulnerabilities or by further penetration on the system, a tester may find new vulnerabilities to test as permissions escalate and access is granted to different parts of the system (see figure 58). These findings will require analysis in order to evaluate them . This process will lead into a loop between the fourth and the fifth stages of this methodology because since new vulnerabilities were found, a tester must evaluate the risk and impact of the findings (as represented on figure 6).

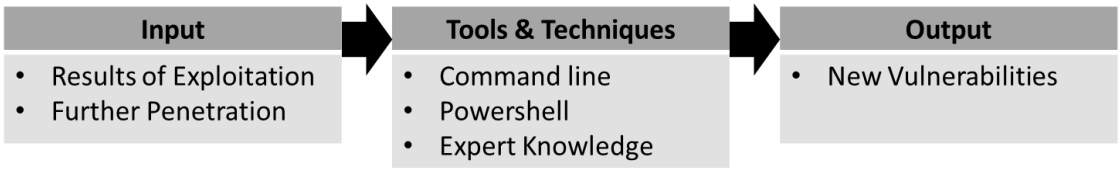

Figure 58: Process 5.2.1 – Further Penetration (ITTO)

3.5.3 Install Backdoors

The ability to modify and manipulate configurations within compromised systems allows testers to install backdoors to persist in the system (see figure 59). Installed backdoors should require authentication (in order to prevent unattended attackers) and, when possible, persist in the system after reboots. Backdoors allow testers to bypass the normal authentication process after compromising a system. This can be done to maintain or to ease future access or to exploit the system further. In order to install a backdoor, a tester can use tools designed for that purpose or use simple tools like NetCat (“Hobbit,” 2021), whose primary goal is to be used to read and write data across network connections. With this, a tester allows himself to persist his access to the system even if he gets disconnected for some reason.

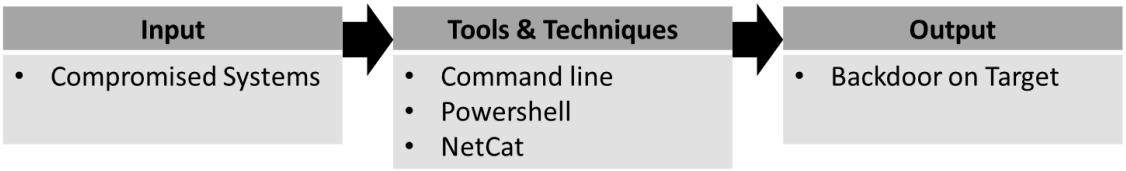

Figure 59: Process 5.3 – Install Backdoors (ITTO)

3.5.4 Clean-up

Before performing a clean-up on the system, a tester should ensure that all the exploitation steps are documented and that there are no more tests to perform on the system. The clean-up is a procedure on which a tester will clean up the system from any trace of the penetration test. This includes removing all executables, returning all the system settings and configurations to their original values, removing all of the installed backdoors, removing any user accounts created to connect to compromised systems, and restoring the database from backup to guarantee no data was damaged during the process (see figure 60).

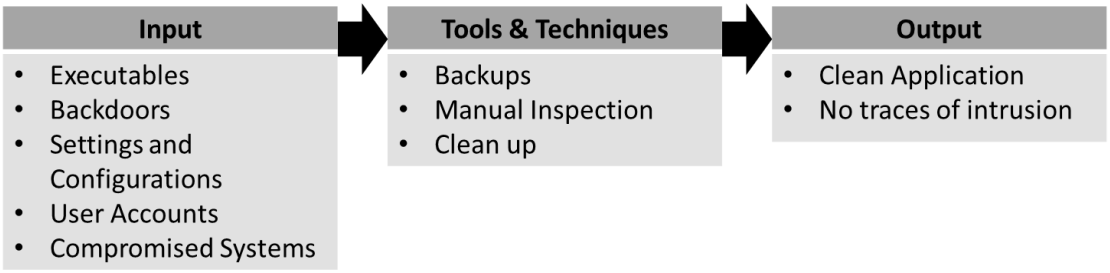

Figure 60: Process 5.4 – Clean-Up (ITTO)

3.6 Analysis of Results

The sixth stage of the methodology is the analysis of results stage. In this stage, a tester is supposed to review the entire process to validate if all the rules of engagement were followed, if the established goals were reached, if all the metrics are applicable, is it also supposed to analyze the overall impact of the penetration test on the application. This stage is divided into three main tasks, a review of established rules of engagement, goals and metrics, the analysis of the impact of the penetration test, and the analysis of the penetration test procedure as a whole (see figure 61).

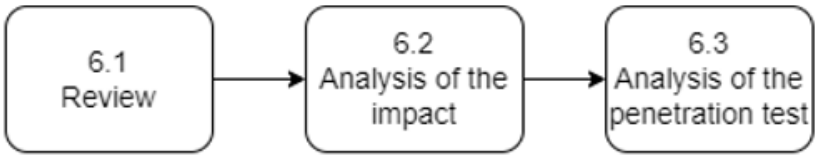

Figure 61: Process 6 - Main Procedures of the Analysis of Results Stage

3.6.1 Review

In order to evaluate compliance to previously established requirements and rules, these need to be reviewed to validate if the whole procedure went accordingly (see figure 62 and figure 63). For this review, the tester needs to check if the entire approach towards the penetration tests fits what was expected.

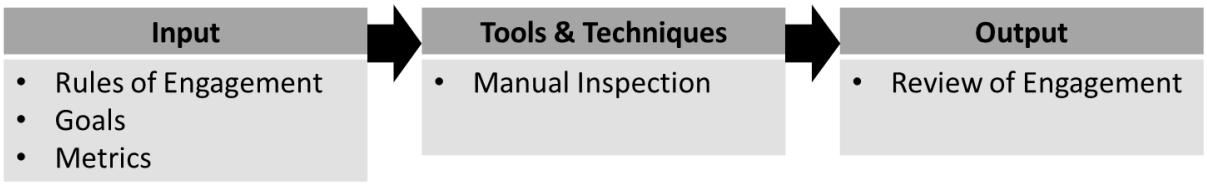

Figure 62: Process 6.1 – Review (ITTO)

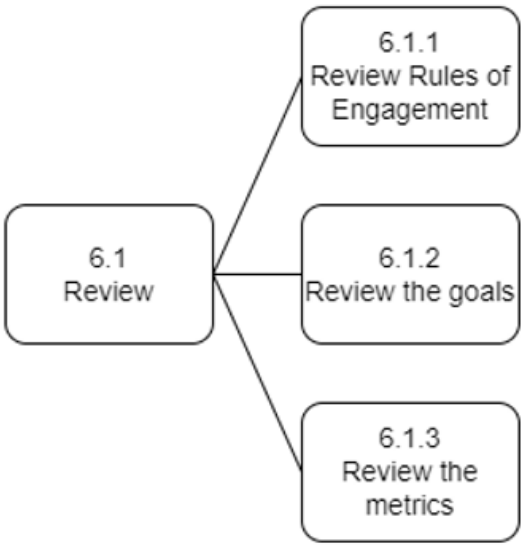

Figure 63: Process 6.1 - Sub-processes of the Review

3.6.1.1 Review Rules of Engagement

Since the rules of engagement are established to protect both parties (tester and customer), the penetration test should be evaluated accordingly to what was defined (see figure 64). A tester should assess all of the procedures taken during the penetration test to check if every action was done accordingly. After reviewing these, the tester should communicate the result of the analysis within the report on the reporting stage.

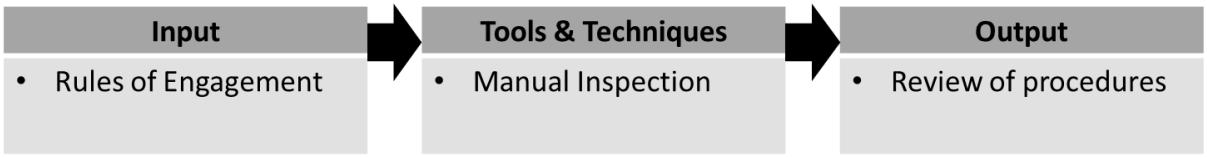

Figure 64: Process 6.1.1 – Review Rules of Engagement (ITTO)

3.6.1.2 Review the goals

The whole penetration testing process was done for a reason, and this is defined through the goals previously established (see figure 65). A tester should evaluate the penetration test results and validate if they meet the defined goals. This information must be present within the report developed in the reporting stage.

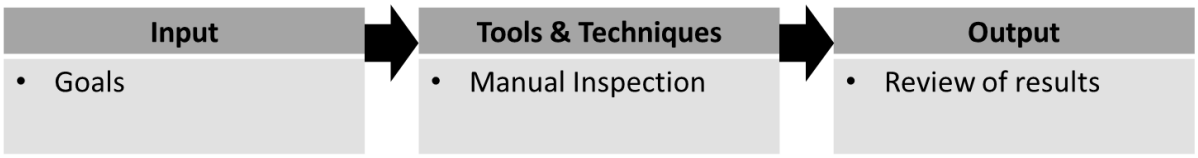

Figure 65: Process 6.1.2 – Review the goals (ITTO)

3.6.1.3 Review Metrics

The review of the metrics and their correlation with the findings is essential to measure the performance and effectiveness of the penetration test (see figure 66). Suppose the metrics are well defined and established. In that case, both parties will have a more clear and concise perspective of the results, not only for the customer to evaluate the work done by the tester but also for the tester to evaluate what went accordingly to the expectations. This information must be present within the report developed in the reporting stage.

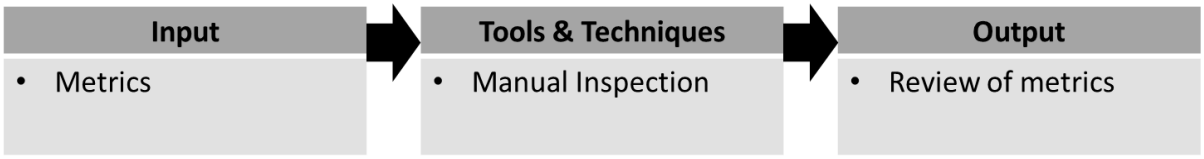

Figure 66: Process 6.1.3 – Review Metrics (ITTO)

3.6.2 Analysis of the impact

After performing all the security tests and assessments and cleaning up all of the traces of the penetration test, a tester should analyze and evaluate the impact of the whole process on the targets (see figure 67). If done correctly, there will be no impact on the systems, but the tester should do this evaluation to guarantee the system's integrity and of the used methods.

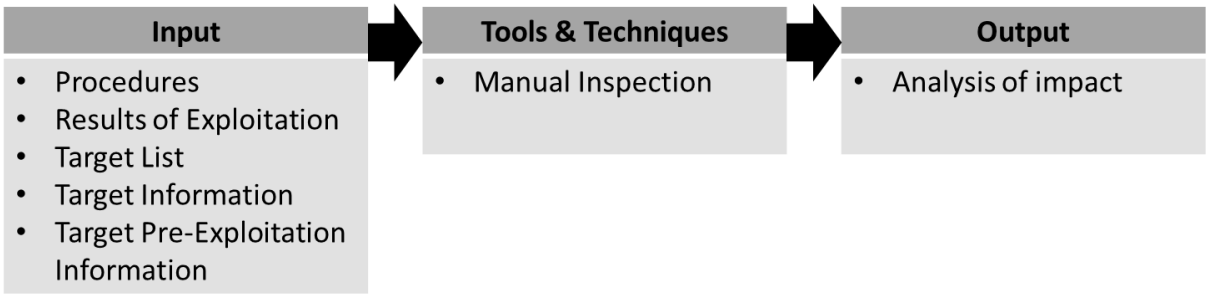

Figure 67: Process 6.2 – Analysis of the impact (ITTO)

3.6.3 Analysis of the penetration test

Considering everything, an ethical tester should evaluate and assess all of the procedures done during the penetration test (see figure 68). What went accordingly to the expectations, what failed, which tools

were best suited for specific tasks, and all additional information should be analyzed. This is mainly a procedure for the tester to evaluate his approach and skills. Self-evaluation is a crucial component for improvement and self-development. This information can be disclosed in the final report, but it is not mandatory.

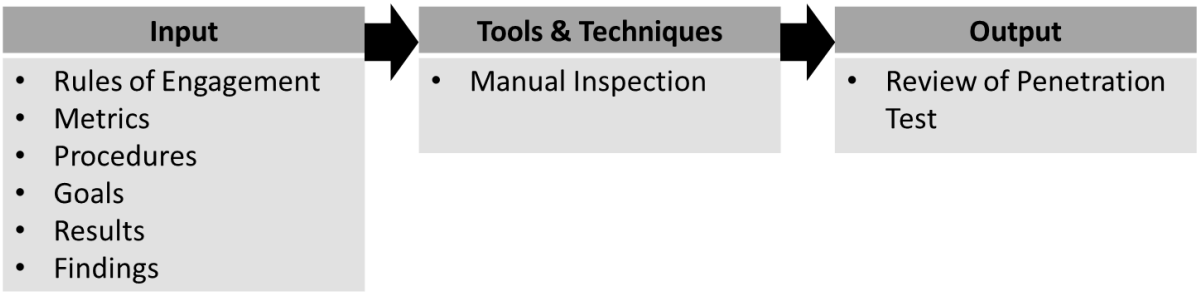

Figure 68: Process 6.3 – Analysis of the penetration test (ITTO)

3.7 Reporting

The seventh stage of the methodology is the reporting stage. The reporting stage is the final product of the penetration test. After finishing all of the previous steps and stages, a tester must elaborate a report on which he will present the customer all of the information regarding his approach, the tools and methods used, a summary of all findings and the risks associated with them, the impact of the exploitation by an outside party, prevention measures to be taken in order to fix security flaws, all of the technical information regarding the test (describing in detail the scope, the findings, the paths of attack, and the risks associated), as well as all the limitations found during the tests. The final report should be well written, informative, easy to understand and appeal to executive management and technical staff. For this methodology, a final report should be divided into three sections, the executive summary, the technical report, and an additional section for additional information (this last section is not crucial for the final product) (see figure 69).

Figure 69: Process 7 - Main Procedures of the Reporting Stage

3.7.1 Executive Summary

The executive summary sums up the overall findings of the assessment (see figure 70). This section aims to elucidate business managers and system owners on a high-level view of the vulnerabilities discovered and how to remediate those. The language of this section should not be very technical, and the tester must guarantee that even if the risks assumed in the report are related to the system, the tester

would not know the risk for the organization in case of exploitation, as this is the job for a risk manager to identify and calculate. In this chapter, the tester should also include information regarding the project objectives, the scope, the timeframe for the tests, the list of targets, the limitations found, a summary of the findings, and the remediation measures (see figure 71).

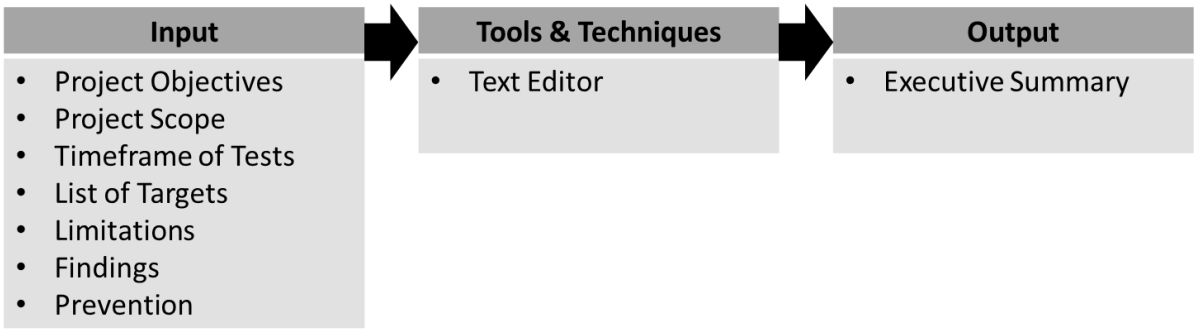

Figure 70: Process 7.1 – Executive Summary (Input and Output)

Figure 71: Process 7.1 - Sub-processes of the Executive Summary

3.7.1.1 Project Objectives

In this subsection of the report, the tester must outline all of the objectives and goals of the penetration test, as well as the metrics used and their values accordingly to the findings and the expected outcome of the tests (see figure 72).

Figure 72: Process 7.1.1 – Project Objectives (ITTO)

3.7.1.2 Project Scope

In this subsection of the report, the tester must outline the agreed scope of the tests and how it was respected during the testing process (see figure 73).

Figure 73: Process 7.1.2 – Project Scope (ITTO)

3.7.1.3 Timeframe of Tests

In this subsection of the report, the tester must outline the timeframe of the tests and the dates of the commence and conclusion of the tests (see figure 74).

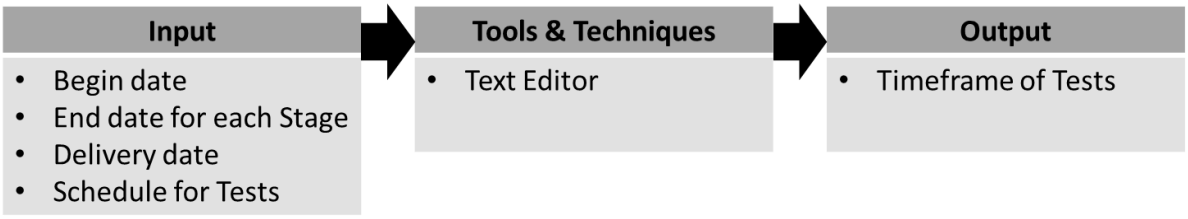

Figure 74: Process 7.1.3 – Timeframe of Tests (ITTO)

3.7.1.4 List of Targets

In this subsection of the report, the tester must list the targets of the tests and any additional information that may be relevant for the topic (see figure 75).

Figure 75: Process 7.1.4 – List of Targets (ITTO)

3.7.1.5 Limitations

In this subsection of the report, the tester must outline every limitation found during the tests. For example, limitations in terms of methods, technical issues, performance, limitation of tools, and any additional information regarding this topic (see figure 76).

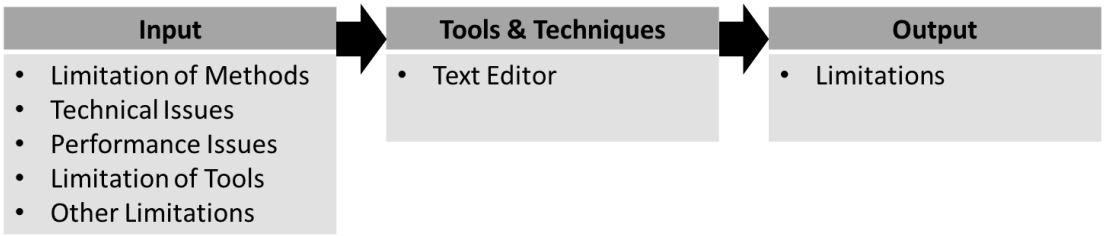

Figure 76: Process 7.1.5 – Limitations (ITTO)

3.7.1.6 Findings

In this subsection of the report, the tester must outline every vulnerability found during the tests with information regarding the risks for the system (see figure 77).

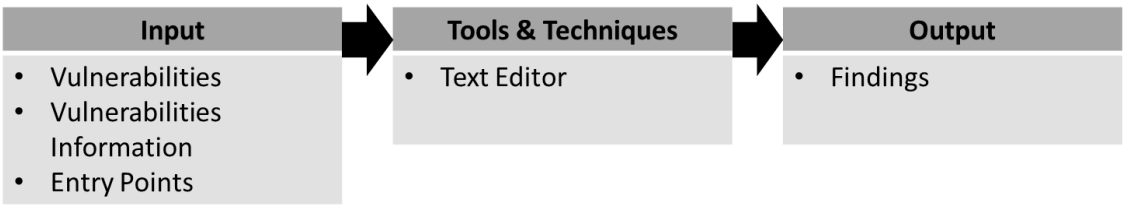

Figure 77: Process 7.1.6 – Findings (ITTO)

3.7.1.7 Prevention

In this subsection of the report, the tester must outline the action plan for remediation and prevention of the vulnerabilities found during the tests (see figure 78).

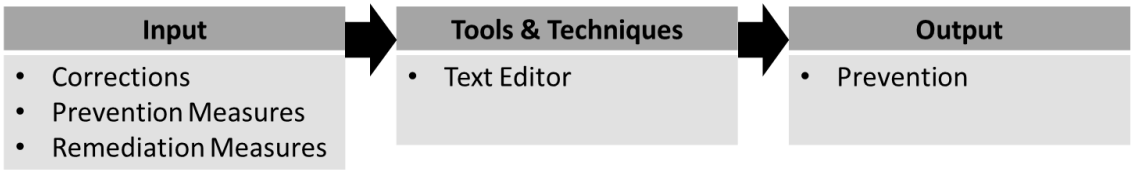

Figure 78: Process 7.1.7 – Prevention (ITTO)

3.7.2 Technical Report

This section of the report should be clear, concise, and more technical than the executive summary and include all the necessary information for the technical teams to understand the security flaws and the prevention measures, and the severity rank of each vulnerability found (see figure 79 and figure 80). In order to give the readers a better understanding, screenshots and command lines may be included to show the steps taken during the exploitation.

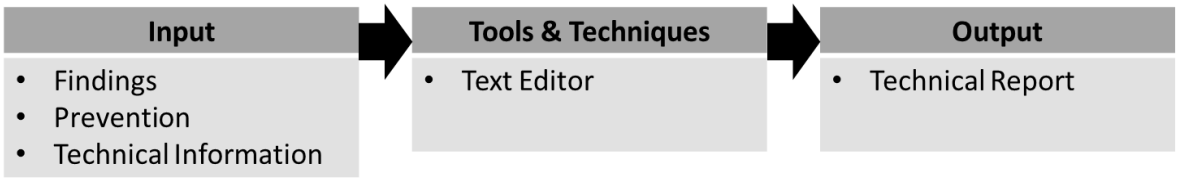

Figure 79: Process 7.2 – Technical Report (ITTO)

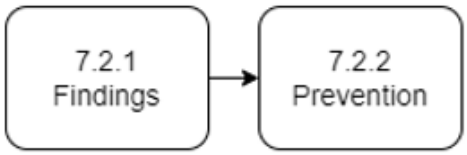

Figure 80: Process 7.2 - Sub-processes of the Technical Report

3.7.2.1 Findings

In this subsection, a tester must include detailed information regarding his findings, including all necessary information for the technical team to understand and replicate (if needed) the scenario of the tests (see figure 81). It should also include a severity ranking for all of the vulnerabilities and a technical description of the issue and the affected objects. Screenshots and command lines may be presented in order to give a better understanding of the procedures.

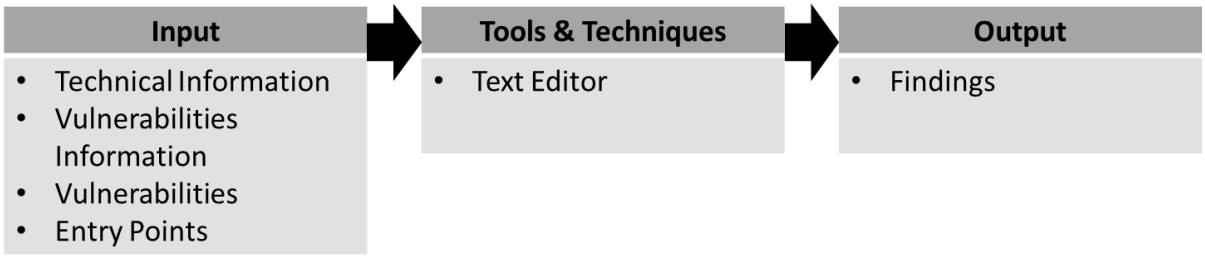

Figure 81: Process 7.2.1 – Findings (ITTO)

3.7.2.2 Prevention

In this subsection, a tester must include detailed information regarding the prevention measures to be taken into account, including all the technical information for the technical team to resolve the issues found (see figure 82).

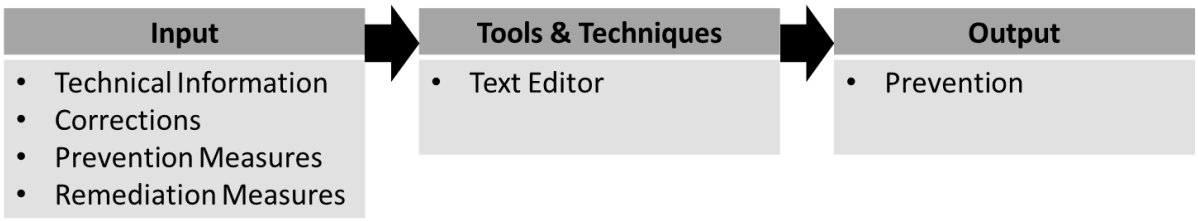

Figure 82: Process 7.2.2 – Prevention (ITTO)

3.7.3 Additional Information

This final section of the reporting is not mandatory. However, a tester may want to add additional information regarding the tests, which could be a self-evaluation of the process or any information regarding previously established points of relevance (see figure 83).

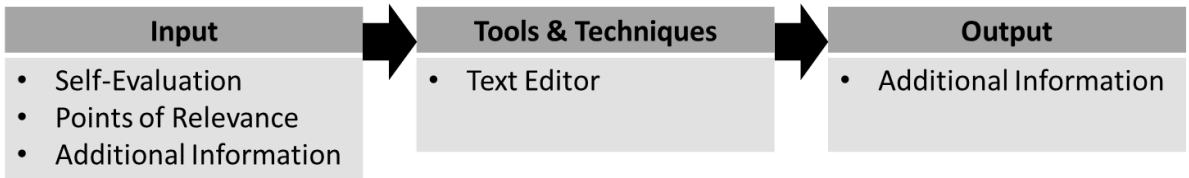

Figure 83: Process 7.3 – Additional Information (ITTO)

3.8 Conclusion

This methodology was defined and specified, taking into consideration some of the most industry-used web application penetration testing methodologies. This methodology can be considered as a work in progress since it requires further validation in a more significant number of test cases to be considered as an applicable methodology, but it can still be developed with that purpose. It was inspired by the research made during state of the art, taking into consideration the base standards and methods used by professional testers and combining them with some of the most common tools that most testers use. Its development aimed to assist any tester to follow a generic guide in order to assess any web application through any method of penetration testing (black-box, grey-box, white-box). The recommended tools are mostly Open-Source to be as generic and inclusive as possible, but some commercial tools were still referred and are recommended. There may be more tools to choose from, but the ones referred on this methodology go accordingly to the industry standards.

Chapter 4 – Demonstration: Penetration Testing Methodology

For the development of this dissertation, it was decided to conduct a pentest to the web application Accipiens. This choice was made due to the access granted as a technical consultant working for VTXRM and the need to develop a methodology to assess Accipiens security with consistent results. Accipiens was the chosen test case for this dissertation in order to prove this methodology appliance.

This chapter will present more information on the environment setup, a small presentation of Accipiens and its infrastructure, outline the most appropriate tools for this process, analyze different testing methodologies, and demonstrate the appliance of the developed methodology.

4.1 Environment

The environment setup was composed of a Virtual Machine with Kali Linux OS installed for the Design and Development stage. During the Pre-Engagement phase, it was defined that the scope of the Pentest should only be a specific port of installation of the software with a specific address for connection, which required access to a Virtual Private Network (VPN). Permissions for the use of the application were granted but with limited access, which restrained direct connection to the database through the given credentials. All-access was granted within the WebApp but not to all directories included in terms of application permissions.

While working for VTXRM, some knowledge was acquired about Accipiens and some of the frameworks and infrastructures used to host the application. However, there were some restrictions regarding access to core source code and the architecture of the network and application. Considering this, the methodology applied to the pentest was planned according to a grey box approach.

It was confirmed that the application was hosted on Microsoft Internet Information Services (Microsoft IIS). Most of the scripts were developed based on .Net Architecture Guides. The database is hosted in Microsoft SQL Server. The back-end scripts were primarily developed in C# programming language, and the web services were developed in Soap, REST JavaScript, and Angular.

4.2 Accipiens

The Accipiens is a financial software developed by VTXRM – Software Factory that achieves to turn their customers into market leaders through an Enterprise Resource Planning (ERP) solution in the market. It is built under a Service Oriented Architecture to support Business Process Management and

developed to be a Web Application used wherever it is needed. It was developed to create software capable of achieving high scalability associated with a flexible and modular architecture, acting as a contract management system that helps standardize processes globally, increase productivity, and reduce costs. It is developed in various languages and can cover the whole business life cycle, from Leasing to Credit and Factoring businesses.

Figure 84: Accipiens Worldwide Footprint

Accipiens is used in more than ten countries over three continents (see figure 84). It offers software that benefits its clients with fully automated processes, tailored workflows, a configurable solution, industry standards implementation, and powerful credit analysis features (see figure 85).

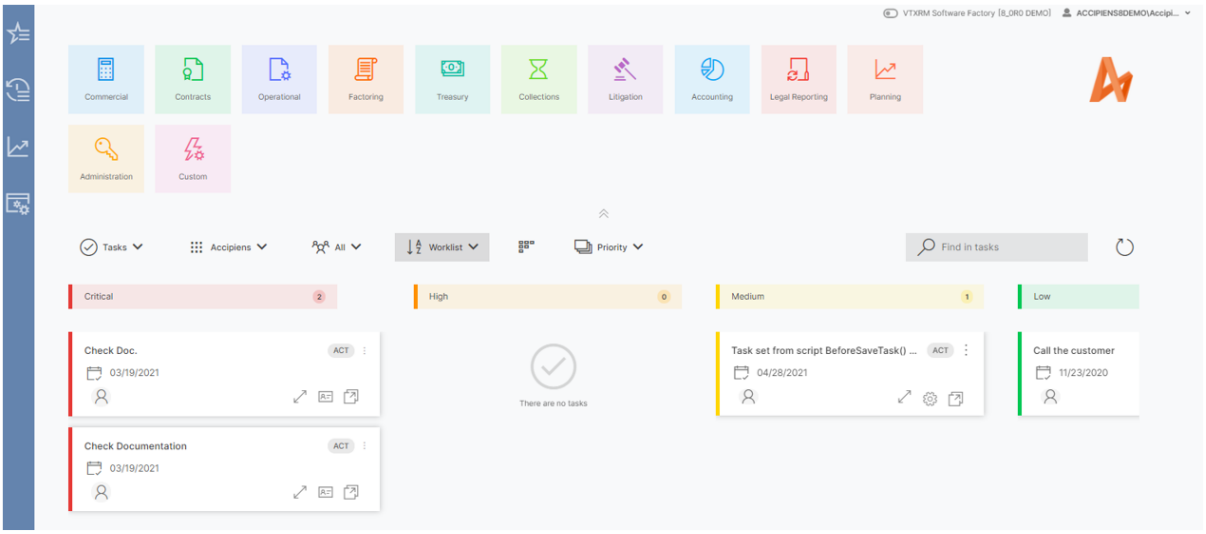

Figure 85: Accipiens Homepage

The application is divided into three tiers, the data tier, the logic tier, and the web tier (see figure 86). The data tier is where the database is located, organized by data sources. The stored data is distributed through ADO.NET (a set of classes used to access data sources and is developed under the .Net Framework) to the logic tier. The logic tier is divided into two layers, the data layer that accesses data and manipulates it and the business layer where the data is processed and through the implemented web

services is transported into the web tier. The web tier is composed of a presentation layer where the UI (user interface) is responsible for presenting the information required.

Figure 86: Accipiens three-tier architecture

The authentication of users is done by HTTP, and most of the application is developed under the .Net framework with most scripts developed in C#, a Microsoft SQL Server database, most of the web services are developed in JavaScript, Angular, SOAP and REST and is hosted on Microsoft Internet Information Services. The version control of the application development was made using Microsoft Team Foundation Server (TFS) and, more recently, upgraded into Azure DevOps. The security of the application is defined through user profiles with specific permissions per user/role.

A standard technical consultant at VTXRM has access to all modules of the application and most of the directories within (see figure 87), but the permissions for each port of installation are given, taking into account the projects that the consultant is allocated.

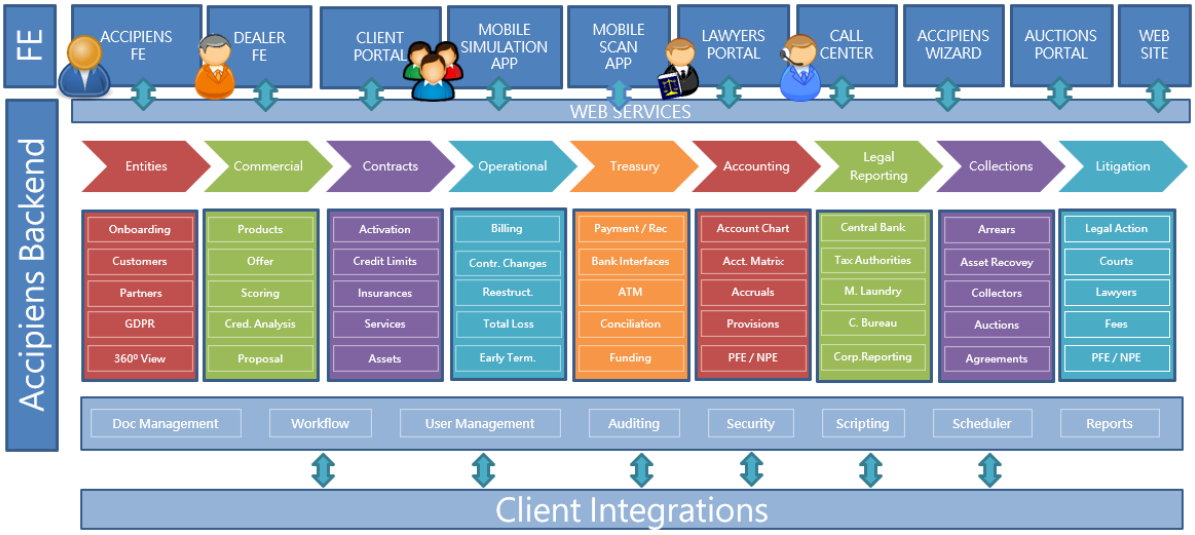

Figure 87: Accipiens modules diagram

4.3 Tools

The first part of the Pentest was to understand which were the best tools to perform a vulnerability and security assessment within the available Open Source scanners and including some commercial scanners. The selected tools were chosen based on previous research made during the development of section 2.7, by taking into account the review made by Shay Chen on SecTool (Chen, 2016), which evaluates (commercial and open-source) vulnerability scanners based on the features of each one, having in mind that the authentication for Accipiens is made through NTMLv2 (an authentication protocol used by Windows to authenticate network users) using the VPN connection and by consulting professional security testers on the most used scanners in the industry.

The initial tools selected were Nmap, Metasploit, Burp Suite, OWASP ZAP, Nessus, w3af, and Acunetix. Later, this list was narrowed down since the w3af tool was removed from Kali Linux, and its installation was not possible due to Module problems and the lack of ConfigParser. Acunetix was removed from the list since its usage was restricted, and no access was granted for academic purposes. A free license was asked for to PortSwigger for Burp Suite Pro, but it was not granted, resulting in the usage of the free version, Burp Suite Community Edition.

4.3.1 Nmap

As stated previously, Nmap is an open-source tool for port scanning and OS fingerprinting; its everyday use is network discovery and security auditing. Nmap uses raw IP packets to determine available hosts, services, operating systems, possible firewalls or packet filters, plus many other characteristics. It was chosen due to its popularity among security testers and its variety of utilities, which make it possible to

scan many different components and elements belonging to a particular network, giving fundamental output during a network scan.

As described by Nmap Organization, it is flexible by including many port scanning mechanisms, powerful since it has been used to scan massive networks with thousands of machines, portable due to the compatibility with all major OS's, easy to use, free by letting any user download it without a cost, well documented and translated in multiple languages, supported by a big community of developers and users, acclaimed with many awards and popular.

4.3.2 Metasploit

Metasploit is a very versatile framework, easy to customize, and compatible with most operating systems. Built into Kali, this framework made it easier for pentesters to remote test the security of systems. This tool allows users to develop and execute exploits via command line or through GUI, with many commercial-grade exploits and an extensive exploit development environment. This framework is divided into modules that contain a lot of scripts, tools, and plugins with a library that allows exploiting different vulnerabilities without having to write additional code by only defining an exploit and its payload.

4.3.3 Burp Suite Community Edition

Burp Suite is a platform used for the security testing of web applications. It is a tool capable of mapping and analysis of an application environment. The Community Edition features some essential tools, such as a Repeater, a Decoder, a Sequencer, a Comparer, Burp Intruder, HTTP(s) and WebSocket's proxy. With many plugins and functionalities available, this tool is one of the most recommended by professional testers.

Since the use of Burp Suite was restrained to the Community Edition, the features available are as follows:

Repeater – A simple tool for manually manipulating and reissuing individual HTTP and WebSocket messages in order to analyze the application's responses.

Decoder – This tool is mainly used to transform encoded or raw data into its canonical, encoded, or hashed form by recognizing different encoding formats.

Sequencer – This tool allows testers to validate unpredictable information such as session tokens, anti-CSRF (Cross-Site Request Forgery) tokens, password reset tokens, and more.

Comparer – The Compare is a simple tool used to compare any two items of data.

Proxy – With Burp Suite, it is possible to operate a web proxy server between the browser and the targeted application, which can intercept, inspect and modify raw traffic passing in both directions.

Intruder – The Intruder is a tool for automated customized attacks. It is highly configurable to perform various tasks, from brute-force guessing of web directories to active exploitation of complex vulnerabilities.

4.3.4 OWASP ZAP

The OWASP ZAP is an open-source web application security scanner maintained by a team of international volunteers, being one of the world's most used web app scanners. Used as a proxy server, it allows users to manipulate the total traffic that passes through it, making it possible to map web application directories and resources through the ZAP spider. This tool is easy to set up and use, finding and detecting security vulnerabilities related to SQL injection, Broken Authentication, Sensitive data exposure, Broken Access control, Security misconfiguration, Cross-Site Scripting (XSS), Insecure Deserialization, Components with known vulnerabilities, and Missing security headers. Through its compatibility, it is possible to setup in all major OS's.

4.3.5 Nessus

Nessus is an Open-source remote security scanning tool mainly used for vulnerability assessments and penetration testing engagements. Testing each port of a computer can determine what service it is running and then test it for any known vulnerability to prevent attackers from carrying out malicious attacks. It can search for vulnerabilities related to authentication and access to sensitive data, misconfigurations, denial of service vulnerabilities, software flaws, malware, and missing patches. It uses a server-client architecture, and it allows for the installation of several plugins to handle a wide variety of vulnerability checks.

4.4 Methodology used

The designed methodology used for this dissertation is divided into seven stages, similar to the methodology developed by PTES. The whole testing process starts with a scoping meeting on which an NDA (non-disclosure agreement) between the tester and client is signed to guarantee the client that no confidential or sensitive data will be disclosed or shared in any form. After the agreement, the scope of the test should be defined, in terms of infrastructure limitations, requirements, metrics to evaluate the security and the targets of the test (database, application, network, social engineering, and others). Additional information about the target should also be disclosed accordingly to the scope and the expected approach (if a tester is expected to have a black-box approach, no information should be disclosed, if it is a white-box approach, the tester should have access to all the information on the system as well as its source code and if it is a grey-box approach, only relevant information should be disclosed), as well as the technical details of the infrastructure and other relevant processes. With this information, a tester is supposed to define a time estimation for the test and decide the tools to use and

the methodology he is going to apply. The next step of this methodology is the Reconnaissance stage, in which the tester should investigate more information about the application, the network, and the systems where it is hosted. For this, through tools like Nmap, Burp Suite, or Metasploit, it is possible to map the network, the open host ports, and its running services, to map some of the application directories, and to discover information about the database used, the system where it is running on, disclosed addresses, versions, instance, and server name and the TCP port where it is running. After gathering basic information about the target, the tester should start the Scanning stage by taking a more active approach towards the application. Utilizing automatic scanners makes it less time consumable, but some manual inspections should also be done. For the active scan, a tester can use Nessus to test the application security and scan for active vulnerabilities; OWASP ZAP can also be done with the same goal and use the built-in crawler to map the application deeply, discover new directories or vulnerabilities. Burp Suite is an excellent tool for manually inspecting intercepted messages between the browser and the application to review and analyze different interactions. The Vulnerability Analysis stage starts with understanding the impacts of the previously found vulnerabilities, assessing their entry points, and classifying them according to their severity. After analyzing all the information on the vulnerabilities found, the Exploitation starts on which a tester is going to try to exploit the entry points to confirm or deny the presence of a vulnerability on the target, as well as test possible entry points or exploits that were not found during the scanning stage but that the application could be vulnerable. During this process, a tester could find new vulnerabilities. In that case, those should be analyzed again, creating a loop between the Vulnerability Analysis and the Exploit stages until no more new vulnerabilities are found. Suppose there are no more vulnerabilities to test. In that case, the tester should proceed to the Analysis of the Results stage in order to evaluate and assess the results of the exploitation, concluding on the possible impact that the exploitation may have on the system, the risks associated, how to prevent it and the overall result of the pentest in terms of the metrics defined. The last stage of the methodology is the Reporting stage on which a tester is intended to fill and deliver to the customer a report with detailed information about the scope, the vulnerabilities found, the steps to find it and to exploit it, the impact it may have, remediation for the vulnerability and any additional technical detail considered relevant.

4.5 Test Case - Accipiens

This chapter will present the test case for the application of the developed methodology, using specific methods and tools to assess the security of Accipiens. Every phase of the methodology will be explained in terms of the procedures.

4.5.1 Planning

Table 5 – Planning Stage Process

Input	Tools	Output
NDA	Microsoft Word	Basic Application Information
System Information	Virtual Machine	Set of Tools
Technical Details	Adobe Reader	Chosen Methodology
Scope of the test		

The main goal of the pentest is to focus the effort on process review and manual inspection of the application and its functionalities while also making code reviews and security testing. The first step was to plan the pentest. Together with VTXRM, it was decided to take a grey box approach. The system behind Accipiens was partially explained, access was granted to a specific port of installation, there was no NDA signed since the employment contract already has a non-disclosure clause in order to restrain sensitive information from being leaked (specific directories, the information on some ports, and the localization of some vulnerabilities), and the tools to use were selected (Nmap, Metasploit, Burp Suite Community Edition, OWASP ZAP, and Nessus). Knowing the system infrastructure and some of the protocols used may be a good starting point to list known vulnerabilities of its components. However, due to the previous pentest made on Accipiens, they were all fixed in terms of application configurations and server flaws.

4.5.2 Reconnaissance

This stage intends to do recon into the application to find more about its structure, components, and application insight. With this, the second step of the pentest starts with scanning the application and the network that supports it.

Table 6 – Reconnaissance Stage Process

Input	Tools	Output
Network Address	Nmap	Network Map
Web Application	Burp Suite	Application Map
MS SQL	Metasploit	Database Addresses

Using Burp Suite, it was possible to map most directories used by the Accipiens through manual inspection of the application and finding other ports related to the Web Application (which ended up

being a support port with little to no importance for this test). After mapping the application, it was time to map the network using Nmap. Initially, the command used for Nmap was:

nmap -sC -sV -oN nmap/initial 'web application address'

-sC = Script Scan

-sV = Service Version Detection

-oN = output scan in normal format

Nmap then reported all available ports used by the application, what type of ports are used and their states, services running, authentication and the type of authentication present, and the host scripts.

After mapping the application and the network, it was time to search for any database connection, not to exploit it but to have basic knowledge of the structure and all the nodes that belong to it. In order to find all database addresses, one of the auxiliary modules of Metasploit was used, which in this case was MSSQL_Ping. By setting up credentials (Username and Password) and a target address or range of hosts, this module queries on port 1434 to determine the listening TCP ports of any MSSQL server. With this, the basic reconnaissance of the application was completed, and most of the application scope was mapped and ready to be scanned and analyzed.

4.5.3 Scanning

This stage aims to find new vulnerabilities and new entry points through a more active scan than the previous stage. Using the right tools and proper configuration is possible to find vulnerabilities in the system. For this pentest, automatic scanners were mainly used while also performing some manual inspection of code on HTTP requests.

Table 7 – Scanning Stage Process

Input	Tools	Output
Accipiens UI	Nessus	Potencial Vulnerabilities
Web Application Address	Burp Suite	Application Dictionary / Wordlist
Accipiens Directories	OWASP ZAP	
	Manual Inspection	

The first step taken towards the scanning phase was to run Nessus and configure a Scan with Web Application Tests as its policy. During the scan configuration, the credentials and the type of authentication were parametrized; it was also defined that the scan should go through every reachable port by pinging hosts that use TCP, ARP, ICMP, or SYN. The scan was also configured to search for

all vulnerabilities while using low bandwidth links to slow down the scan when network congestion is detected.

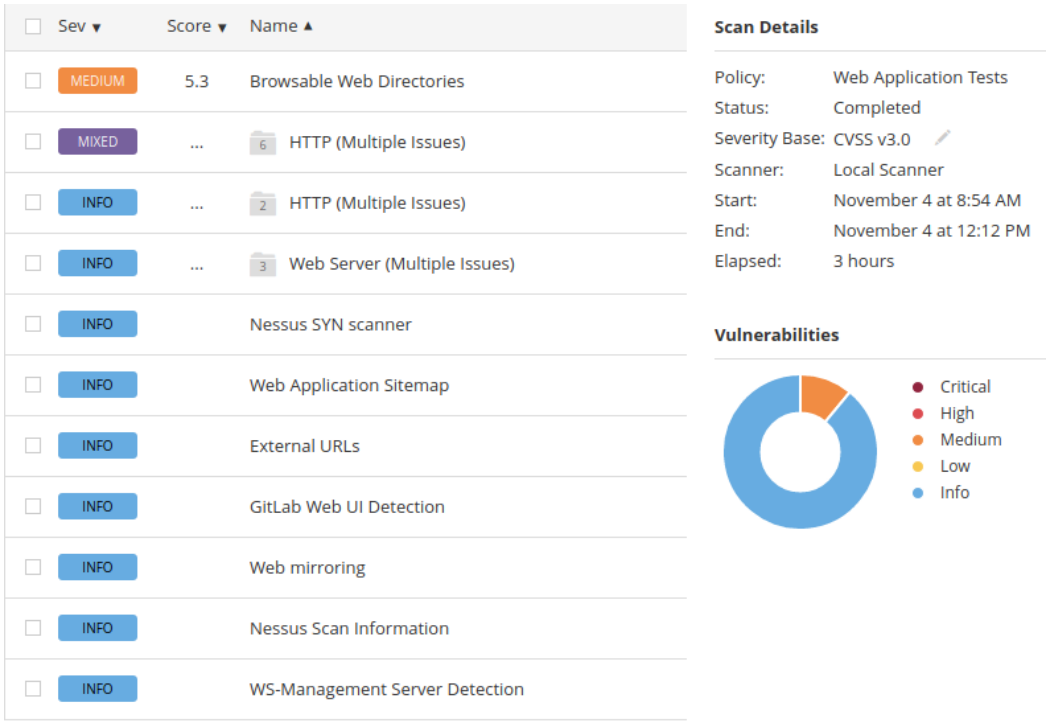

Figure 88: Overall Scan result from Nessus

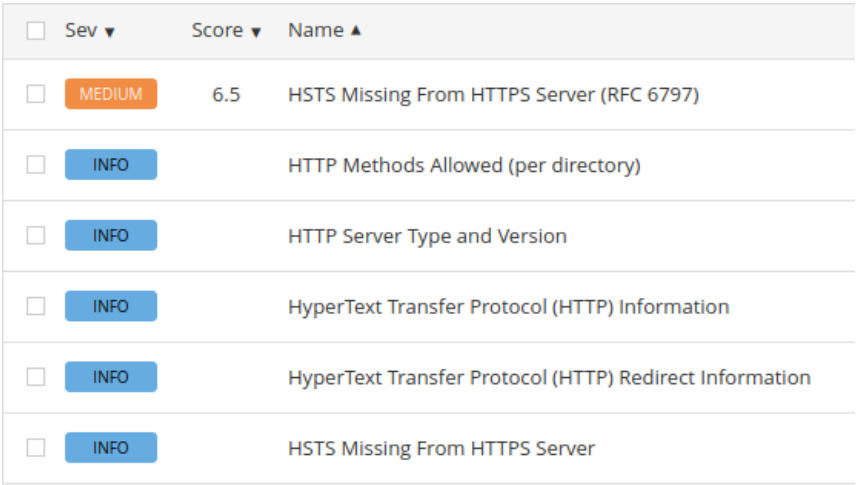

Figure 89: Package HTTP (Multiple Issues) from Nessus scan

As it is observable on figure 88 and figure 89, the results show a maximum severity of the vulnerabilities to be Medium Level, which in this case were Browsable Web Directories and in the package HTTP (Multiple Issues), the only vulnerability with relevant severity was HSTS Missing from HTTPS Server (RFC 6796). Everything else detected by Nessus was merely informative, which does not directly translate into a vulnerability but, it may be important info to exploit flaws in the application configuration.

After running the Nessus scan and giving the results, it was time to use Burp Suite to analyze the requests made between the server and the setup proxy. It was intended to search more specifically into

authentication messages, executions, and calls to the database. In this step, the Burp Suite tools used were the Proxy, the Decoder (used to decode hashes), the Repeater (in order to manipulate requests to try to find some vulnerabilities within the responses of the server), and lastly, the Sequencer (which was used to validate anti-CSRF tokens). This process involved much manual inspection of the application and the messages traded between both systems, but besides new browsable directories that were found in this process, no significant results were found, and since the most sensitive files and directories found while browsing were restricted (no permissions to access) or not found (error 404 – File or directory not found) when accessed, it ended with no added value. Since Burp Suite saves all of the application directories, it was possible to create a wordlist of words used in the context of the application, which could later be used for brute force exploitation or directory guessing. While running the manual inspection of Accipiens, an application error appeared. When Accipiens has an internal error (during the execution of any script that has an error or handling unexpected data), a stack with the error pops up with information from both Frontend and Backend. By analyzing the stack, it was possible to find some sensitive information, which was about the error and information about internal scripts, directories, and files, as well as some database information, which could be used to facilitate exploitation.

OWASP ZAP was then used to scan the application using HTTP authentication with the given credentials, but it was not able to find any vulnerability with the automatic scanner since it was not able to login into the application. For this reason, a manual scan was done, and the following vulnerabilities were flagged (see figure 90).

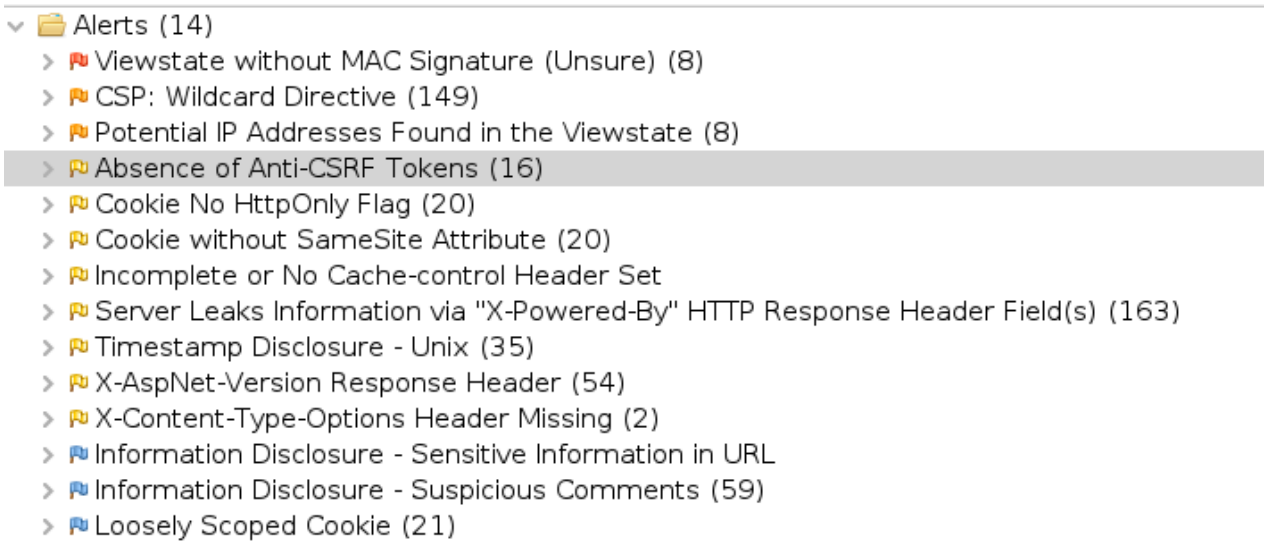

Figure 90: OWASP ZAP Scan results

With OWASP ZAP Spider, it was also possible to map all directories, scripts, and files previously discovered, but it also checked for input forms and payloads unknown until this moment. Given the previous results, it is noticeable that much more vulnerabilities were found and, in this case, with higher severity than with the previous software's.

Table 8 – Vulnerabilities found

Vulnerability	Severity	Scanner used
Viewstate without MAC Signature	High	OWASP ZAP
Browsable Web Directories	Medium	Nessus
HSTS Missing from HTTPS Server (RFC 6796)	Medium	Nessus
CSP: Wildcard Directive	Medium	OWASP ZAP
Potential IP Addresses Found in the Viewstate	Medium	OWASP ZAP
Absence of Anti-CSRF Tokens	Low	OWASP ZAP
Cookie no HttpOnlyFlag	Low	OWASP ZAP
Cookie without SameSite Attribute	Low	OWASP ZAP
Incomplete or No Cache-control Header Set	Low	OWASP ZAP
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	Low	OWASP ZAP
Timestamp Disclosure – Unix	Low	OWASP ZAP
X-AspNet-Version Response Header	Low	OWASP ZAP
X-Content-Type-Options Header Missing	Low	OWASP ZAP

4.5.4 Vulnerability Analysis

The next step for the pentest was to analyze the vulnerabilities found until this point. In order to do this, it is mandatory to check the vulnerabilities found, understand how they work and their impact in case of exploitation.

Table 9 – Vulnerability Analysis Process

Input	**Tools**	**Output**
List Vulnerabilities	Research	Severity
System Information	Acquired Knowledge	Impact
		Entry Points
		Attack Vectors

From the scanning phase, we ended up with thirteen different vulnerabilities, of which one of them had a high severity, four with medium severity, and seven with low severity. The severity of a vulnerability is defined by the Common Vulnerability Scoring System (CVSS); it evaluates the severity and not the risk directly associated with the vulnerability by representing only the intrinsic characteristics of a vulnerability, which are constant over time and across user environments. The severity is a measure used for assessing and communicating the characteristics and impacts of security vulnerabilities.

The vulnerabilities found with the most considerable severity are as follows:

- Viewstate without MAC Signature – The ViewState is a client-side state management technique that allows storing user data on the page during the post back. By default, the server signed the serialized value to prevent tampering by the user, but it can be disabled by setting the Page.EnableViewStateMac property to false, allowing an attacker to modify the contents of the ViewState and cause arbitrary data to be deserialized and processed by the server. To prevent it, the Page.EnableViewStateMac property must be defined as TRUE (PortSwigger, 2021a).
- Browsable Web Directories – This issue means that a web directory was found and is browsable. By using access restriction or disabling directory indexing, this can be prevented. Another prevention option is to ensure that the directories do not leak any confidential information or give access to sensitive data (Rapid7, 2021).
- HSTS Missing from HTTPS Server (RFC 6796) – The remote web server is not forcing HSTS as defined by RFC 6797. HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The remote web server can be configured to use HSTS in order to prevent it (Tenable, 2021a).
- CSP: Wildcard Directive – CSP adds a layer of security that helps detect and mitigate certain types of attacks such as XSS or data injections. Poorly configured CSP allows attackers to execute malicious code on the browser. This can be prevented by configuring the webserver to set the CSP header(Scan Repeat, “CSP Scanner: Wildcard Directive”).
- Potential IP Addresses Found in the Viewstate – The ViewState is turned on by default on ASP.NET, serializing the data in every control on the webpage. If the ViewState is turned on, it allows a user to encrypt or decrypt the content of hidden fields. If an IP address is found, it can be easily read and used to access servers and services that may be less secure than other entry points. This can be prevented by ensuring that the page uses HTTPS, checking if ViewState contains any sensitive information, and encrypting the ViewState to hide it from all users (Repeat, 2021).

Not all the vulnerabilities found had a direct impact on the application, even though some could cause arbitrary data to be deserialized and processed by the server or allow SSL-stripping man-in-the-middle attacks and weaken cookie-hijacking protection.

4.5.5 Exploitation

In this stage, the main goal is to confirm if the vulnerabilities found are real vulnerabilities or just false-positive artifacts found during the previous stages; it is also intended to explore new entry points or exploitations whose vulnerabilities were not flagged but still may be present.

Table 10 – Exploitation Stage Process

Input	Tools	Output
Vulnerabilities	Metasploit	New Vulnerabilities
Entry Points	Burp Suite	Confirmation of Vulnerabilities
Exploits	OWASP ZAP	
Attack Vectors	Manual Testing	

All the above vulnerabilities are stated as positive, but not exploited during the test, since they existed due to configurations made on the network and the application itself intentionally. Nonetheless, other vulnerabilities were also tested to assure the maximum efficiency of the pentest. The configuration setting for each vulnerability was checked in order to confirm the findings. For this, vulnerabilities such as SQL injection, Cross-site scripting, HTTP request smuggling, XML injection, and brute-force attacks were also tested. After all the tests, SQL injection was well handled by Accipiens, not revealing to be a flaw in the application since all information is dealt through the backend, which tests for injections in all the forms used, the CSS, and the XML injection were both handled by the Request.Form validation defined on the page directive. The brute-force attacks for directories and authentication made of diverse wordlists, not only from commonly known wordlists, as with the wordlist created from the directories mapped by Burp Suite and OWASP ZAP, but since the authentication is handled by HTTP, which timeout after some tries and no new directories were found it is plausible to assume that it is not a vulnerability of the application. As for the HTTP request smuggling, it was impossible to test if it was a real vulnerability since it was impossible to bypass the requests' authentication. Some known vulnerabilities of the system were also tested in terms of the frameworks used and system specification and from reports of previous pentests made to the application.

Figure 91: Result of CSS on Accipiens

Since the access was already granted and none of the vulnerabilities found were exploited, there was no privilege escalation and effort involved to maintain access.

4.5.6 Analysis of the Results

Despite the results, and even though not many vulnerabilities were found, it does not imply that the system is unbreakable and secure.

Table 11 – Analysis of the Results Stage Process

Input	Tools	Output
Results of Exploitation	Manual Inspection	Impact on the System
Established Goals and Scope		Result of Pentest

Some of the vulnerabilities found are still something to worry about, for instance, the information contained in the errors stack, which exposes internal information about the server configuration and paths to any user that runs into an error (see figure 91). Most of the vulnerabilities found are positive, but they can all be avoided by changing the parametrization of the application and its configurations.

4.5.7 Reporting

The communication of the technical details of the test and all of the aspects planned on the first stage of the pentest is done in the Reporting stage. At this point, the tester should elaborate a document containing a description of the scope, information disclosed during the process, the steps taken by the

tester to exploit any vulnerability, the impact of the performed exploitation, and solutions to solve the found vulnerabilities.

Table 12 – Reporting Stage Process

Input	Tools	Output
Executive Summary	Manual Inspection	Final Report
Technical Report	Microsoft Word	
Additional Information		

In this case, the report of the whole process is mainly done through this dissertation, except for sensitive details regarding local addresses, the location of the vulnerabilities, and additional technical information due to the NDA previously signed.

4.6 Conclusion

The penetration testing results show that the application has some vulnerabilities, mainly in terms of configuration and the disclosure of sensitive data (directories, scripts, and additional system information) and browsable directories. Most of these vulnerabilities can be prevented by setting the correct configuration for the application and setting up the proper permissions for the existing directories. However, since Accipiens is hardly reachable to users outside of the organization, these vulnerabilities do not pose a critical threat to the application.

Chapter 5 – Conclusions and Future Work

5.1 Conclusions

In this dissertation, a new methodology to perform web application penetration testing was developed, involving seven stages, similar to the PTES methodology, applying some of the standards within the main sections, performing a scoping meeting, using a similar approach to the intelligence-gathering process, and undergoing the same rules of engagement, while also taking a more active approach towards the application, following the same basics of each stage of the OWASP methodology.

The primary difference and value in the design methodology is its genericity. PTES presents a standard methodology to be followed by both customers and testers, allowing businesses to get more information on specific baselines of work and testers to get more information on what should be taken into account and the activities needed to perform a penetration test. OWASP presents a detailed methodology to test applications during the whole software development life cycle and is directly related not only to the application but also the whole system and its network. In the developed methodology, any tester may find generic methods and tasks to perform a complete penetration test, with the main focus being the application itself and not the whole system.

As a proof of concept, Accipiens software was used as a test case for a penetration test using the defined methodology. Through the appliance of this methodology on a real application scenario that was assessed through previous penetration tests and corrections and having results that go accordingly to standards of two of the most applied methodologies for penetration testing, it is possible to assume that it applies to assessing the security of an application. Each step taken towards this process was explained, and the tools used, ending with the finding of some vulnerabilities.

This investigation adds value to the academic level because it contains not only a comparison between different types of tools and methodologies but also an applicable methodology to test web applications using specific testing standards and tools that ended resulting in no false-positive vulnerabilities found, and even though there were not many vulnerabilities found, some could have severe consequences for the application and its integrity. It is also clear that constant security analysis should be performed on developed software as the risks increase with technological adoption. Even if companies take measures to prevent any security flaw, there is still human error. It is better to prevent than to cure.

5.2 Limitations and Future Work

The limitations and future work are related and can be explained through four main topics: limitation in terms of the chosen tools, the limited test cases, the analysis of vulnerabilities and their exploits, and limitation of the scope.

5.2.1 Limited choice of vulnerability scanners

While performing the pentest, only open-source vulnerability scanners were used. According to the research made, they were chosen based on evaluation, coverage, and accuracy while detecting vulnerabilities and their usage as standard. During the demonstration of the methodology, with the use of the scanners and due to familiarization with the software, some of the configurations and proper use of the tools turned out to be easier as the tests were going through, so it would be recommended to perform some tests to the chosen tools before applying this methodology, as each application is different. However, there are many more scanners to be used; thus, it is recommended to apply this methodology to test other web applications and include more tools within the industry standards.

5.2.2 Limited test cases

As this methodology was only used to test Accipiens, future work can be done by testing and applying this methodology on other web applications to develop it to cover a wider variety of vulnerabilities, using more tools and becoming more generic.

5.2.3 Analysis of vulnerabilities and their exploits

Future work can be done for a more in-depth approach in analyzing vulnerabilities and their exploits as the knowledge applied was based on the research made during the systematic literature review and the vulnerability analysis stage of the methodology. It is recommended to have more expertise towards the penetration testing subject and a better understating of the process behind these methods of assessing security.

5.2.4 Limitation of the scope

Due to the limitation of the scope, certain operative services were excluded from the scope, as well as the database. In order to thoroughly test the developed methodology, future work can be done in order to test the appliance of this methodology to test the entire infrastructure behind the web application; this way, it may be possible to conclude more limitations and tasks to be developed and performed.

Bibliography

Abdul Raman, R. H. (2019). Enhanced Automated-Scripting Method for Improved Management of SQL Injection Penetration Tests on a Large Scale. *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 259–266. https://doi.org/10.1109/ISCAIE.2019.8743936

Alzahrani, M. E. (2018). Auditing Albaha University Network Security using in-house Developed Penetration Tool. *Journal of Physics: Conference Series*, *978*(1). https://doi.org/10.1088/1742-6596/978/1/012093

Auger, R. (2012). *Threat Classification "Taxonomy Cross Reference View."* http://projects.webappsec.org/w/page/13246975/Threat Classification Taxonomy Cross Reference View

Bechtsoudis, A., & Sklavos, N. (2012). Aiming at Higher Network Security through Extensive Penetration Tests. *IEEE Latin America Transactions*, *10*(3), 1752–1756. https://doi.org/10.1109/TLA.2012.6222581

Bernardo Damele A. G., & Miroslav Stampar. (2021). *sqlmap*. https://sqlmap.org/

Cangea, O. (2018). Ethical Hacking Solution to Defeat Cyber Attacks. *Petroleum - Gas University of Ploiesti Bulletin, Technical Series*, *70*(2), 29–36. http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=136232619&site=eds-live

Chen, S. (2016). *Price and Feature Comparison of Web Application Scanners*. SECTOOL Market. http://sectoolmarket.com/price-and-feature-comparison-of-web-application-scanners-opensource-list.html

Christian Mehlmauer. (2021). *Gobuster*. Github. https://github.com/OJ/gobuster

Denis, M., Zena, C., & Hayajneh, T. (2016). Penetration testing: Concepts, attack methods, and defense strategies. *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 1–6. https://doi.org/10.1109/LISAT.2016.7494156

EC-Council. (2020). *TOP PENETRATION TESTING TOOLS THAT ARE EASY TO USE*. https://blog.eccouncil.org/top-penetration-testing-tools-that-are-easy-to-use/

Farah, T., Alam, D., Kabir, Md. A., & Bhuiyan, T. (2015). SQLi penetration testing of financial Web applications: Investigation of Bangladesh region. *2015 World Congress on Internet Security (WorldCIS)*, 146–151. https://doi.org/10.1109/WorldCIS.2015.7359432

Federal Bureau of Investigation - Internet Crime Complaint Center. (2020). *Internet Crime Report*.

Gondim, J. J. C., de Oliveira Albuquerque, R., Nascimento, A. C. A., Villalba, L. J. G., & Kim, T. H. (2016). A methodological approach for assessing amplified reflection distributed denial of service on the internet of things. *Sensors (Switzerland)*, *16*(11). https://doi.org/10.3390/s16111855

Google. (2021). https://www.google.pt/

Gupta, C., Singh, R. K., & Mohapatra, A. K. (2020). A survey and classification of XML based attacks on web applications. *Information Security Journal: A Global Perspective*, *29*(4), 183–198. https://doi.org/10.1080/19393555.2020.1740839

HCL Software. (2021). *AppScan*. https://www.hcltechsw.com/appscan

Hevner, A., & Chatterjee, S. (2010). *Design Research in Information Systems* (Vol. 22). Springer US. https://doi.org/10.1007/978-1-4419-5653-8

"Hobbit." (2021). *Netcat 1.10*. https://nc110.sourceforge.io/

Horvath, I. (2007). Comparison of three methodological Approaches of design research. *16th International Conference on Engineering Design*.

Invicti. (2021a). *Acunetix*. https://www.acunetix.com/

Invicti. (2021b). *Netsparker*. Invicti. https://www.netsparker.com/

ISECOM. (2021). *OSSTMM*. OSSTMM. https://www.isecom.org/OSSTMM.3.pdf

Jain, T., & Jain, N. (2019). Framework for Web Application Vulnerability Discovery and Mitigation by Customizing Rules Through ModSecurity. *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*, 643–648. https://doi.org/10.1109/SPIN.2019.8711673

Josep. (2021). *FOCA*. Github. https://github.com/ElevenPaths/FOCA

Kalirathinam, S. P. (2019). *Penetration Testing Tool for Web Applications*.

Kim, S., Han, H., Shin, D., Jeun, I., & Jeong, H. (2009). A study of international trend analysis on web service vulnerabilities in OWASP and WASC. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 5576 LNCS* (pp. 788–796). https://doi.org/10.1007/978-3-642-02617-1_80

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Krasniqi, G., & Bejtullahu, V. (2018, October 27). Vulnerability Assessment and Penetration Testing: Case study on web application security. *2018 UBT International Conference*. https://doi.org/10.33107/ubt-ic.2018.213

Liu, L., Su, G., Xu, J., Zhang, B., Kang, J., Xu, S., Li, P., & Si, G. (2017). An Inferential Metamorphic Testing Approach to Reduce False Positives in SQLIV Penetration Test. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 675–680. https://doi.org/10.1109/COMPSAC.2017.276

Logic, B. (2021). *Knoxss*. https://knoxss.me/

Metacrawler. (2021). *Metacrawler*. https://www.metacrawler.com/

Morgan, S. (2020). *Global Cybercrime Damages Predicted To Reach $6 Trillion Annually By 2021*. CyberCrime Magazine. https://cybersecurityventures.com/annual-cybercrime-report-2020/

Mudiyanselage, A. K., & Pan, L. (2020). Security test MOODLE: a penetration testing case study. *International Journal of Computers and Applications*, *42*(4), 372–382. https://doi.org/10.1080/1206212X.2017.1396413

Mukhopadhyay, D., Karmakar, S., Meshram, A., & Jadhav, A. (2019). A Prototype of IoT based Remote Controlled Car for Pentesting Wireless Networks. *2019 Global Conference for Advancement in Technology (GCAT)*, 1–7. https://doi.org/10.1109/GCAT47503.2019.8978354

Nagendran, K., Adithyan, A., Chethana, R., Camillus, P., & Bala Sri Varshini, K. B. (2019). Web application penetration testing. *International Journal of Innovative Technology and Exploring Engineering*, *8*(10), 1029–1035. https://doi.org/10.35940/ijitee.J9173.0881019

Nagpure, S., & Kurkure, S. (2017). Vulnerability Assessment and Penetration Testing of Web Application. *2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 1–6. https://doi.org/10.1109/ICCUBEA.2017.8463920

Nmap. (2021). *Nmap*. Nmap. https://nmap.org/

OffSec. (2021). *Kali Linux*. https://www.kali.org/

Org., K. (2021). *Searchsploit*. Kaliorg. https://www.kali.org/tools/exploitdb/

Org., W. (2021). *w3af*. http://w3af.org/

OSP International LLC. (2021). *PMP ITTO Complete Guide (Inputs, Tools, Techniques & Outputs)*. PMP ITTO Complete Guide (Inputs, Tools, Techniques & Outputs).

OWASP. (2021a). *OWASP ZAP*. OWASP. https://owasp.org/www-project-zap/

OWASP. (2021b). *Penetration Testing Methodologies*. https://owasp.org/www-project-web-security-testing-guide/latest/3-The_OWASP_Testing_Framework/1-Penetration_Testing_Methodologies#owasp-testing-guides

Palma Salas, M. I., & Martins, E. (2015). A Black-Box Approach to Detect Vulnerabilities in Web Services Using Penetration Testing. *IEEE Latin America Transactions, 13*(3), 707–712. https://doi.org/10.1109/TLA.2015.7069095

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2020). *Design Science Research Process: A Model for Producing and Presenting Information Systems Research*. http://arxiv.org/abs/2006.02763

Pereira, R., & Serrano, J. (2020). A review of methods used on IT maturity models development: A systematic literature review and a critical analysis. *Journal of Information Technology, 35*(2), 161–178. https://doi.org/10.1177/0268396219886874

Poltavtseva, M. A., & Pechenkin, A. I. (2017). Intelligent data analysis in decision support systems for penetration tests. *Automatic Control and Computer Sciences*, *51*(8), 985–991. https://doi.org/10.3103/S014641161708017X

PortSwigger. (2021a). *ASP.NET ViewState without MAC enabled*. https://portswigger.net/kb/issues/00400600_asp-net-viewstate-without-mac-enabled

PortSwigger. (2021b). *Burp Suite*. PortSwigger. https://portswigger.net/

PTES. (2014). *High Level Organization of the Standard*. Penetration Testing Execution Standard. http://www.pentest-standard.org/index.php/Main_Page

Rapid7. (n.d.). *Metasploit Framework*. Retrieved January 5, 2021, from https://docs.rapid7.com/metasploit/msf-overview/

Rapid7. (2021). *Browsable web directory*. https://www.rapid7.com/db/vulnerabilities/http-generic-browsable-dir/

Repeat, S. (2021). *IP Addresses Found in the Viewstate can be dangerous*. https://scanrepeat.com/web-security-knowledge-base/potential-ip-addresses-found-in-the-viewstate

Riverbed Technology. (2021). *Windump*. https://www.winpcap.org/windump/

Security, O. (2021). *Exploit-DB*. Exploit-DB. https://www.exploit-db.com/

Singh, N., Meherhomji, V., & Chandavarkar, B. R. (2020). Automated versus Manual Approach of Web Application Penetration Testing. *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 1–6. https://doi.org/10.1109/ICCCNT49239.2020.9225385

Skare, M., & Riberio Soriano, D. (2021). How globalization is changing digital technology adoption: An international perspective. *Journal of Innovation & Knowledge*. https://doi.org/10.1016/j.jik.2021.04.001

SonarSource. (2021). *SonarQube*. https://www.sonarqube.org/

Sullo, C. (2021). *Nikto*. https://github.com/sullo/nikto

Tenable. (2021a). *HSTS Missing From HTTPS Server (RFC 6797)*. https://www.tenable.com/plugins/nessus/142960

Tenable. (2021b). *Nessus*. Tenable. https://pt-br.tenable.com/products/nessus

Teodoro, N., & Serrao, C. (2011a). Assessing the Portuguese Web applications security. *2011 World Congress on Internet Security (WorldCIS-2011)*, 21–26. https://doi.org/10.1109/WorldCIS17046.2011.5749875

Teodoro, N., & Serrao, C. (2011b). Web application security: Improving critical web-based applications quality through in-depth security analysis. *International Conference on Information Society (i-Society 2011)*, 457–462. https://doi.org/10.1109/i-Society18435.2011.5978496

Tetskyi, A., Kharchenko, V., & Uzun, D. (2018). Neural networks based choice of tools for penetration testing of web applications. *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 402–405. https://doi.org/10.1109/DESSERT.2018.8409167

The Tcpdump Group. (2021). *tcpdump*. https://www.tcpdump.org/

Vijayalakshmi, K., & Syed Mohamed, E. (2020). Case Study: Extenuation of XSS Attacks through Various Detecting and Defending Techniques. *Journal of Applied Security Research*, 1–36. https://doi.org/10.1080/19361610.2020.1735283

Vondráček, M., Pluskal, J., & Ryšavý, O. (2018). Automated Man-in-the-Middle Attack Against Wi-Fi Networks. *The Journal of Digital Forensics, Security and Law*. https://doi.org/10.15394/jdfsl.2018.1495

Wang, S.-L., Wang, J., Feng, C., & Pan, Z.-P. (2016). Wireless Network Penetration Testing and Security Auditing. *ITM Web of Conferences*, *7*, 03001. https://doi.org/10.1051/itmconf/20160703001

Wireshark. (2021). Wireshark. https://www.wireshark.org/