DEMOCRATIC REPUBLIC OF ALGERIA PEOPLE

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Informatic

DEPARTEMENT of computer science

N° : …….

DOMAIN: Mathematics and Informatics
BRANCH: Computer Science
OPTION: RTIC

A Dissertation in Fulfillment

For the Requirement of the Degree of MASTER

By: Drahmoun Mohammed || Chergui Zineddine

Subject:

Penetration test of web applications

Defended to the jury:

Mezrag Fares	University of M'sila	Chairman
Noureddine Chikouche	University of M'sila	Supervisor
Amroune Nacereddine	University of M'sila	Examiner

Academic year: 2022/2023

DEMOCRATIC REPUBLIC OF ALGERIA PEOPLE

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Informatic

DEPARTEMENT of computer science

N° : …….

DOMAIN: Mathematics and Informatics
BRANCH: Computer Science
OPTION: RTIC

A Dissertation in Fulfillment

For the Requirement of the Degree of MASTER

By: Drahmoun Mohammed || Chergui Zineddine

Subject:

Penetration test of web applications

Defended to the jury:

Mezrag Fares	University of M'sila	Chairman
Noureddine Chikouche	University of M'sila	Supervisor
Amroune Nacereddine	University of M'sila	Examiner

Academic year: 2022/2023

Dedication

We thank our God for giving us the ability to write and think, the strength to believe, the patience to pursue our dreams, and happiness.

We dedicate this modest work to the one who gave us life, a symbol of tenderness, who sacrificed for our happiness and success, our mother.

To our parents, the school of our childhood, who was our shadow during all the years of our studies, and who watched over us throughout our lives to encourage, help, and protect us. May God keep and protect them.

To our lovely sister...

To our brother...

To our friends...

To all those who love us...

We dedicate this work...

Acknowledgements

We thank God for granting us knowledge of science and for helping us to complete this work.

At the end of this modest work, we would like to warmly and respectfully thank all those who contributed directly or indirectly to the completion of this modest end-of-study project, namely our supervisor, Mr. Noureddine Chikouche.

This work was difficult but very beneficial in every way.

Contents

CHAPITRE 3 OWASP top 10 vulnerabilities

CHAPITRE 4 Penetration testing of company's website

List of Figures

GENERAL INTRODUCTION

In today's digital landscape, companies with web applications face an ongoing challenge of defending against cyber-attacks. This defense is typically reactive, meaning that measures are taken in response to newly discovered vulnerabilities and exploits. However, proactive measures such as performing penetration tests can help identify vulnerabilities before they can be exploited by malicious hackers. In our dissertation, we will discuss the importance of penetration testing to improve the security of web applications.

Many companies overlook software security as an essential part of the development life cycle, considering it a cost rather than an investment. This mindset can lead to inadequate measures and increased vulnerability to threats, putting both their systems and stakeholders at risk. Recognizing the value of investing in software security is crucial for protecting assets, reputation, and overall business stability.

How do common attacks operate?

What knowledge and experience are needed to perform the attack and how serious are the consequences of a successful exploit?

What is the meaning of a penetration test?

Do we rely entirely on the programmer to build secure applications or are there other ways to ensure that the application is safe?

We will explore the subject of risk analysis, a crucial process that entails identifying vulnerabilities, assessing the probability of an attack, estimating potential consequences, evaluating severity, and making informed decisions on appropriate courses of action. Moreover, the dissertation will extensively cover prevalent vulnerabilities encountered in web applications by delving into the OWASP top 10, highlighting and examining these critical security risks.

In collaboration with a Company, we will perform a comprehensive penetration test on their web application. This test will involve a combination of manual and automated tests, with the automated tests conducted using well-known tools (Burp Suite, Metasploit, Nmap…). These tools will provide me with the ability to simulate more complex and potentially dangerous attacks (File upload, XSS, RCE…). Overall, our

goal is to help companies mitigate potential risks and stay one step ahead of cyber threats.

Apart from the introductory and concluding sections, this dissertation is structured into four distinct chapters.

In the first chapter, we provide a thorough introduction to the concept of penetration testing, elucidating its importance and methodologies.

Building upon this, the second chapter delves into the evaluation of risks and the crucial process of determining which vulnerabilities should be addressed.

The third chapter revolves around the presentation of the OWASP Top 10 vulnerabilities. This section offers an in-depth analysis of these common security risks prevalent in web applications, shedding light on their significance and potential impact.

Lastly, the fourth chapter centers around the execution of a practical demonstration on Penetration Testing, showcasing our methodology and the outcomes we achieved. This section serves as a tangible showcase of our research, validating our findings and contributions to the field.

CHAPTER 1

Introduction to penetration testing

1.1. Introduction

In 2023, web applications are highly relevant, attracting both developers and threat actors. Common vulnerabilities, like server misconfigurations and weak authentication, are often exploited by hackers. To thrive, companies must prioritize security by promoting secure coding practices, conducting penetration testing, and staying vigilant against evolving exploits. Proactive measures help identify and mitigate vulnerabilities, safeguarding integrity, reputation, and finances.

1.2. Penetration testing

NIST (National Institute of Science and Technology) defines penetration testing "A specialized type of assessment conducted on information systems or individual system components to identify vulnerability that could be exploited by adversaries" [1].

Network security assessments are crucial for robust cybersecurity. They involve examining networks, systems, and applications to find weaknesses. By identifying and addressing vulnerabilities, organizations protect their assets and information from exploitation. Network penetration testing, using simulated attacks, detects entry points and vulnerabilities. It helps fortify defenses, find security gaps, and implement effective remedies. These efforts enhance protection, stopping attackers from exploiting vulnerabilities.

Among the various vulnerabilities in web applications, injection and authentication issues pose significant risks. Injection attacks arise when malicious actors exploit input data to execute harmful code and obtain unauthorized access to critical information. Similarly, authentication vulnerabilities, including weak passwords and inadequate access controls, can enable unauthorized individuals to breach sensitive systems and data. To mitigate these risks, organizations should regularly perform penetration testing. By proactively identifying and addressing these vulnerabilities, they can enhance their security posture and reduce the likelihood of succumbing to cyber-attacks.

1.2.1. Styles of penetration testing

Depending on the extent of information about the system being tested, penetration testing can be categorized into various styles. The level of knowledge may include credentials, technical and language-related specifics, application versions, and the codebase. Three common styles of penetration testing are white, gray, and black box testing. Despite having similar objectives, each style employs distinct methodologies and has its own set of advantages.

The black box style of penetration testing is characterized by having the least amount of knowledge about the system being tested. The only information provided is typically an IP or hostname. This style is often likened to what actual hacking is like for malicious actors in the wild [2], [3]. To discover vulnerabilities that can be exploited, black box penetration testers rely on a combination of manual testing and automated tools. They must also take extensive notes and attempt to map out the network being tested to reverse engineer the inner workings of the applications they are testing. One of the advantages of black box testing is the identification of weaknesses that any hacker could potentially exploit. However, a disadvantage is that if the tester is unable to breach the outer defenses, internal services may not be tested [3].

White box penetration testing is the polar opposite of black box testing, offering full knowledge of the system being tested, including access to source code, documentation, architecture, network maps, credentials, and other related information [2], [3]. With such comprehensive knowledge, white box testers can conduct in-depth code analysis in hopes of uncovering vulnerabilities, covering multiple attack vectors to test both internal and external applications. However, the downside of white box testing is that it can be time-consuming and resource-intensive since there are more resources to work through [4].

Gray box penetration testing serves as the middle ground between white and black box testing. During a gray box penetration test, the tester is given access to certain credentials, providing an opportunity to assess the extent of the damage that a privileged user could cause [5]. With focused testing, gray box testing saves time and money without requiring tedious reconnaissance. The primary differences between these styles boil down to speed, efficiency, and coverage. Companies must determine how much time and money they are willing to allocate to testing, as well as the perspective from which they want their applications tested.

1.3. Professionals and adversaries

1.3.1. White hat hackers

Penetration testing is a critical process that should be performed by professionals with extensive experience in the relevant technologies. This expertise may come from developing similar applications, working with comparable system structures, and having a deep understanding of the vulnerabilities associated with the technologies involved. Ethical hackers, also known as white hat hackers, are the professionals who conduct

penetration testing to identify security flaws and weaknesses in applications and systems [6].

1.3.2. Adversaries

In today's digital landscape, businesses are confronted with an ongoing and relentless challenge from black hat hackers. These individuals, possessing advanced knowledge in computer science, employ their skills to identify and exploit security weaknesses. Operating independently or as part of organized groups, these malicious hackers are primarily motivated by personal gain. Their arsenal includes tactics such as extortion, blackmail, corporate espionage, and the theft of sensitive data, which they employ strategically to accomplish their objectives. The ever-evolving threat landscape necessitates that companies remain vigilant and continuously fortify their cybersecurity defenses to protect against these determined adversaries [7].

1.3.3. Collaborative security

The protection of IT systems and applications is of utmost importance for both private individuals and companies [7]. However, achieving perfection in development is a far-fetched goal, as human error can result in vulnerabilities. Besides, applications usually involve multiple programming languages and technologies developed by third-party companies, making the security of the system dependent on external factors. Therefore, it is crucial to educate developers on secure coding practices and conduct regular penetration testing to assess the security of applications. These two fundamental cybersecurity measures form the basis of the Blue team vs. Red team approach, where both teams collaborate to achieve a common goal of improving organizational security posture.

Red team and Blue team are two key components of cybersecurity. The Red team, also known as the "ethical hackers," are responsible for simulating attacks on systems and applications to identify vulnerabilities that can be exploited by real hackers. They use various tools and techniques to test the security of a system or application, and their goal is to find as many vulnerabilities as possible. On the other hand, the Blue team, also known as the "defenders," are responsible for implementing security measures to protect against attacks. They develop and deploy security measures such as firewalls, intrusion detection systems, and access controls to protect the system. The back-and-forth between the Red team and the Blue team helps to identify vulnerabilities and improve the security of the system. It is important for both teams to work together to create a secure environment [8].

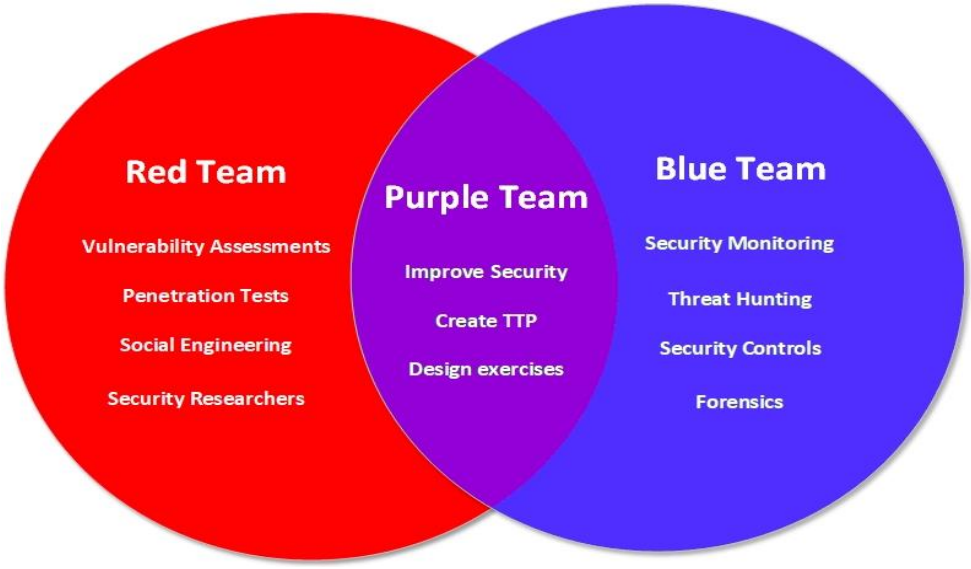


Figure 2.1: Collaboration between Red Team and Blue Team, create a cooperative mindset [9]

1.4. Security attributes

Companies must prioritize general strategies to enhance their security effectively. These strategies encompass proper data handling, both in storage and during transit. Additionally, the careful management of user input is crucial, as it is a prominent source of vulnerabilities. Lastly, measures are implemented to steer clear of pitfalls such as relying on security through obscurity.

1.4.1. CIA Triad

The CIA triad forms a fundamental aspect of information security, encompassing three key concepts related to user data. These concepts include confidentiality, integrity, and availability. Confidentiality ensures that user data remains undisclosed to unauthorized individuals or entities [10]. This can be achieved through encryption, access controls, and cryptographic techniques such as hashing and salting. Integrity focuses on maintaining data in an unaltered state, preventing unauthorized and undetected modifications [10]. Upholding integrity involves restricting access to sensitive data and implementing measures to protect against data breaches. Availability aims to ensure uninterrupted access to information within an application [10]. This necessitates the continuous operation of systems responsible for storing, processing, protecting, and serving information, even in the face of power outages or disruptive attacks like DoS. Building resilient applications capable of withstanding external pressures and implementing backup procedures for contingencies are key to upholding availability.

1.4.2. Security through obscurity

Security through obscurity refers to the practice of relying on adversaries' lack of knowledge about how a security system or application works, as a means of protecting it from exploitation. However, this approach is widely considered a flawed methodology of security, as once adversaries gain information on how the system works, they can exploit it. This information may be obtained through reverse engineering, educated guesses, or other means. Instead of relying on security through obscurity, best practices dictate creating robust security systems and subjecting them to penetration testing to identify vulnerabilities that can be fixed. This approach ensures that even if adversaries have knowledge of how the application works, there are no known vulnerabilities for them to exploit [11].

1.4.3. Handling user input

To ensure the safety of user input, developers must implement measures such as input validation and sanitization. Input validation ensures that user input adheres to the correct format, preventing errors and unintended behavior. On the other hand, input sanitization involves removing malicious content from user input to prevent vulnerabilities like XSS and injections, typically performed on the server side. Server-side validation and sanitization are crucial for security, as they prevent tampering by malicious users. Client-side validation primarily enhances the user experience by providing feedback on input accuracy [12].

There are two approaches to input sanitization: whitelisting and blacklisting. Whitelisting involves maintaining a list of acceptable and safe inputs, discarding anything not on the whitelist. Although considered best practice, whitelisting can be strict and inconvenient for users. In contrast, blacklisting involves listing disallowed and dangerous inputs. However, blacklists are often deemed insufficient since they only protect against known inputs, leaving the application vulnerable to new or bypassed threats.

By combining proper input validation and sanitization techniques, developers can strengthen the security of user input and safeguard their applications effectively.

1.4.4. Same-origin policy

The Same-Origin Policy (SOP) serves as a crucial web browser security mechanism that prevents cross-origin attacks between websites. This policy restricts interactions between documents or scripts from different origins, which are defined by the

combination of the HTTP protocol, domain name, and port number [13]. Any requests from a different origin are discarded, effectively preventing unauthorized access.

By enforcing SOP, web browsers prevent scenarios where malicious websites could exploit vulnerabilities to steal cookies or access data from other authenticated domains. The introduction of the same-origin policy dates back to Netscape Navigator 2.0, and it has since been adopted by other popular web browsers. SOP has become an indispensable security feature in modern browsers, ensuring the protection of user data from malicious attacks.

CHAPTER 2

Risk evaluation

2.1. Introduction

The Open Web Application Security Project (OWASP) is a nonprofit organization committed to enhancing software security. Their mission centers around empowering organizations to create and sustain secure applications and APIs, ultimately striving to ensure robust security measures. OWASP achieves this through a diverse range of resources, including the provision of application security tools, comprehensive educational materials like books and videos dedicated to application security, concise cheat sheets, and standardized security controls and libraries. These offerings serve as invaluable assets for developers and organizations aiming to fortify the security of their software solutions.

2.2. OWASP Risk Rating Methodology

Web applications are vulnerable to various potential attacks that can harm organizations. Each of these vulnerabilities represents a path with a possible risk, where a successful attack could cause significant harm to the business. To assess the risk associated with these vulnerabilities, OWASP has developed a Risk Rating Methodology. This methodology divides risk into six factors, providing a comprehensive framework to evaluate the severity of any given vulnerability. The OWASP Risk Rating Methodology is an effective tool to help companies prioritize their security efforts and protect their applications from potential threats [14].

2.3. Identifying a Risk

Once a security vulnerability is found, the tester needs to learn as much about the vulnerability as possible [15]. This is done by gathering information about threat agents, the attack that is performed, the vulnerability involved, how much damage the attack can do and the possible business impact.

2.4. Factors for Estimating Likelihood

When assessing potential risks, it's important to take into account the potential impact. Once a vulnerability has been identified and the likelihood of exploitation has been estimated, the next step is to estimate the "impact". This is a measure of how much damage could be done if an attacker were to successfully exploit the vulnerability. As with the likelihood estimate, it's not necessary to be overly precise here. A rough estimate of low, medium, or high is generally sufficient.

To estimate the impact, there are a number of factors that should be considered. These include the value of the asset that could be affected, the extent of the damage that

could be caused, and the potential harm to individuals or the organization as a whole. It's important to consider both the immediate and long-term impact, as well as the potential for secondary effects.

Each factor has a set of options, and each option has an impact rating from 0 to 9 associated with it. These numbers will be used later to estimate the overall impact. By considering both the likelihood and impact of a potential risk, testers can determine which vulnerabilities are most critical and prioritize their efforts accordingly [15].

2.4.1. Threat Agent Factors

The primary focus revolves around the threat agent in question, aiming to determine the probability of a successful assault orchestrated by this group. The assessment centers on evaluating the likelihood of victory in an attack by this particular set of adversaries.

- **Skill level:** Minimal understanding (1) implies basic technical skills (3) and advanced computer proficiency (5). It encompasses familiarity with networking, programming (6), and security penetration skills (9).
- **Motive:** The motivation factor holds significant importance in assessing this group of threat agents. It pertains to their drive and inclination to identify and exploit the vulnerability at hand. Evaluating their level of motivation helps gauge the potential risk they pose. On a scale of 1 to 10, where (1) signifies low or no reward, (4) represents a possible reward, and (9) indicates a high reward, we can determine the extent of their motivation.
- **Opportunity:** The aspect of opportunity is crucial to consider when evaluating this group of threat agents. It pertains to the resources and opportunities necessary for them to identify and exploit the vulnerability in question. By examining the level of access and resources required, we can ascertain their potential capabilities. Using a scale of 0 to 9, where (0) signifies full access or expensive resources required, (4) represents special access or resources required, (7) denotes some access or resources required, and (9) indicates no access or resources required, we can determine the degree of opportunity available to this group.
- **Size:** Evaluating the size factor helps determine the scale of potential threats. Using a scale of 1 to 10, where (1) represents a small group and 10 signifies a large group, we can gauge the size of this group of threat agents. For instance, a rating of (2) suggests a group consisting of developers or system administrators, a rating of (4)

indicates intranet users, a rating of (5) represents partners, a rating of (6) denotes authenticated users, and a rating of (9) signifies anonymous Internet users, emphasizing the varying degrees of size within this group.

2.4.2. Vulnerability Factors

The subsequent set of elements pertains to the vulnerability itself. These factors are aimed at estimating the probability of the specific vulnerability being discovered and exploited. The assumption is made that the threat agent selected above remains the focal point. By assessing the vulnerability factors, we gain insights into the potential risks associated with its discovery and exploitation.

- **Ease of discovery:** The consideration of discovery ease is a critical factor when evaluating this specific group of threat agents. It revolves around assessing how readily they can stumble upon the vulnerability in question. By evaluating the level of ease, we can determine the potential risk associated with its discovery. Employing a rating scale from 1 to 10, where (1) signifies it is extremely improbable to discover, and (10) indicates the availability of automated tools, we can assess the ease of discovery for this group of threat agents. A rating of (3) suggests it is considerably challenging, while a rating of (7) denotes it is relatively easy for them to discover the vulnerability.
- **Ease of exploit:** The consideration of exploit ease is a crucial factor when evaluating this particular group of threat agents. It revolves around assessing the level of vulnerability exploitability. By determining the degree of ease, we can ascertain the potential risk associated with actual exploitation. Utilizing a rating scale from 1 to 10, where (1) signifies a highly challenging theoretical exploit and (10) indicates the availability of automated tools, we can assess the exploit ease for this group of threat agents. A rating of (3) suggests it is significantly difficult, while a rating of (5) denotes a relatively manageable level of ease for them to exploit the vulnerability.
- **Awareness:** The factor of awareness is essential when evaluating this group of threat agents. It pertains to their knowledge and familiarity with the vulnerability under consideration. Assessing their awareness helps determine their level of understanding and potential exploitation. Using a scale of 1 to 10, where (1) represents the vulnerability being unknown to them and (10) indicates it being public knowledge, we can gauge the extent of awareness among this group of threat agents. A rating of (4) suggests the vulnerability is hidden from them, while a rating of (6) implies it is relatively obvious. Finally, a rating of (9) signifies that the vulnerability is widely known and accessible to this group.

- **Intrusion detection:** The evaluation of intrusion detection plays a vital role in determining the chances of detecting an exploit. It focuses on the effectiveness of detection measures in place to identify and respond to potential breaches. By analyzing the intrusion detection aspect, we can assess the level of risk associated with undetected exploits. We employ a rating scale ranging from 1 to 10 to gauge the likelihood of detection for this particular group of threat agents. A rating of (1) indicates active detection within the application, (3) represents logged and reviewed detection, (8) denotes logged detection without subsequent review, and (9) signifies no logging of the exploit.

2.5. Factors for Estimating Impact

When evaluating the consequences of a successful attack, it is crucial to take into account the technical impact on the application and data, as well as the business impact on the company. While the business impact holds greater significance, testers may lack all the essential information to assess it accurately. Providing comprehensive details about the technical risk can assist in informing the relevant business representative to make an informed decision. Similar to the likelihood assessment, impact factors are rated on a scale from 0 to 9, which will be utilized later to estimate the overall impact [15].

2.5.1. Technical Impact Factors

The technical impact can be assessed by considering traditional security areas: confidentiality (C), integrity (I), availability (A), and accountability (Ac). These factors help estimate the impact on the system if the vulnerability is exploited. Evaluating C, I, A, and Ac allows for determining the magnitude of the impact on the system in case of exploitation. [15].

- **Loss of confidentiality:** The level of confidentiality at risk and the extent of potential data exposure varies across different scenarios. In cases where minimal non-sensitive data is disclosed, the impact on confidentiality is relatively low, earning a rating of (2). Similarly, when minimal critical data is exposed, the consequences can be significant but limited, warranting a rating of (6). On the other hand, if extensive non-sensitive data is disclosed, the potential breach of confidentiality increases to a rating of (6), highlighting a higher level of concern. Moreover, when extensive critical data is compromised, the impact on confidentiality escalates to a rating of (7), signifying a

substantial breach. Finally, in the worst-case scenario where all data is disclosed, the breach of confidentiality reaches its maximum severity with a rating of (9), necessitating immediate action to mitigate the consequences.

- **Loss of integrity:** The extent of data corruption and the resulting damage may vary in different situations. When minimal data is slightly corrupted, the impact on integrity is minimal, warranting a rating of (1). However, if minimal data is seriously corrupted, the level of damage escalates, earning a rating of (3). In cases where extensive data is slightly corrupted, the integrity of the information is significantly compromised, meriting a rating of (5). Furthermore, if extensive data is seriously corrupted, the damage to data integrity becomes more severe, warranting a rating of (7). Finally, in the worst-case scenario where all data is completely corrupted, the integrity of the entire dataset is compromised, resulting in a rating of (9). In such cases, immediate action is imperative to mitigate the consequences and restore data integrity.
- **Loss of availability:** The potential loss of service and its significance can vary depending on the situation. When minimal secondary services are interrupted, the impact on availability is relatively low, earning a rating of (1). On the other hand, if minimal primary services are interrupted, the consequences become more substantial, warranting a rating of (5). Similarly, in cases where extensive secondary services are interrupted, the availability of essential services is significantly affected, meriting a rating of (5). Moreover, if extensive primary services are interrupted, the impact on availability escalates to a rating of (7), indicating a substantial loss. Finally, in the worst-case scenario where all services are completely lost, the availability of critical services is compromised entirely, resulting in a rating of (9). In such cases, immediate action is crucial to restore services and minimize the impact on operations.
- **Loss of accountability:** Is it possible to trace the actions of the threat agents to an individual? The level of traceability can vary across different scenarios. If the actions are fully traceable back to an individual, it suggests a high degree of accountability, warranting a rating of 1. However, if the traceability is only possible to a certain extent, with some potential leads or evidence, it falls under the category of possibly traceable, earning a rating of 7. Conversely, if the threat agents' actions are completely anonymous and leave no identifiable traces, it indicates a significant challenge in attributing responsibility, resulting in a rating of 9. The level of traceability plays a crucial role in investigations and determining appropriate measures to address the threat.

2.5.2. Business Impact Factors

The technical impact of security problems in a company's application can have a business impact, which is important to understand in order to justify investment in fixing security issues. It is recommended to support risks with business impact, especially when presenting to executives. Companies may have asset classification guides or business impact references to help prioritize security. If not available, it's important to consult with people who understand the business to identify what's important. Common areas for businesses to consider for security are unique to each company [15].

- **Financial damage:** Assessing the potential economic impact resulting from an exploit is vital. Will it be less than the costs associated with addressing the vulnerability (1), have a minor impact on annual revenue (3), significantly affect annual revenue (7), or potentially even lead to financial insolvency (9)? Understanding the severity of a vulnerability and its potential consequences on the overall financial stability of an organization hinges upon evaluating the potential economic ramifications.
- **Reputation damage:** If an exploit were to occur, what could be the potential reputation impact that could harm the business? Would it result in minimal damage to the reputation (1), the loss of major accounts (4), a loss of goodwill (5), or even significant brand damage (9). Evaluating the potential repercussions on reputation is crucial in gauging the severity of an exploit and its potential consequences on the overall image and trustworthiness of a business.
- **Non-compliance:** To what extent does non-compliance introduce vulnerability? Is it a minor violation (2), a clear violation (5), or a high-profile violation (7). Assessing the level of non-compliance exposure is vital in understanding the potential risks and consequences associated with failing to meet regulatory or legal requirements. It helps determine the severity of the situation and the necessary steps to address and mitigate any potential repercussions stemming from non-compliance.
- **Privacy violation:** How much personally identifiable information could be at risk in the event of a privacy violation? Would it affect just one person (3), a few hundred people (5), several thousand people (7), or potentially even millions of people (9)? It's essential to evaluate the potential extent of a privacy breach to grasp the magnitude of its impact on individuals' sensitive information. This assessment plays a crucial role in understanding the severity of the violation and determining the necessary

measures to address and mitigate any potential harm resulting from the disclosure of personally identifiable information.

2.6. Determining the Severity of the Risk

When it comes to managing risks, it is crucial to evaluate the severity of each risk. A widely used approach involves employing a scale that ranges from 0 to 9. This scale considers various factors, including the identified threats and vulnerabilities discovered during the earlier steps. By calculating a score based on these factors, the risk can be classified as low, medium, or high. Such classification is significant as it assists in prioritizing which risks should be addressed first, based on their potential impact on the organization [15].

Likelihood and Impact Levels	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

Table 2.1: Likelihood and impact levels.

2.7. Deciding What to Fix

Once the risks associated with the application have been identified and classified, it becomes essential to prioritize them accordingly. The main objective is to address the most severe risks first, as they pose the greatest threat to the organization. It can be tempting to focus on fixing less critical risks that are easier or cheaper to address, but this approach fails to enhance the overall risk profile of the application. In fact, by neglecting the most severe risks, the organization exposes itself to potential attacks that could lead to serious consequences. Therefore, it is crucial to allocate resources and efforts towards mitigating the most critical risks initially, in order to ensure the highest level of protection for the organization's assets [15].

2.8. Customizing the Risk Rating Model

Having a customizable risk ranking framework is crucial for businesses to effectively adopt and implement it. Customization helps in producing results that align with the organization's perception of what constitutes a serious risk. The framework can be tailored by adding factors that are important to the organization, customizing options

associated with each factor, and weighting factors to emphasize those that are more significant for the business. Tuning the model can be done by comparing the risk ratings produced by the model with those produced by a team of experts and adjusting the scores accordingly. This will help in accurately identifying the most critical risks and prioritizing them accordingly [15].

CHAPITRE 3

OWASP top 10 vulnerabilities

3.1. Introduction

The OWASP Top 10 is a well-known compilation of the 10 most prevalent application vulnerabilities that organizations should take into account when assessing the security of their web applications. It serves as a valuable reference for identifying potential vulnerabilities that attackers may exploit. In this chapter, we will delve into each vulnerability listed in the OWASP Top 10 and provide a comprehensive explanation of their mechanics, along with recommended measures to address and rectify them. By comprehending these vulnerabilities and implementing the appropriate actions to mitigate them, organizations can substantially decrease their risk of falling victim to successful attacks on their web applications.

3.2. Injection

Injection attacks occur when untrusted data is sent to an interpreter via various attack vectors such as environment variables, internal and external web services, and parameters. The main objective of injection attacks is to trick the interpreter into executing malicious data, often through structured queries and commands. Languages and applications such as SQL, LDAP, XPath, NoSQL, command injection, XML parsers, and more are vulnerable to these attacks. Injection is considered the most severe vulnerability due to several factors. It is a relatively easy vulnerability to detect, and many applications will reveal sensitive information about how they function when they error out. Additionally, injection attacks are easy to exploit, with many requiring minimal effort. Successful injection attacks can have catastrophic impacts on both technical and business factors as they can reveal, alter, or delete sensitive data, and in some cases, lead to remote code execution, compromising systems [16].

An example vulnerable query would be: String query = “SELECT * FROM accounts WHERE custID=’” + request.getParameter(“id”) + “‘“;. To exploit this all one needs to do is send the payload ‘or 1=1 in the URL parameter id, example: http://example.com/apps/account?id=’or 1=1. When sent the payload will be executed, and since 1=1 is always TRUE the entire table will be returned.

3.2.1. How to Prevent Injection

To prevent injection attacks, there are various options available. One approach is to use a server-side whitelist, which can help with input validation. However, this method is not foolproof since many applications need to allow special characters that could be part of injection attacks. Another effective way to prevent injection attacks is by using

parameterized queries, also known as prepared statements, which are supported in languages such as Java, PHP, and C#. Prepared statements are pre-compiled queries that only require parameters to function, thereby separating user input from vulnerable interpreters and preventing injection attacks. Another preventive measure that can be useful is to use the LIMIT keyword from SQL, which can limit the number of records returned by any query, thus preventing data leaks in case of successful injection attacks.

3.3. Broken authentication

Broken authentication vulnerabilities occur when attackers exploit errors in authentication logic, allowing them to bypass or manipulate the login process. This includes session management attacks. Examples of such attacks include injection attacks, password spraying, exploiting default credentials, and session management attacks. Broken authentication is prevalent as most web applications require authentication for user access. Modern web authentication relies on cookies, where a generated cookie is sent with each request to the web server after successful authentication [17].

The consequences of broken authentication can be severe, compromising user accounts and potentially compromising administrator accounts. Compromised accounts may expose sensitive information, while compromised administrator accounts can lead to server and application compromises. Administrators often have access to powerful tools and controls that can be exploited for executing malicious code. Addressing broken authentication vulnerabilities is crucial to mitigate these risks.

3.3.1. How to Prevent Broken authentication

To mitigate broken authentication risks, implement effective measures such as multi-factor authentication, avoiding default credentials, enforcing strong passwords, limiting login attempts, and using secure session management. These steps bolster security, reducing the likelihood of unauthorized access to user accounts and sensitive information.

3.4. Sensitive data exposure

Sensitive data exposure is a critical vulnerability that can have catastrophic impacts on an organization. These attacks occur when sensitive data is not properly encrypted, making it vulnerable to exploitation by malicious actors. Data at rest, such as stored credentials, and data in transit, which is data traveling between clients and servers, are both at risk of sensitive data exposure. Man-in-the-middle (MITM) attacks pose a

significant risk to data in transit as attackers can intercept and manipulate data transmitted between two parties [18].

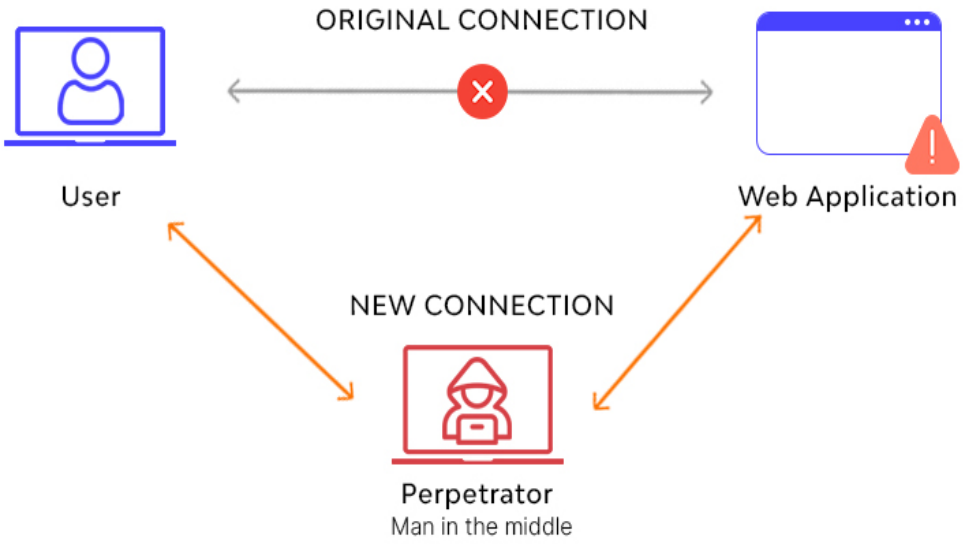


Figure 3.1: Illustration of Man in The Middle attack [19].

Hackers frequently target vulnerable storage systems or databases that contain sensitive information such as personal data, payment details, or intellectual property. Exploiting security flaws or weaknesses in these systems can result in data breaches and the exposure of sensitive data, causing significant harm to individuals and organizations alike. The consequences of compromised personal data can include identity theft, financial loss, and reputational damage. Furthermore, companies that neglect to safeguard sensitive data may face legal and financial repercussions under data protection regulations. Therefore, it is crucial to prioritize implementing robust security measures and encryption techniques to protect data at rest and maintain compliance with relevant laws and regulations.

3.4.1. How to Prevent Sensitive data exposure

To safeguard sensitive data from exposure, it is crucial to implement appropriate protective measures. Firstly, it is important to identify and classify data according to its sensitivity, in accordance with relevant laws and regulations. Unnecessary data should be avoided and promptly discarded when no longer needed, minimizing potential targets for leaks. Strong encryption protocols should be employed to protect data at rest, ensuring its security. Credentials should be hashed using robust hashing functions and

salted, preferably incorporating delay factor functions like Argon2 or bcrypt. These steps enhance data protection and reduce the risk of sensitive data exposure.

When it comes to data in transit, encryption protocols like Secure Socket Layer (SSL) through HTTPS should be implemented, with keys approved by trusted Certificate authorities. This will ensure that the data communicated between servers and clients are encrypted and, therefore, protected against MITM attacks. Failure to implement such measures could lead to severe consequences, including financial loss, reputation damage, and legal action.

3.5. XXE

XML External Entities (XXE) is a security vulnerability that arises from misconfigured XML processors handling data from untrusted sources. Exploiting this vulnerability allows attackers to inject malicious data, potentially leading to code execution or unauthorized file access, thereby compromising the server. These attacks can also enable server-side request forgeries, where an attacker incorporates a defined entity through a target URL, causing the application's response to include the response from that URL [20].

Request	Response
POST /endpoint HTTP/1.1 <!--?xml version="1.0" ?--> <!DOCTYPE foo [<!ELEMENT foo ANY> <!ENTITY xxe **SYSTEM** **"file:///etc/passwd"**>]> <details> <second>**&xxe;**</second> </details>	HTTP/1.1 200 OK root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh (...)

Figure 3.2: XXE payload that reads the /etc/passwd file of the vulnerable server [21]

A classic example of XXE is this payload, where an attacker attempts to read the sensitive file /etc/passwd to gain insight into which users are on a server.

When the attacker sends the following XML data with the XXE payload included, the XML parser processes external entities which ends up the server returning the contents of the internal file /etc/passwd.

3.5.1. How to Prevent XXE

To prevent XXE attacks, several measures can be taken to enhance application security. One of the most effective measures is to use less complex data formats such as JSON, and to avoid serialization of sensitive data. Additionally, it is crucial to ensure

that all XML processors, libraries and dependencies are kept up to date by regularly patching them. Another important step is to disable XML external entity and DTD processing in all XML parsers wherever possible, as this will prevent any exploitation of XXE vulnerabilities. Moreover, implementing server-side whitelisting within XML documents, headers or nodes can be extremely effective in improving security posture, as it will make it less likely for payloads to succeed even if an XXE vulnerability exists.

3.6. Broken Access Control

Access control is crucial to the security of any application, as it restricts who can perform certain actions or access certain resources. In web applications, access control is often tied to authentication and session management, which identify users and track their activity [22].

There are various access control structures, including Mandatory Access Control (MAC), Role-Based Access Control (RBAC), and Discretionary Access Control (DAC). MAC uses strict policies that an administrator controls to determine which users can access which resources. Users cannot modify these permissions. RBAC assigns permissions based on the user's role, and a user can belong to multiple groups. DAC grants permissions to users to access specific objects, and they can further grant access to other users [22].

However, implementing secure access control can be difficult, as it can vary significantly between applications and can be prone to errors and exploitation. Developers need to thoroughly test access control by attempting to exploit the logical structure to identify potential vulnerabilities [22].

Broken access control vulnerabilities can manifest in various ways, highlighting the importance of robust security measures. One scenario involves an exposed tool that possesses privileged functions or access to sensitive data, but lacks adequate authentication safeguards. Exploiting this weakness allows attackers to masquerade as administrative users, gaining access to critical functionalities that can potentially compromise sensitive information or even compromise the servers themselves. Mitigating these risks requires implementing effective access control mechanisms to prevent unauthorized access and ensure the protection of sensitive resources.

3.6.1. How to Prevent Broken Access Control

To ensure robust access control, it is essential to implement server-side mechanisms that prevent unauthorized modification of access control functionalities by potential

attackers. Restricting unprivileged users to the minimum necessary privileges is also crucial for maintaining security. By default, all non-public resources should be denied access to enhance protection. Implementing model access controls can further enforce record ownership, preventing unauthorized creation, reading, updating, or deletion of records. Additionally, disabling web server directory listing and ensuring the absence of common backup files in web roots contribute to a more secure environment. It is equally important to log access control failures for effective monitoring and detection of potential security breaches.

3.7. Security misconfigurations

Security misconfigurations can have a significant impact on the security of a system. These misconfigurations typically occur due to unpatched vulnerabilities, default accounts, unused pages, unprotected files and directories, and other similar issues. When these vulnerabilities are successfully exploited by attackers, they can gain unauthorized access to the system or even execute malicious code, leading to system compromise [23].

Security misconfigurations can be exploited easily with automated tools, as well as through improper error handling that reveals sensitive information like version numbers. This information can be used to find detailed explanations or premade scripts that can exploit the vulnerabilities, which are often available through online resources.

3.7.1. How to Prevent Security misconfigurations

To prevent security misconfiguration vulnerabilities, keep applications updated and regularly check for framework vulnerabilities. Remove unnecessary features, components, documentation, and samples. Implement a segmented application architecture with access control lists for secure component separation.

3.8. XSS

Cross-Site Scripting (XSS) attacks are a form of injection attack where malicious scripts are inserted into websites. These attacks exploit the vulnerability of a web application that fails to properly sanitize and validate user input, allowing the execution of the injected malicious code. XSS poses a significant risk as it bypasses the Same Origin Policy, which is designed to isolate websites from one another. The three main types of XSS attacks are reflected XSS, stored XSS, and DOM-based XSS. It is crucial for web applications to implement robust input sanitization and validation mechanisms to mitigate the risk of XSS vulnerabilities.

In a stored XSS attacks the injected malicious code is permanently stored on the target servers. In this type of attack, attacker first tries to find vulnerability in web application. If such vulnerability is present, he injects a malicious script that will be able to steal user's confidential information or cause other damages. The script then resides permanently on the server and therefore, when any user access this information through web application, the malicious script gets executed and the confidential information becomes accessible to attacker. Stored XSS attacks are generally performed on web applications that takes input from user in the form of text and store it in the database of the web application [24].

As opposed to stored XSS attacks, in reflected XSS attacks the injected code doesn't reside on the web server. In reflected attacks, malicious links are sent to victims using email, or embedding the link in a web page residing on another server. When user clicks on this link, the injected code goes to attacker's web server, which sends the attack back to victim's browser. Then, browser executes the code because it comes from a trusted server. In this way an attacker bypass the same origin policy. When this code executes on browser, it performs the malicious work like stealing the confidential information of victims [24].

DOM Based XSS is an XSS attack where the DOM environment in the victim's browser is modified by the original client-side script, so that the client-side code runs in an unexpected manner. In this kind of attack, the page doesn't change but the client-side code gets executed in a different manner because of the modification in the DOM environment. It is different from the other two XSS attacks as the attack is executed at the client side [24].

3.8.1. How to Prevent XSS

Cross-Site Scripting (XSS) attacks can be prevented by separating untrusted data from active browser content on websites. This can be achieved by using frameworks that automatically escape XSS, like Ruby on Rails or React JS. Sanitizing user input by using filters can also decrease the risk of XSS attacks. Additionally, using the HTTP Only attribute in cookies can greatly decrease the damage from successful attacks as it prevents JavaScript from accessing cookie values.

3.9. Insecure Deserialization

Serialization converts objects into bytes, while deserialization does the opposite [25]. It simplifies object storage and transmission across networks and is supported in

languages like Python, Java, Ruby, and PHP. However, deserialization can be a security risk when user-controlled objects are involved [25]. Malicious manipulation of these objects can lead to code execution. Exploiting deserialization is challenging due to complex payloads and limited automated tools for detection. It requires deep knowledge of language-specific libraries and often involves blind exploitation. As a result, insecure deserialization is not ranked higher in the OWASP top 10 list [25].

3.9.1. How to avoid Insecure Deserialization

Preventing insecure deserialization poses challenges, with one viable solution being the implementation of cryptographic checksums to ensure object integrity. Given the difficulty of prevention, the best approach is to avoid using serialization and deserialization in applications where user input is accepted.

3.10. Using Components with Known Vulnerabilities

Exploiting known component vulnerabilities occurs when organizations lack oversight on the components used in their applications, both server-side and client-side [26]. Security researchers and bug bounty hunters actively search for and report vulnerabilities in the Common Vulnerabilities and Exposures (CVE) list, maintained by The MITRE Corporation. This widely used list provides valuable information on vulnerabilities that can affect applications and their components, including detailed exploit descriptions and proof of concept. Attackers can easily find existing exploits through internet searches once they identify vulnerable component versions [26].

3.10.1. How to avoid Using Components with Known Vulnerabilities

To effectively avoid using components with known vulnerabilities, it is important to have a robust patch management process in place. This process should include the removal of unused dependencies, unnecessary features, components, files, and documentation. Regularly checking the versions of components on both server and client sides and updating them whenever possible is also crucial to avoid using vulnerable components. Monitoring CVE's and vulnerabilities relevant to your components is another key component of a solid patch management process. Additionally, having a plan for monitoring and applying updates for components throughout the lifetime of your application is essential for keeping your application secure against known vulnerabilities. By following these steps, we can significantly reduce the risk of using components with known vulnerabilities and keep the application secure.

3.11. Insufficient Logging and Monitoring

Monitoring and logging play crucial roles in maintaining application security. Monitoring entails vigilant observation of security incidents such as login activities, failed login attempts, critical transactions, and events blocked by web application firewalls or similar security mechanisms. On the other hand, logging involves systematically collecting and storing data pertaining to monitoring, enabling thorough analysis of security-related events. In worst-case scenarios, logs can even be utilized to reconstruct specific attacks executed by malicious actors, providing invaluable insights for security experts to understand vulnerabilities and devise effective solutions [27].

Insufficient logging and monitoring in web applications pose a significant security challenge. Strong logging and monitoring practices are essential to enhance application security. Without them, the risk of successful attacks increases. A comprehensive defense strategy includes both preventive and reactive measures, with vigilant monitoring of suspicious activity. Inadequate logging hinders understanding of vulnerability exploitation, making issue resolution difficult and leaving applications vulnerable. Robust logging and monitoring practices are vital for web application security [27].

3.11.1. How to Prevent Insufficient Logging and Monitoring

Sufficient logging and monitoring are vital for web application security. It involves logging security events, such as failed logins and access control failures, with detailed context. Logs must be securely stored with backups. Effective monitoring systems should promptly alert security professionals of potential threats. Swift action should be taken following attack detection, guided by a defined protocol. Proper logging and monitoring practices enable timely incident response and vulnerability identification, enhancing web application security.

CHAPITRE 4

Penetration testing of company’s website

4.1. Introduction

In this chapter, we will present the penetration testing process we conducted on the web application of a company. We will present the security vulnerabilities found on the website and the method of exploiting these vulnerabilities.

4.2. Tools used

This section provides a comprehensive list of software used for conducting thorough penetration test, accompanied by descriptions of their specific applications. These tools play a crucial role in assessing the security vulnerabilities and weaknesses within a system or network.

4.2.1. Burp Suite

Burp Suite is a powerful tool that plays a crucial role in security testing and vulnerability assessment for web applications. Developed by PortSwigger, a reputable provider of web security tools, Burp Suite offers a comprehensive array of features and functionalities. These capabilities enable security professionals to effectively identify and address potential security weaknesses within web applications, enhancing overall security measures [28].

Burp Suite is a comprehensive and widely used tool in the field of web application security testing. Its intuitive interface, extensive range of features, and regular updates make it a popular choice among security professionals for identifying and remediating web application vulnerabilities.

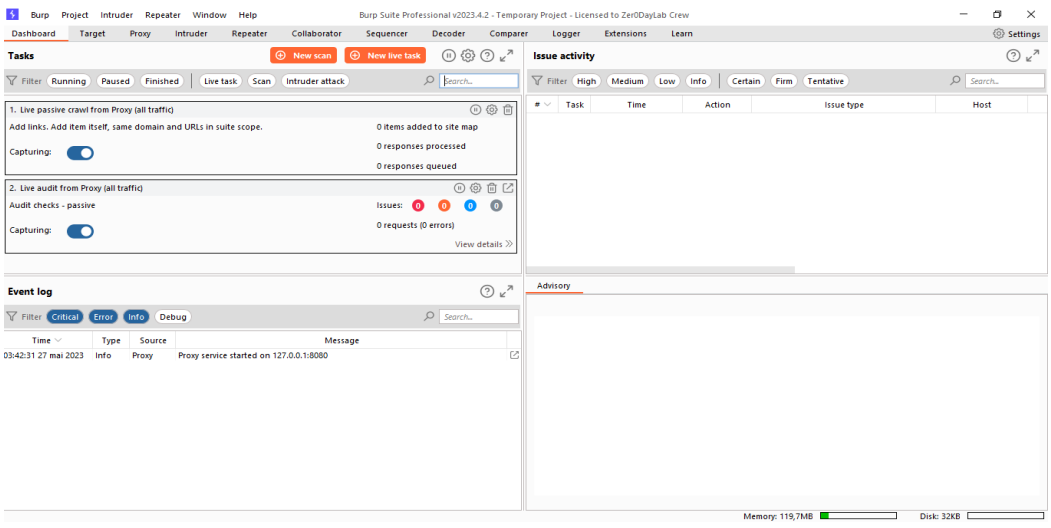


Figure 4.1: Burp Suite Tool.

The tool consists of several modules that work together seamlessly

HTTP Proxy, sits between the user's browser and the target application, allowing the user to intercept and modify requests and responses. This enables security testers to analyze and manipulate web traffic, making it easier to discover vulnerabilities.

Scanner, Automates the process of identifying common security issues such as SQL injection, cross-site scripting (XSS), and insecure direct object references. The Scanner performs active scanning by automatically sending requests to the target application and analyzing the responses for potential vulnerabilities.

Spider, crawls through a web application, systematically exploring different parts of the site and mapping out its structure. This helps in identifying hidden or forgotten pages that might contain security vulnerabilities.

Intruder, allows for the automated testing of multiple input parameters using customizable attack payloads.

Repeater, it can be used to modify requests to the server, resend them, and observe the results.

Decoder, Decoder lists the common encoding methods like URL, HTML, Base64, Hex, etc. This tool comes handy when looking for chunks of data in values of parameters or headers. It is also used for payload construction for various vulnerability classes. It is used to uncover primary cases of IDOR and session hijacking.

Extender, Burp Suite supports external components to be integrated into the tools suite to enhance its capabilities. These external components are called BApps. These work just like browser extensions. These can be viewed, modified, installed, uninstalled in the Extender window.

Sequencer, A tool for analyzing the quality of randomness in a sample of data items. It can be used to test an application's session tokens or other important data items that are intended to be unpredictable.

4.2.2. Metasploit

Metasploit, developed by Rapid7, is a renowned and robust penetration testing framework that has gained significant recognition in the field. It offers a wide range of tools, exploits, and payloads that empower security professionals and ethical hackers to assess and exploit vulnerabilities across different systems, applications, and networks. The comprehensive capabilities of Metasploit make it an indispensable resource for conducting thorough security assessments [29].

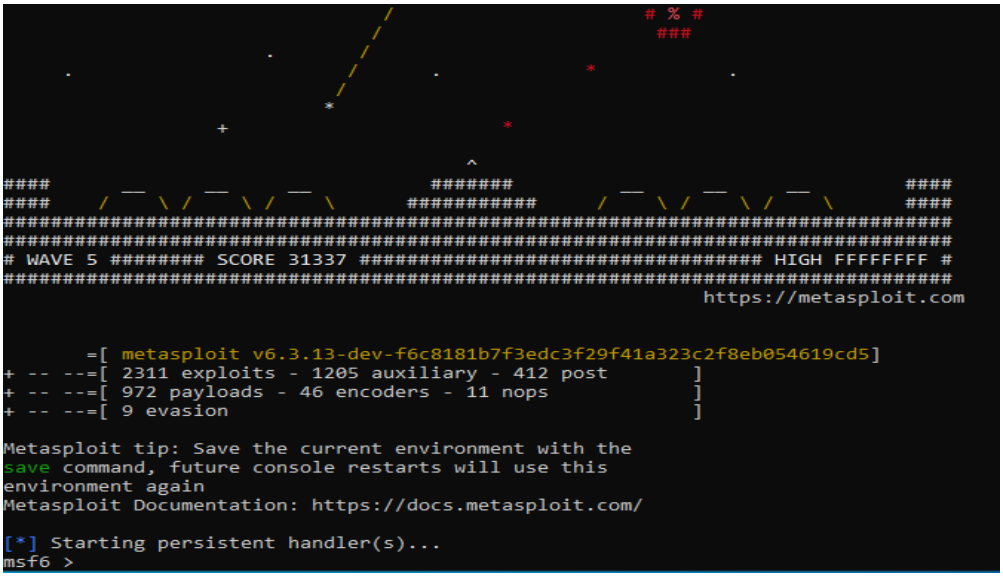


Figure 4.2: Metasploit Tool.

Msfconsole provides a convenient all-in-one interface for all the available options and settings.

You can launch an exploit, load an auxiliary module, perform enumeration, create listeners, or conduct a mass exploitation against an entire network.

Msfvenom, is a command-line tool in the Metasploit framework used to generate various types of payloads and shellcode for exploitation purposes. It enables the creation of custom payloads for different operating systems, architectures, and scenarios. With flexible options, it supports multiple output formats, payload types, encoding techniques, and payload customization. Msfvenom integrates with Metasploit components and aids in generating malicious code for penetration testing and offensive security engagements.

4.2.3. Nmap

Nmap, short for "Network Mapper," is a widely used open-source network scanning and exploration tool. It is designed to discover hosts, services, and open ports within a network, providing valuable information for network administrators, security professionals, and ethical hackers.[30]

Nmap employs a variety of scanning techniques to gather data about target systems. It can perform basic host discovery by sending ICMP echo requests (ping) or more advanced TCP/IP and ICMP scanning methods. By sending specific packets to target hosts and analyzing their responses, Nmap can determine which ports are open, the services running on those ports, and even the operating system being used.

The tool offers a range of scanning options, allowing users to customize the intensity and scope of their scans. These options include TCP SYN scan (also known as half-open scanning), TCP connect scan, UDP scanning, OS detection, service version detection, script scanning, and more. Nmap also supports advanced scanning techniques like timing and performance optimization to improve efficiency.

Besides network scanning, Nmap includes additional features such as scripting capabilities and the Nmap Scripting Engine (NSE). NSE allows users to write and execute scripts to automate tasks, perform specific checks, or gather more detailed information about target systems.

Nmap is known for its flexibility, extensibility, and cross-platform compatibility. It is available for multiple operating systems, including Windows, macOS, Linux, and BSD, making it accessible to a wide range of users.

Overall, Nmap is a powerful and versatile network scanning tool that provides insights into network topology, host discovery, open ports, and services running on target systems. Its extensive feature set and active community support make it a popular choice for network reconnaissance, security auditing, and vulnerability assessment.

Zenmap is a graphical user interface (GUI) for Nmap, a network scanning tool. It provides an intuitive interface for users to perform network scans and view the results in a visual format. Zenmap simplifies the process of using Nmap by offering scan profiles, customization options, and real-time monitoring of scans. It integrates with Nmap's advanced features, such as script scanning, and supports different output formats for saving and sharing scan results. Zenmap is ideal for users who prefer a graphical interface for network scanning tasks [30].

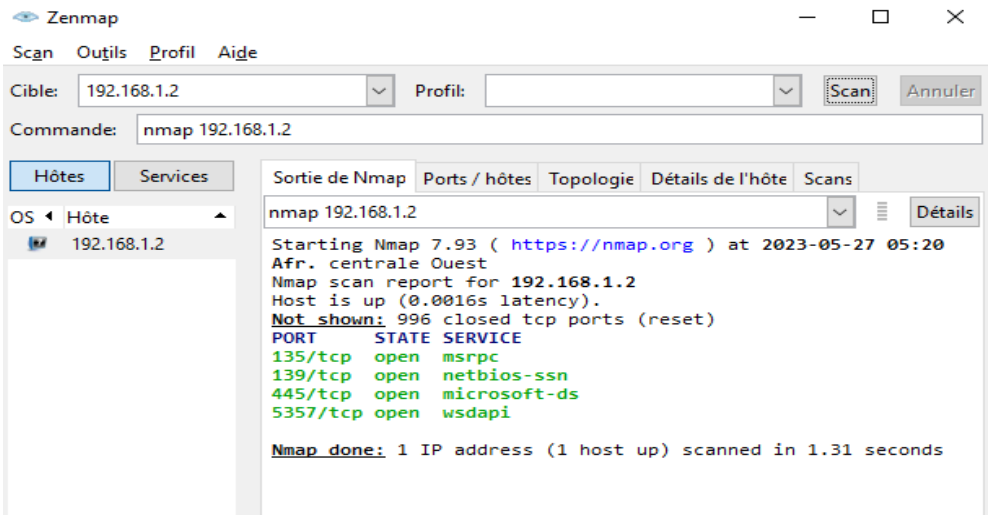


Figure 4.3: Zenmap Tool.

4.3. Penetration test

A comprehensive penetration test was performed on a company's web application with the objective of validating its adherence to OWASP security standards and identifying as many vulnerabilities as possible.

All the sensitive information about the company, their software and their clients have been obfuscated in order to avoid sensitive data exposure and privacy problems.

4.3.1. Information Gathering

The tested web application, it is a web application dedicated to sales of medical and surgical equipment. It is hosted on a server reachable.

We have been provided with information that the web application is hosted on a Linux server (specifically, Ubuntu 20.04 64-bit). Furthermore, the website is built using the WordPress framework, specifically version 6.0.0.

4.3.2. Automated scan

Using Zenmap, when we used Zenmap for scanning, we discovered that four ports are open. However, we are only interested in two specific ports: port 80 and port 8080.

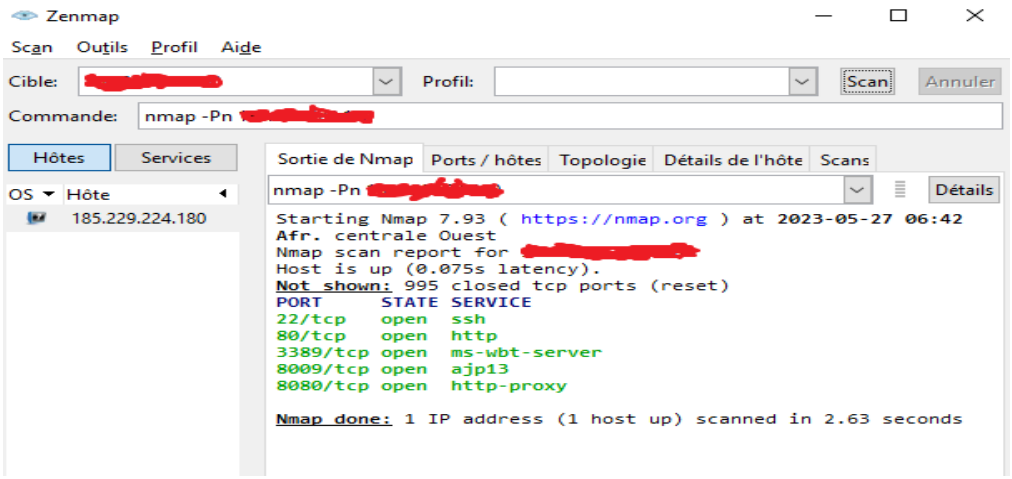


Figure 4.4: Scanning with Zenmap Tool.

Port 80 is commonly used for Hypertext Transfer Protocol (HTTP) communication. HTTP is the protocol that allows communication between web servers and web browsers. When you type a website address in your browser, the browser sends an

HTTP request to the web server on port 80, and the server responds with the requested web page.

Port 8080, on the other hand, is often used as an alternative port for HTTP communication. It is commonly used for web servers that run on a non-standard port, such as development servers or proxy servers.

Using Burp Suite, when we used Burp Suite tool, we didn't discover any high issues or vulnerabilities.

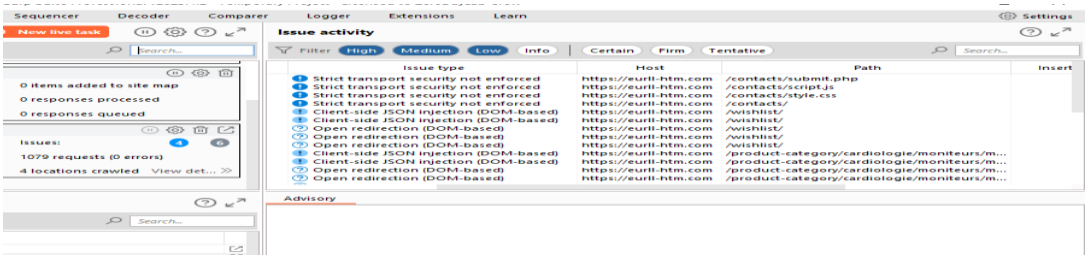


Figure 4.5: Scanning with Burp Suite Tool.

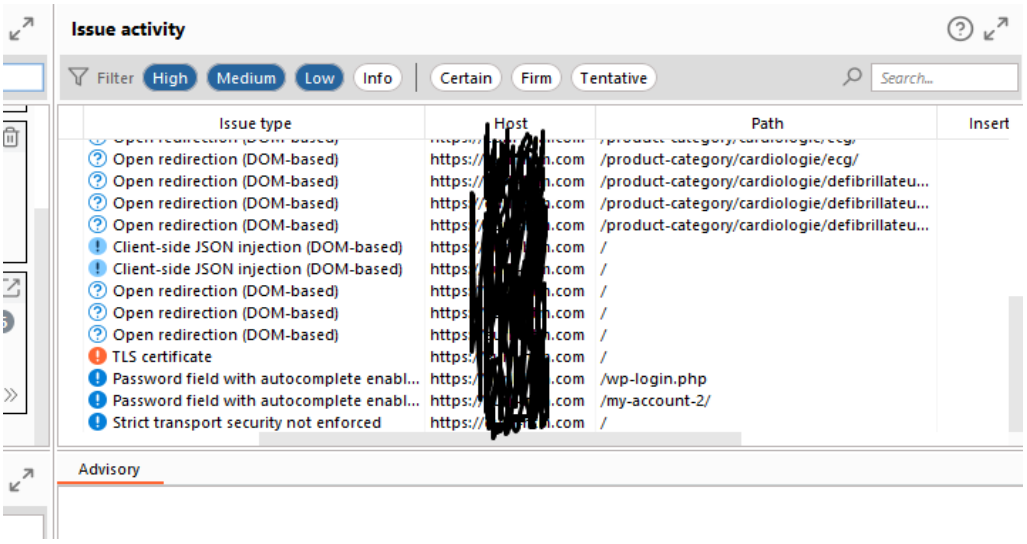
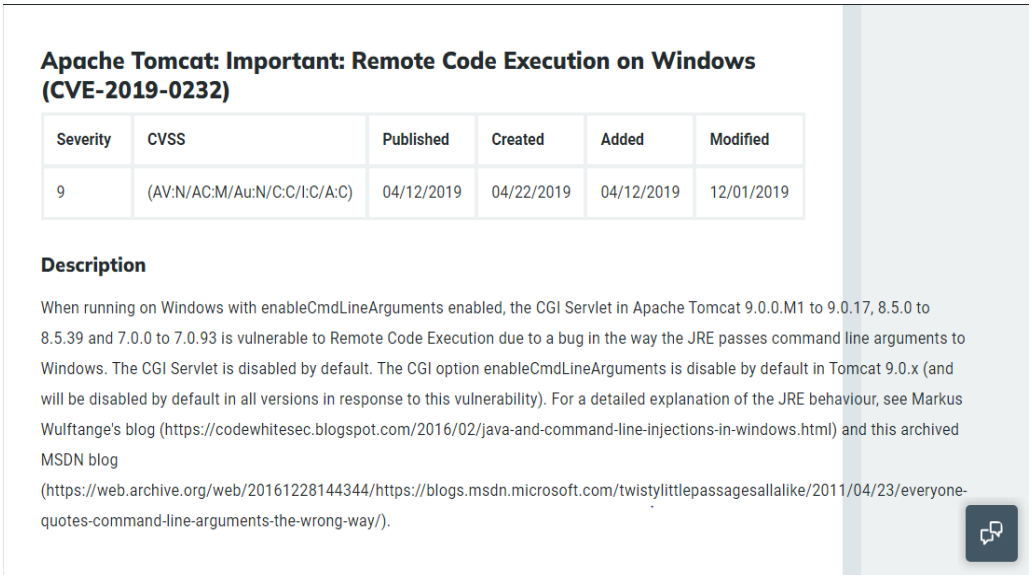


Figure 4.6: Scanning with Burp Suite Tool.

4.3.3. Tomcat vulnerability

When we visit https://Companywebsite.com:8080, a Tomcat page opens up, displaying information about the Tomcat server. We can see that the version is 9.0.0.M1. Knowing the Tomcat version allowed us to discover that this version is vulnerable to CVE-2019-0232, and CVE-2017-12617.

However, CVE-2019-0232 vulnerability can only be exploited if Tomcat is running on a Windows operating system and if the CGI Servlet is enabled. Since the website is running on a Linux server, we conclude that we cannot exploit this vulnerability.



Apache Tomcat: Important: Remote Code Execution on Windows (CVE-2019-0232)

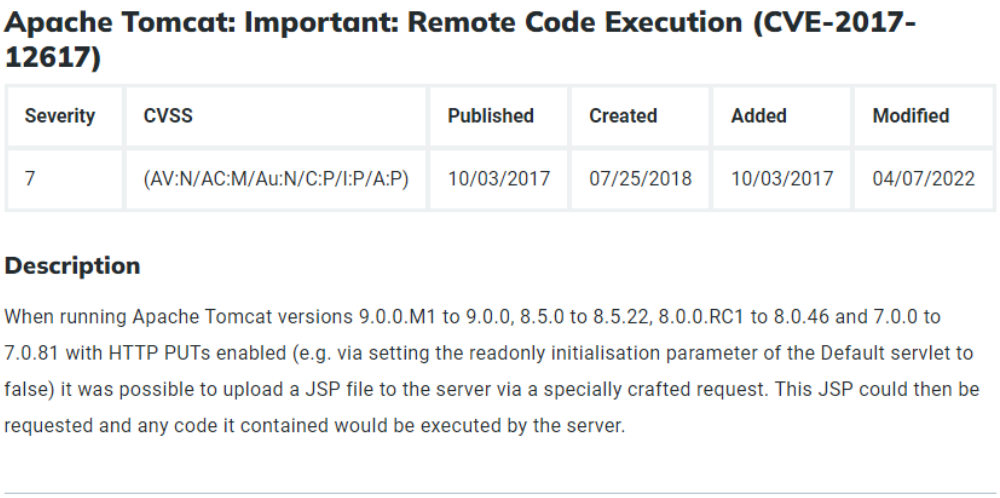
Severity	CVSS	Published	Created	Added	Modified
9	(AV:N/AC:M/Au:N/C:C/I:C/A:C)	04/12/2019	04/22/2019	04/12/2019	12/01/2019

Description

When running on Windows with enableCmdLineArguments enabled, the CGI Servlet in Apache Tomcat 9.0.0.M1 to 9.0.17, 8.5.0 to 8.5.39 and 7.0.0 to 7.0.93 is vulnerable to Remote Code Execution due to a bug in the way the JRE passes command line arguments to Windows. The CGI Servlet is disabled by default. The CGI option enableCmdLineArguments is disable by default in Tomcat 9.0.x (and will be disabled by default in all versions in response to this vulnerability). For a detailed explanation of the JRE behaviour, see Markus Wulftange's blog (https://codewhitesec.blogspot.com/2016/02/java-and-command-line-injections-in-windows.html) and this archived MSDN blog (https://web.archive.org/web/20161228144344/https://blogs.msdn.microsoft.com/twistylittlepassagesallalike/2011/04/23/everyone-quotes-command-line-arguments-the-wrong-way/).

Figure 4.7: Apache Tomcat 9.0.0.M1 Vulnerability CVE-2019-0232 [31].

But, CVE-2017-12617 can be exploited because when running Apache Tomcat versions 9.0.0.M1 via setting the readonly initialisation parameter of the Default servlet to false, it was possible to upload a JSP file to the server via a specially crafted request. This JSP could then be requested and any code it contained would be executed by the server.



Apache Tomcat: Important: Remote Code Execution (CVE-2017-12617)

Severity	CVSS	Published	Created	Added	Modified
7	(AV:N/AC:M/Au:N/C:P/I:P/A:P)	10/03/2017	07/25/2018	10/03/2017	04/07/2022

Description

When running Apache Tomcat versions 9.0.0.M1 to 9.0.0, 8.5.0 to 8.5.22, 8.0.0.RC1 to 8.0.46 and 7.0.0 to 7.0.81 with HTTP PUTs enabled (e.g. via setting the readonly initialisation parameter of the Default servlet to false) it was possible to upload a JSP file to the server via a specially crafted request. This JSP could then be requested and any code it contained would be executed by the server.

Figure 4.8: Apache Tomcat 9.0.0.M1 Vulnerability CVE-2017-12617 [32].

4.3.4. File Upload vulnerability

File upload vulnerability refers to a security weakness in a web application that allows attackers to upload malicious files. This can lead to various risks, such as executing harmful code on the server, disclosing sensitive information, or causing

denial of service. To mitigate this vulnerability, web applications should validate file types, limit file sizes, store files securely, and implement malware scanning. Regular security testing and keeping software up to date are important for identifying and addressing file upload vulnerabilities.[33]

During our exploration of the website, we discovered a contact page. When accessing Companywebsite.com, you will find a button labeled "Contactez-nous si vous souhaitez effectuer un achat en gros". Clicking this button redirects you to the page Companywebsite.com/contacts. On this page, there is a form that can be filled out and submitted. It allows for file attachments.

The attached files are stored on the website at Companywebsite.com /contacts/uploads/. However, the website only permits the attachment of JPG, PNG, and PDF files. Fortunately, we were able to bypass this restriction using the Burp Suite tool. So, this page has a file upload vulnerability that can be potentially exploited.

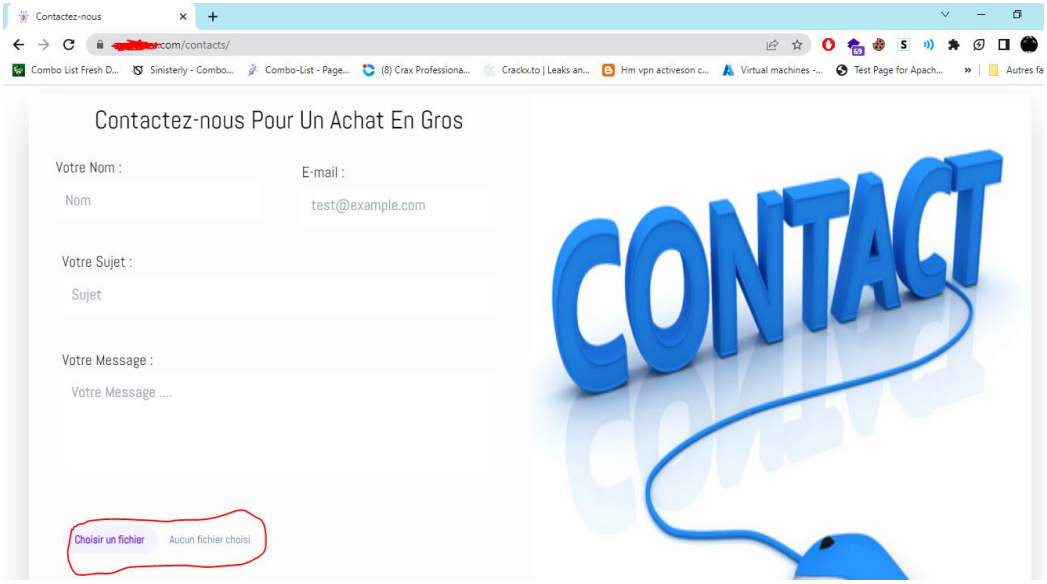


Figure 4.9: File Upload vulnerability.

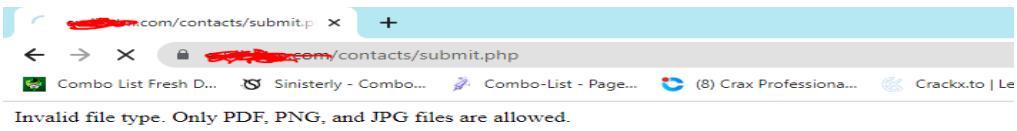


Figure 4.10: The website only permits the attachment of (JPG, PNG, PDF) files.

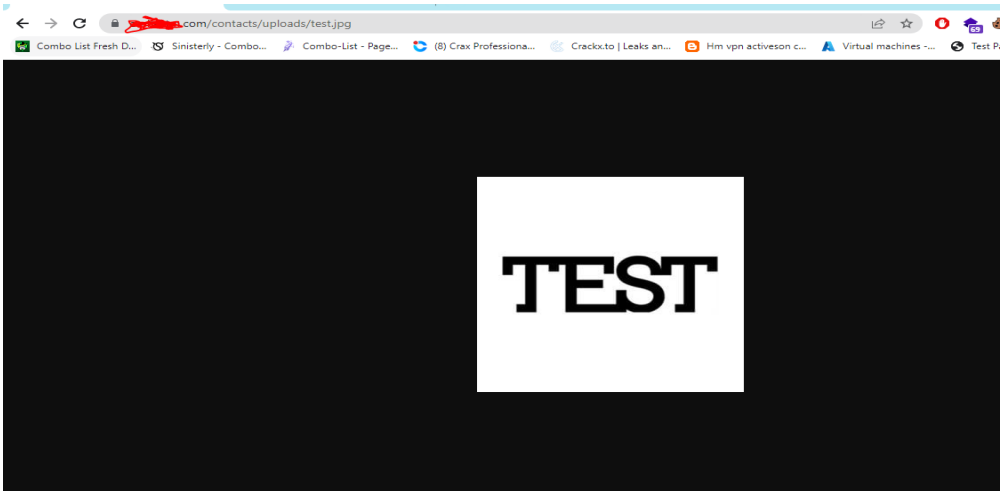


Figure 4.11: The attached files are stored on the website.

4.3.5. Default Credentials

Default credentials are pre-configured usernames and passwords set by manufacturers or developers for initial access to systems or devices. If left unchanged, they pose a security risk as attackers can exploit them for unauthorized access [34].

In our case as we mentioned in Tomcat vulnerability section, we found that we were able to access the Tomcat page, and knowing that default credentials for tomcat are typically set as username = admin and password = password. When we attempted to login using these credentials, we successfully gained access. This vulnerability allows us to exploit the system with remote code execution, granting us unauthorized control over the system.

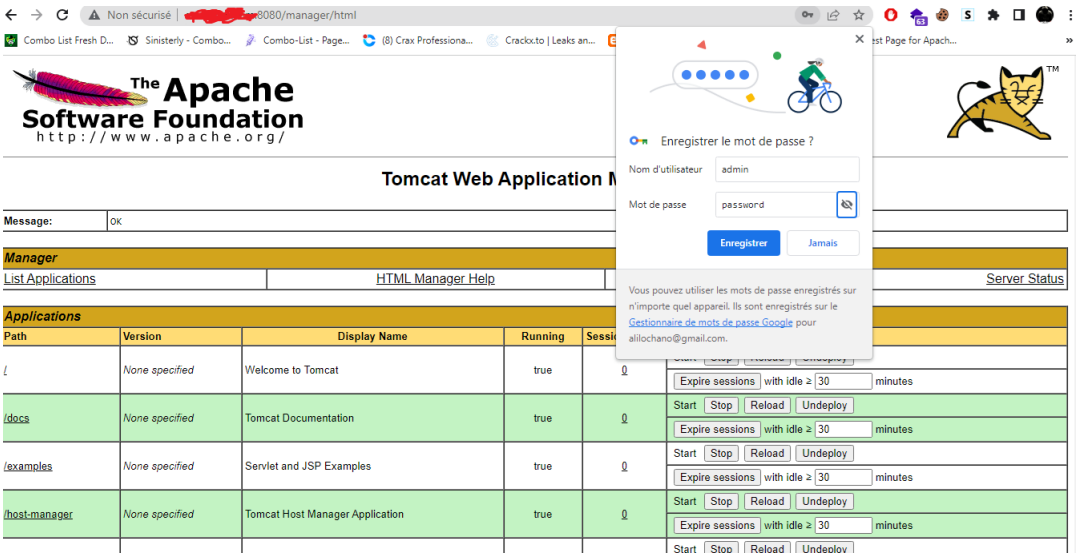


Figure 4.12: Login to Apache Tomcat manager with default credentials.

4.3.6. Guessable account

A guessable account refers to an account with easily predictable or easily guessed credentials. This can include weak passwords, common usernames, or readily available account information. Guessable accounts pose a significant security risk as attackers can easily gain unauthorized access by guessing the credentials or using automated techniques.

Knowing that the website is built using the WordPress framework, we attempted a brute-force attack by using a list of easily guessable usernames and passwords. We employed a Perl script to automate the brute-force attack on the WordPress login page, using the list we created.

Upon executing the script, we successfully discovered the login credentials for an admin account. The username and password were identified as highly guessable and vulnerable.

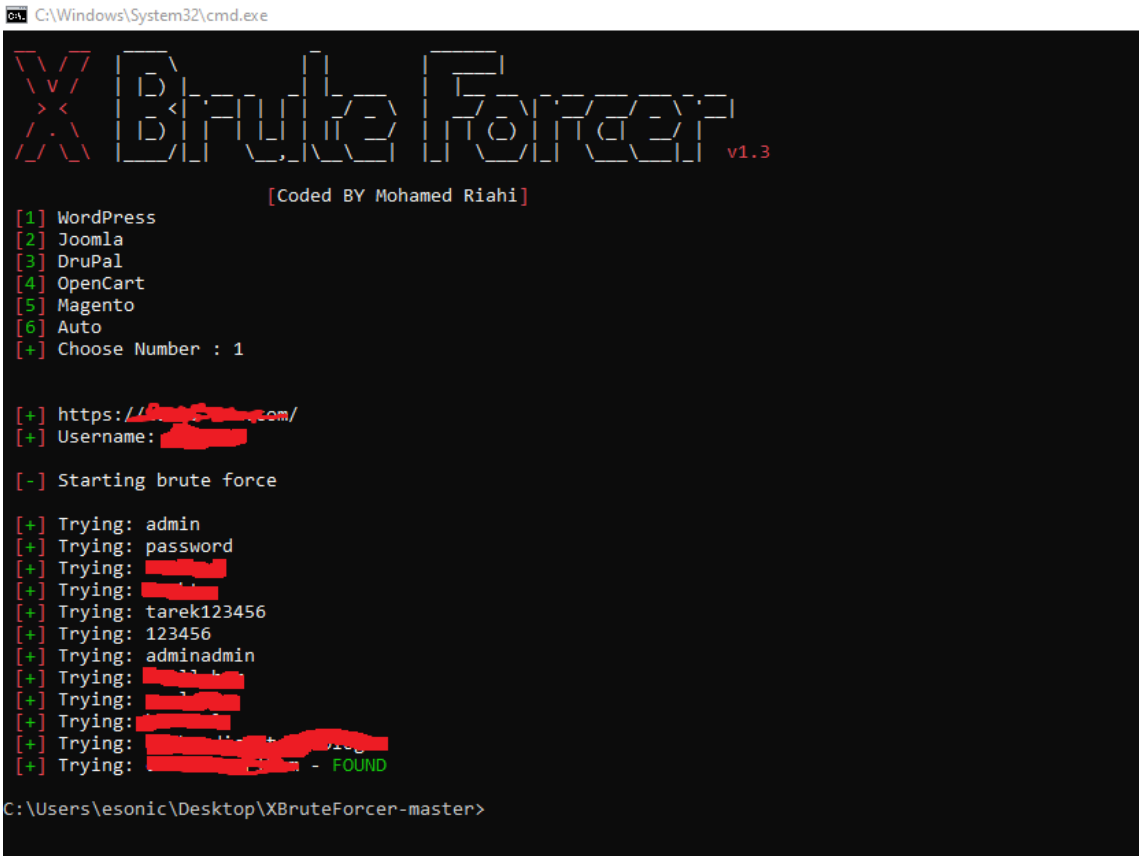


Figure 4.13: WordPress Brute-force attack.

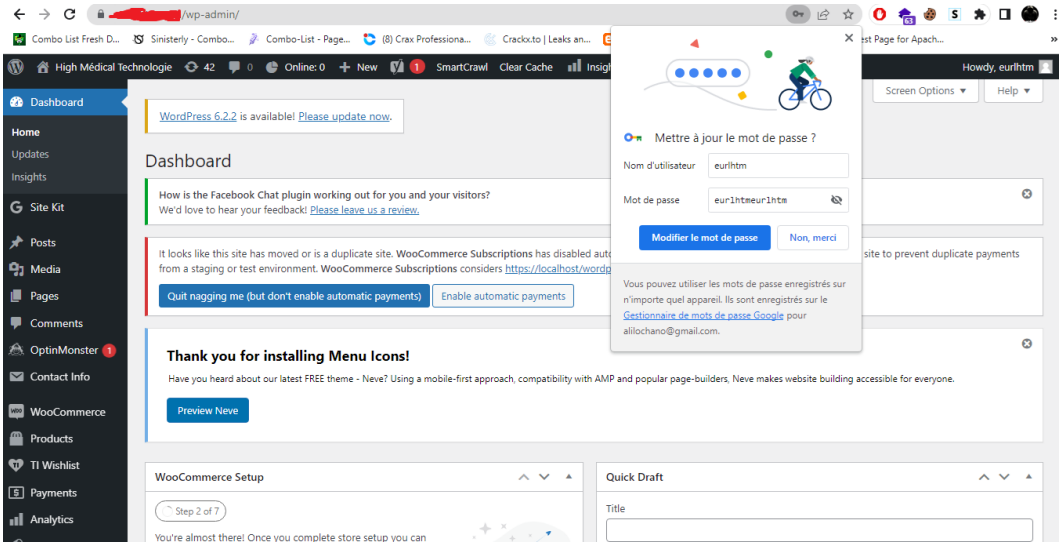


Figure 4.14: Successful Login to Wp admin.

4.3.7. XSS vulnerability

On the webpage Companywebsite.com/contacts, there is a form available for users to enter and submit their information. Upon investigation, we discovered a concerning vulnerability. The message box within the form lacks proper protection against special characters such as (<,>). This poses a potential risk for cross-site scripting (XSS) attacks. Interestingly, once the admin accesses their panel, they can view all the submitted information using a plugin. Consequently, it appears that the vulnerability at hand is specifically categorized as a stored XSS vulnerability.

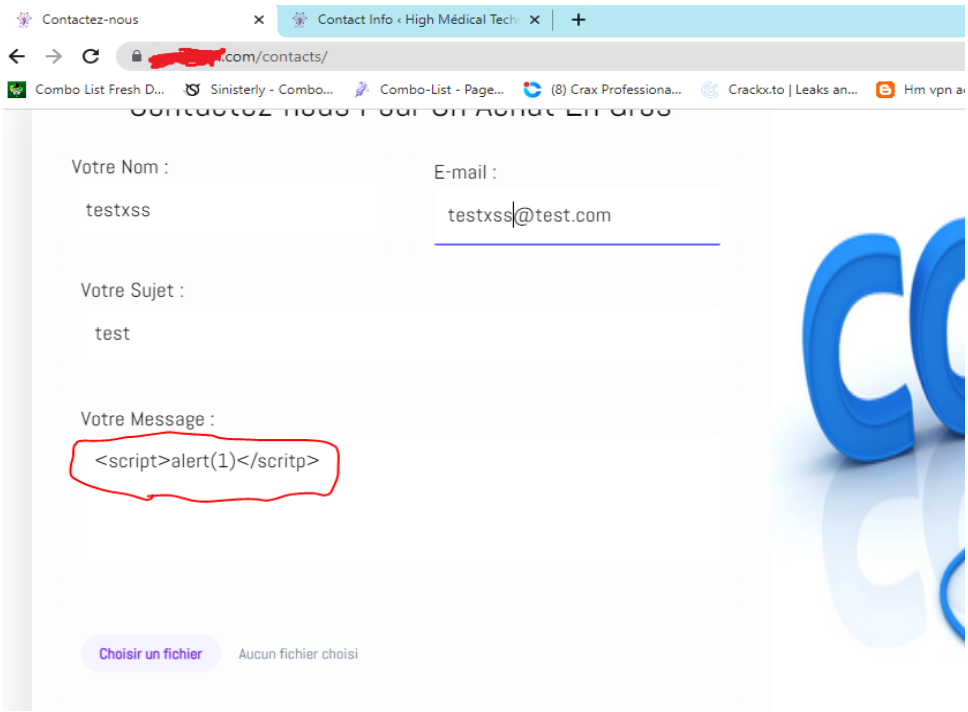


Figure 4.15: Manual XSS test.

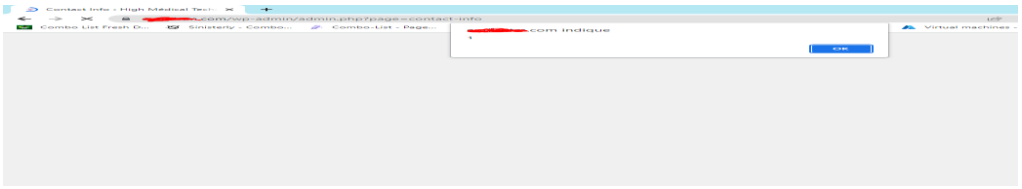


Figure 4.16: Script worked.

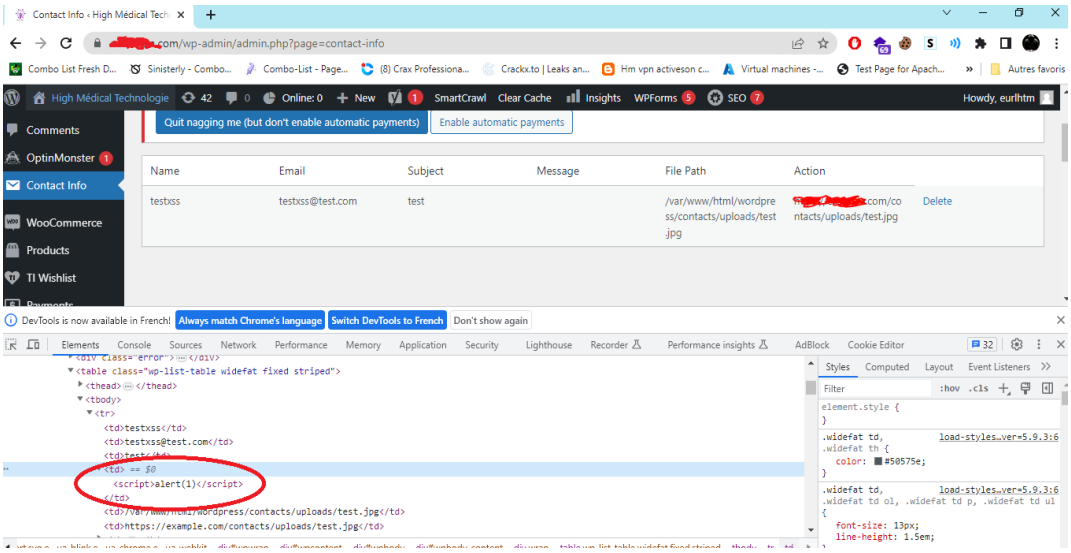


Figure 4.17: Script injected in the admin panel.

4.3.8. Elementor plugin vulnerability

Upon logging into the admin panel, we made a concerning discovery. It appears that an older version of the Elementor plugin (3.6.2) is being utilized, which contains a vulnerability that can potentially be exploited. This particular vulnerability poses a risk of remote code execution (RCE).

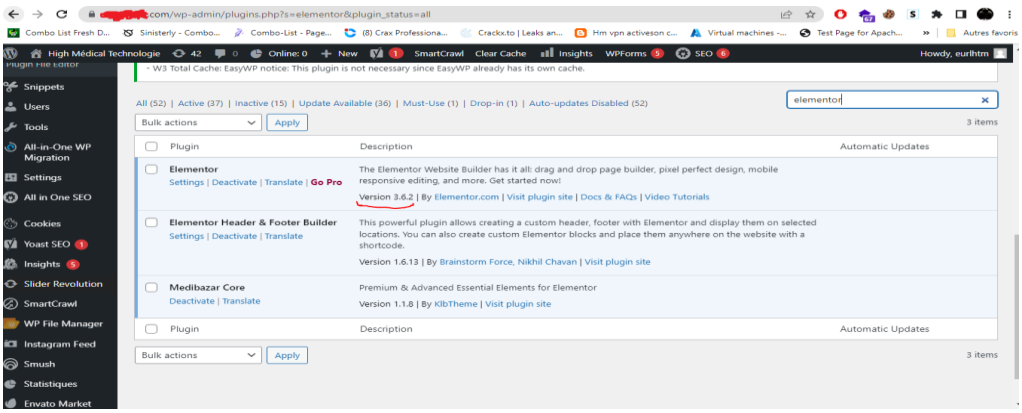


Figure 4.19: Elementor 3.6.2 plugin installed on the website.

4.4. Exploit vulnerabilities

4.4.1. Exploit Tomcat vulnerability CVE-2017-12617

This vulnerability can be exploited by sending a specially crafted HTTP PUT request with a JSP payload to a Tomcat server. The code within the uploaded JSP file is executed when it is accessed through a web browser.

There is an exploit module available in the Metasploit tool that automates these steps. By modifying the RHOST attribute to the target website (Companywebsite.com), the exploit module can be used effectively.

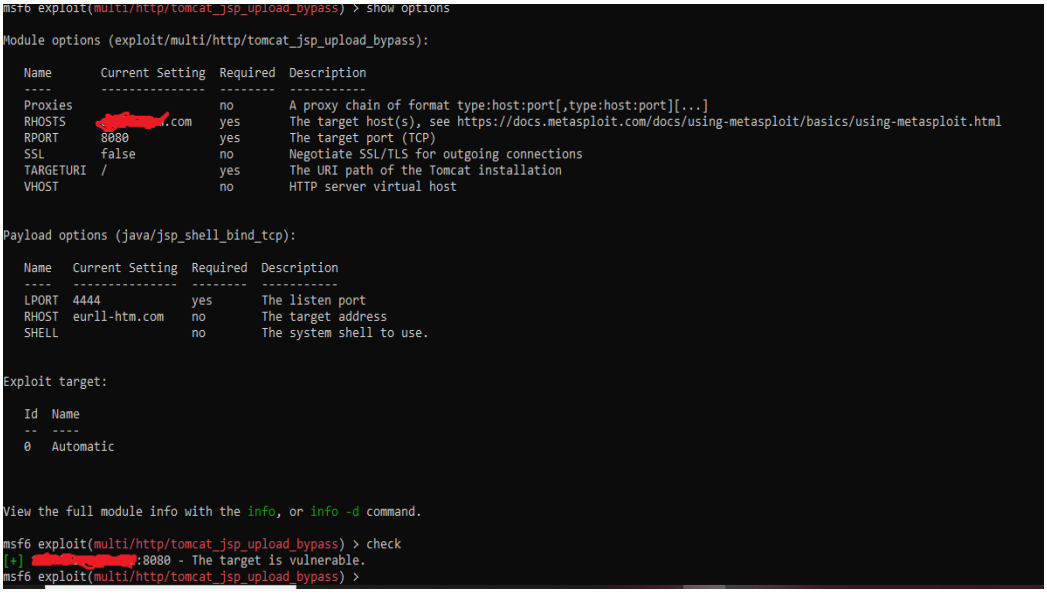


Figure 4.20: Check the target if is vulnerable with Metasploit.

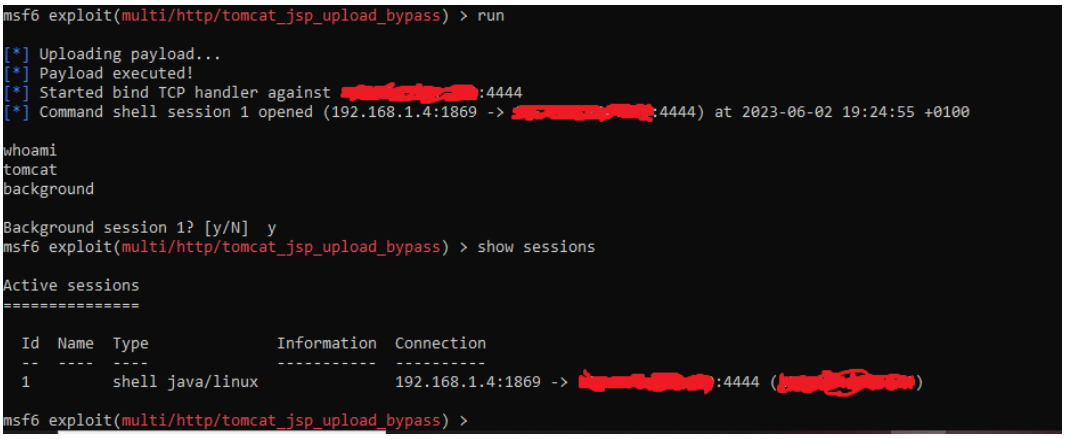


Figure 4.21: Exploit the vulnerable target with Metasploit.

4.4.2. Exploit Elementor plugin vulnerability

We will run a Python script that exploits this vulnerability. This script will upload a web shell, allowing us to access it at (Companywebsite.com /index.php?activate=1).

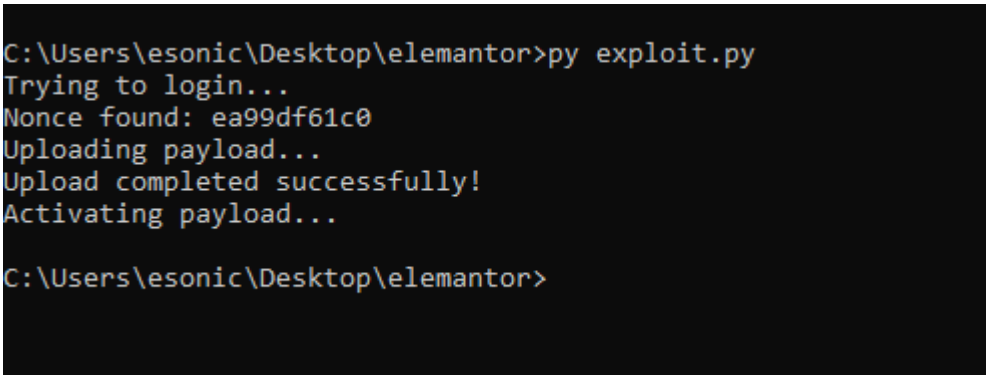


Figure 4.22: Exploit the Elementor plugin vulnerability.

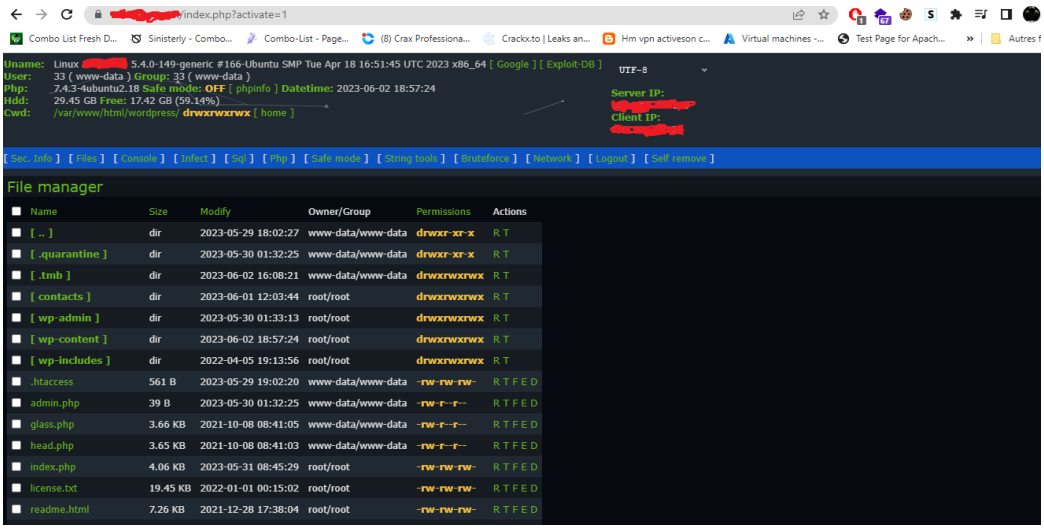


Figure 4.23: Navigate to the web shell.

4.4.3. Exploit XSS vulnerability

Knowing that there is an XSS vulnerability on the contacts page, specifically in the message box, we will submit a script that adds an admin user to the WordPress panel. This script will execute when the admin views the submitted information.

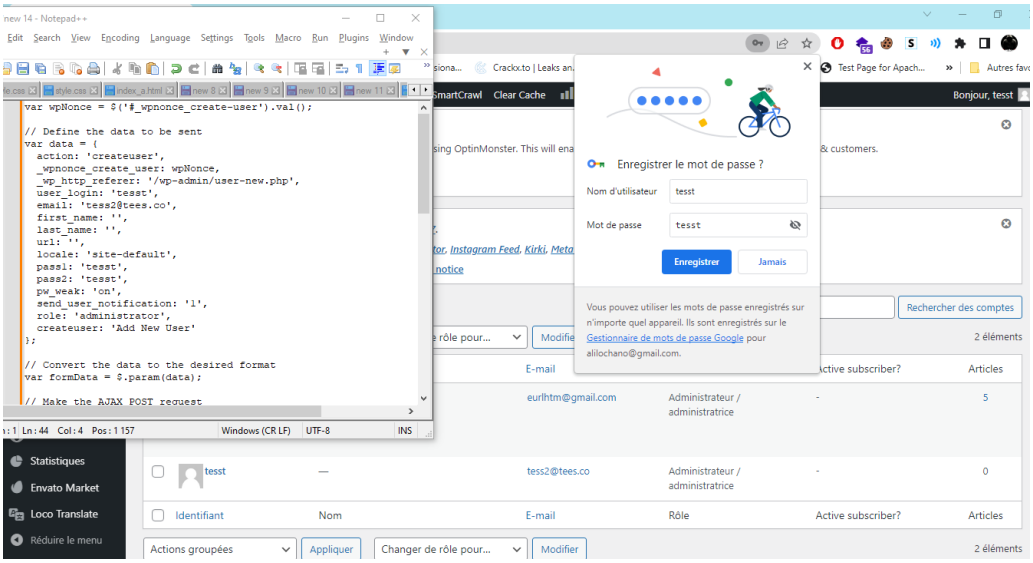


Figure 4.24: XSS attack (inject script to add admin account).

4.4.2. Exploit File upload vulnerability

As mentioned before, we have the ability to upload files on the page (Companywebsite.com/contacts). The attached files are stored on the website at Companywebsite.com/contacts/uploads/. However, the website restricts attachments to JPG, PNG, and PDF files only. Thankfully, we managed to overcome this limitation by utilizing the Burp Suite tool and successfully uploaded a web shell.

We will intercept the request with Burp Suite and modify the name of the webshell file from wso.php.jpg to wso.php

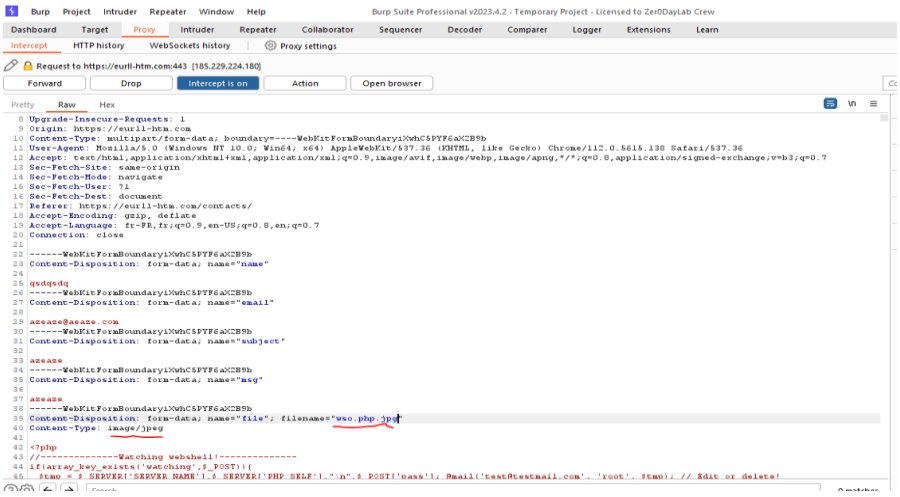


Figure 4.25: Intercept the request with Burp Suite.

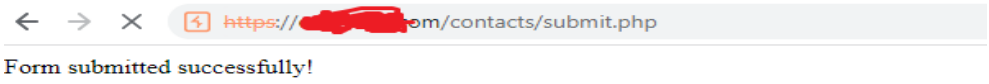


Figure 4.26: The web shell has been successfully uploaded.

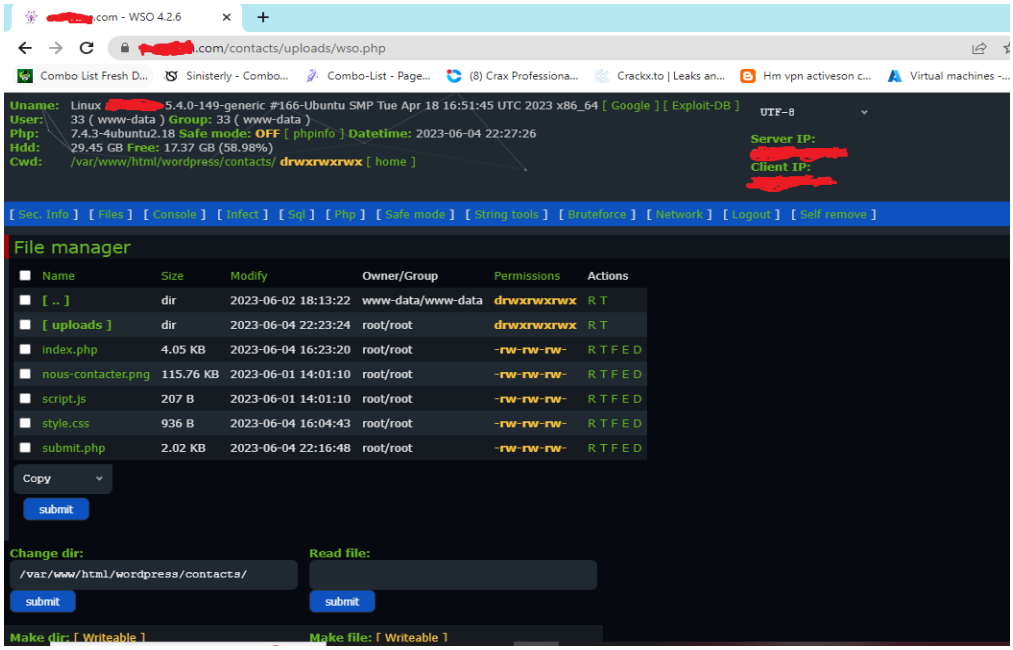


Figure 4.27: Navigate to the web shell.

4.4.2. Tomcat Default Credentials leads to RCE

We will demonstrate how default credentials for the Tomcat manager can lead to remote code execution. Firstly, we will create a payload with a .war extension using msfvenom.

Then, we will log in to the Tomcat manager and upload the payload, evading the system by disguising the file as a configuration file, which will be accepted by Tomcat. Once the payload is successfully uploaded, we can execute the payload it by navigating to Companywebsite.com:8080/tomcat/payloadname.jsp.

Note that to obtain the payload name, you can open the .war file using WinRAR, copy the name, and then paste it into the browser.

We will utilize the Metasploit multi-handler module to establish a reverse shell connection upon the execution of the payload.

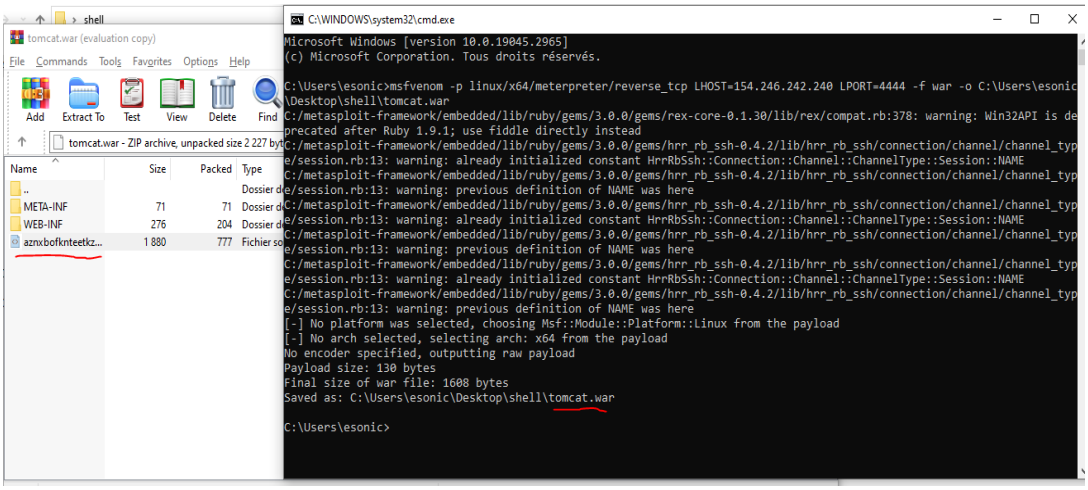


Figure 4.28: Creating Payload using msfvenom.

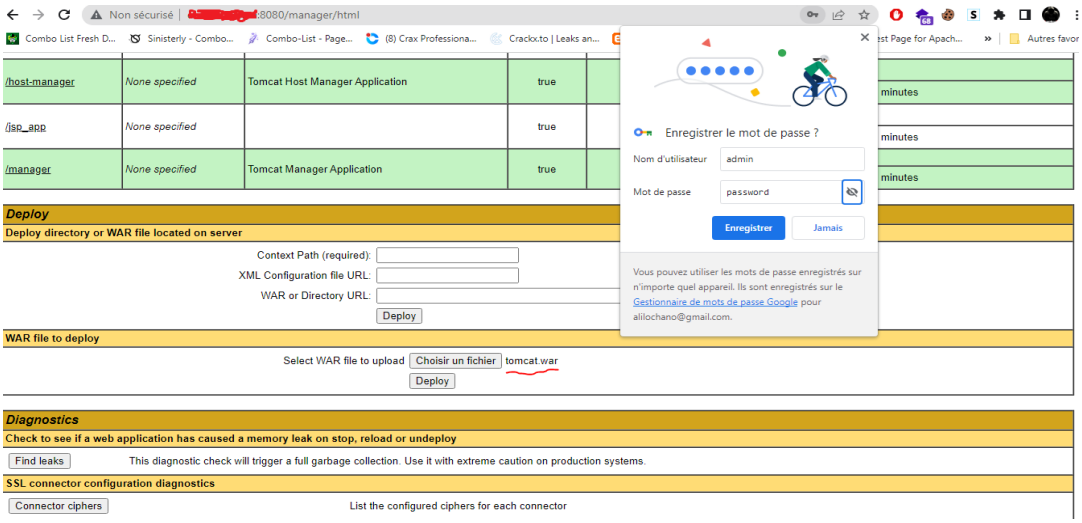


Figure 4.29: Uploading the payload to Tomcat manager.

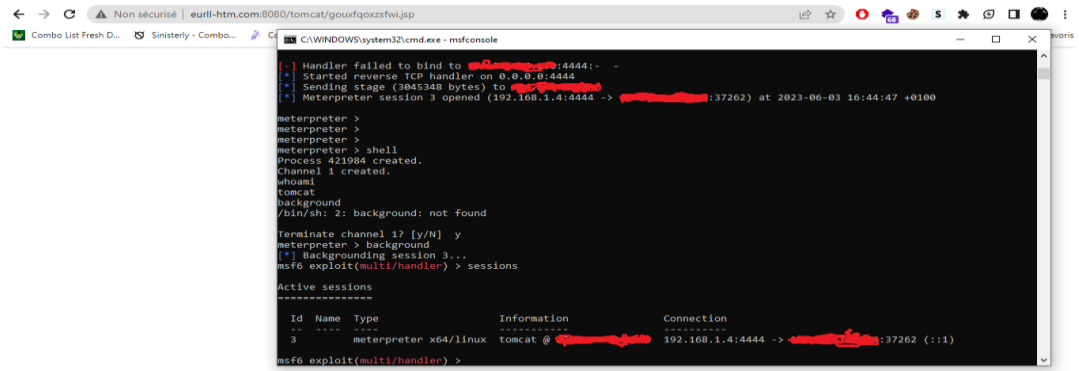


Figure 4.30: Receiving a reverse shell after executing the payload.

4.5. Results summary

All the vulnerabilities found during the penetration testing have been grouped in Table 4.1.

Scope of the test	High risk	Moderate risk	Low risk	OWASP
Tomcat vulnerability (CVE-2017-12617)	**X**			Not Consistent
Tomcat vulnerability (CVE-2019-0232)			**X**	Not Consistent
Tomcat Default Credentials	**X**			Consistent
File upload vulnerability	**X**			Not Consistent
XSS vulnerability	**X**			Consistent
Elementor plugin vulnerability	**X**			Not Consistent
Guessable account	**X**			Consistent

Table 4.1: vulnerabilities found in our penetration test.

4.5.1. How to prevent discovered security vulnerabilities

The first vulnerability we will discuss is the Tomcat vulnerability (CVE-2017-12617). To prevent this vulnerability, you simply need to upgrade the Tomcat version. Another option is to set the readonly initialization parameter of the Default servlet to True.

The next vulnerability is related to the Elementor plugin. To prevent this vulnerability, you should upgrade the Elementor plugin to the latest version. Another vulnerability is the File upload vulnerability. To prevent this, you need to modify the functions that verify the file type allowed for upload. Instead of verifying the content

type of the file, the function should verify the file extension. Additionally, it is advisable to add an additional security verification after the file has been sent to the server.

The following vulnerability is an XSS vulnerability. To prevent this vulnerability, it is necessary to validate inputs on the page (Companywebsite.com/contacts). For example, you can add a filter to deny special characters from being sent to the server.

Lastly, there is the Default Credentials and Guessable account vulnerability. To prevent this, you should edit passwords to be more complex and not easily guessable.

General Conclusion

The main objective of the dissertation was to extensively examine the current state of web application security. We explored the wide range of vulnerabilities and security flaws that exist in internet-connected applications. It is crucial for developers and testers to have a reliable point of reference that they can use to create and certify secure web applications. Currently, the most trusted standard in the field is OWASP (Open Web Application Security Project), which provides guidelines, suggestions, and tools to ensure web application security.

One of OWASP's significant contributions is the OWASP Top 10, which lists the ten most common and dangerous vulnerabilities found in web applications. It offers detailed descriptions and explains how these vulnerabilities can be exploited. However, it's important to note that new vulnerabilities emerge regularly, and even small flaws can have significant consequences for a company. To address this, OWASP has developed a valuable penetration testing technique that covers various types of vulnerabilities. Developers and testers should adopt this methodology and customize it as needed for specific applications. This approach enhances security and effectively reduces risks.

During the case study conducted as part of the dissertation, we found that many vulnerabilities stemmed from inadequate input validation mechanisms (XSS Vulnerability) and a lack of measures to prevent the upload of malicious files (File Upload Vulnerability). We also discovered that even minor oversights, such as using outdated software versions (Tomcat and Elementor plugin outdated version) or weak passwords (Tomcat manager password and Wordpress login password), can lead to severe consequences.

Among the problems we encountered, one significant issue was the lack of access to the main server. If we had been granted access, we could have identified additional vulnerabilities, particularly in terms of privilege escalation.

Our ambitious future goal is to leverage cutting-edge AI technologies to develop a robust and comprehensive web vulnerability scanner application. This groundbreaking solution will revolutionize security processes for companies, streamlining and enhancing their overall efficiency. By harnessing the power of AI, we aim to provide an advanced and intuitive tool that identifies and mitigates potential vulnerabilities in web applications, ensuring unparalleled protection against cyber threats. Our vision is to

empower companies with an all-encompassing security solution that not only saves time and resources but also fortifies their digital infrastructure against ever-evolving security risks.

Unfortunately, many companies still overlook software security as an essential part of the software development life cycle. They tend to view cybersecurity as a cost rather than a worthwhile investment.

BIBLIOGRAPHY

[1] National Institute of Standards and Technology (NIST). "Penetration Testing." NIST Glossary. Available at:

https://csrc.nist.gov/glossary/term/penetration_testing. Consulted on 13/02/2023.

[2] Nidhra, Srinivas, and Jagruthi Dondeti. "Black box and white box testing techniques-a literature review." International Journal of Embedded Systems and Applications (IJESA) 2.2 (2012): 29-50.

[3] Redscan. "Types of Pen Testing: White Box, Black Box and Everything in Between." Available at:

https://www.redscan.com/news/types-of-pen-testing-white-box-black-box-and-everything-in-between. Consulted on 14/02/2023.

[4] JavaTpoint. "Advantages and Disadvantages of White Box Testing." Available at: https://www.javatpoint.com/advantages-and-disadvantages-of-white-box-testing. Consulted on 14/02/2023.

[5] Check Point. "What is Gray Box Testing?" Check Point Cyber Security Hub. Available at:

https://www.checkpoint.com/cyber-hub/cyber-security/what-is-gray-box-testing. Consulted on 14/02/2023.

[6] Simplilearn. "White Hat Hacker: All You Need to Know About Ethical Hacking." Available at:

https://www.simplilearn.com/white-hat-hacker-article. Consulted on 18/02/2023.

[7] Kaspersky. "Hacker Hat Types." Kaspersky Resource Center. Available at: https://www.kaspersky.com/resource-center/definitions/hacker-hat-types. Consulted on 18/02/2023.

[8] Varonis. "Red Team vs. Blue Team: How They Battle Hackers." Available at: https://www.varonis.com/blog/red-team-vs-blue-team/. Consulted on 18/02/2023.

[9] SQA Consulting. "InfoSec Colour Team Structure: The Purple Team." Available at: https://sqa-consulting.com/infosec-colour-team-structure-the-purple-team/. Consulted on 18/02/2023.

[10] S. Jajodia, P. Liu, and V. Swarup, "A unified ontology for security and resiliency of cyber-physical systems," Journal of Cybersecurity, vol. 6, no. 1, 2020.

[11] Duraiswamy, K., and Mrs R. Uma Rani MCA. "Security through obscurity." KSR College Of Technology, Tiruchengode (2005).

[12] OWASP. "Input Validation Cheat Sheet." OWASP Cheat Sheet Series. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Input_Validation_Cheat_Sheet.html. Consulted on 23/02/2023.

[13] Mozilla Developer Network (MDN). "Same-origin policy." Available at: https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy. Consulted on 23/02/2023.

[14] Open Web Application Security Project (OWASP). "OWASP Risk Rating Methodology." Available at:

https://owasp.org/www-community/OWASP_Risk_Rating_Methodology. Consulted on 27/02/2023.

[15] OWASP. "OWASP Risk Rating Methodology." OWASP Wiki. Available at: https://wiki.owasp.org/index.php/OWASP_Risk_Rating_Methodology. Consulted on 27/02/2023.

[16] Open Web Application Security Project (OWASP). "Injection." OWASP Top Ten Project. Available at:

https://owasp.org/www-project-top-ten/2017/A1_2017-Injection.html. Consulted on 08/03/2023.

[17] Open Web Application Security Project (OWASP). "Broken Authentication." OWASP Top Ten Project. Available at :

https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication. Consulted on 08/03/2023.

[18] Open Web Application Security Project (OWASP). "Sensitive Data Exposure." OWASP Top Ten Project. Available at:

https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure. Consulted on 10/03/2023.

[19] Wallarm. "What is MITM (Man-in-the-Middle) Attack?" Available at: https://www.wallarm.com/what/what-is-mitm-man-in-the-middle-attack. Consulted on 08/03/2023.

[20] Open Web Application Security Project (OWASP). "XML External Entities (XXE)." OWASP Top Ten Project. Available at:

https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE). Consulted on 11/03/2023.

[21] Cobalt. "How to Execute an XML External Entity (XXE) Injection." Cobalt Blog. Available at:
https://www.cobalt.io/blog/how-to-execute-an-xml-external-entity-injection-xxe. Consulted on 11/03/2023.

[22] Open Web Application Security Project (OWASP). "Broken Access Control." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control. Consulted on 14/03/2023.

[23] Open Web Application Security Project (OWASP). "Security Misconfiguration." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration. Consulted on 16/03/2023.

[24] Open Web Application Security Project (OWASP). "Cross-Site Scripting (XSS)." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS). Consulted on 20/03/2023.

[25] Open Web Application Security Project (OWASP). "Insecure Deserialization." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization. Consulted on 24/03/2023.

[26] Open Web Application Security Project (OWASP). "Using Components with Known Vulnerabilities." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities. Consulted on 26/03/2023.

[27] Open Web Application Security Project (OWASP). "Insufficient Logging & Monitoring." OWASP Top Ten Project. Available at:
https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%26Monitoring. Consulted on 02/04/2023.

[28] PortSwigger. "Burp Suite." Available at:
https://portswigger.net/burp. Consulted on 14/04/2023.

[29] Rapid7. "Metasploit." Available at:
https://www.metasploit.com/. Consulted on 18/04/2023.

[30] Nmap. "Nmap - Free Security Scanner For Network Exploration & Security Audits." Available at: https://nmap.org/. Consulted on 20/04/2023.

[31] Rapid7. "Apache Tomcat CVE-2019-0232." Rapid7 Vulnerability & Exploit Database. Available at:

https://www.rapid7.com/db/vulnerabilities/apache-tomcat-cve-2019-0232/. Consulted on 22/04/2023.

[32] Rapid7. "Apache Tomcat CVE-2017-12617." Rapid7 Vulnerability & Exploit Database. Available at:

https://www.rapid7.com/db/vulnerabilities/apache-tomcat-cve-2017-12617/. Consulted on 22/04/2023.

[33] PortSwigger. "File Upload." PortSwigger Web Security Academy. Available at: https://portswigger.net/web-security/file-upload. Consulted on 01/05/2023.

[34] MITRE. "CWE-1392: Incorrect Access Control." Common Weakness Enumeration. Available at:

https://cwe.mitre.org/data/definitions/1392.html. Consulted on 05/05/2023.

الملخص: في المشهد الرقمي اليوم ، تواجه الشركات التي لديها تطبيقات ويب تحديًا مستمرًا للدفاع ضد الهجمات الإلكترونية. تلعب الإجراءات الاستباقية مثل اختبار الاختراق دورًا مهمًا في تحديد نقاط الضعف قبل أن يتم استغلالها. قمنا ب تسلطنا الضوء على أهمية اختبار الاختراق ، وتمكين الشركات من اتخاذ قرارات مستنيرة في تقييم نقاط الضعف. غطينا على نطاق واسع الثغرات الأمنية الشائعة في تطبيقات الويب ، بما في ذلك OWASP top 10.من خلال التعاون الناجح مع الشركة ، أجرينا اختبار اختراق شامل ، باستخدام الاختبارات اليدوية والآلية باستخدام أدوات مثل Burp Suite و Metasploit. كان هدفنا مساعدة الشركة في التخفيف من المخاطر والبقاء متقدمة على التهديدات السيبرانية.
الكلمات المفتاحية: اختبار الاختراق،OWASP top 10،تطبيقات الويب،الثغرات الأمنية.

Abstract: In today's digital landscape, companies with web applications face an ongoing challenge of defending against cyberattacks. Proactive measures such as penetration testing play a crucial role in identifying vulnerabilities before they can be exploited. We highlighted the importance of penetration testing enabling companies to make informed decisions when evaluating weaknesses. We extensively covered common security vulnerabilities in company's web application, including the OWASP top 10. Through a successful collaboration with the Company, we conducted a comprehensive penetration test using both manual and automated tests with tools like Burp Suite and Metasploit. Our goal was to assist the company in mitigating risks and staying ahead of cyber threats.

Keywords: Penetration testing, OWASP top 10, web applications, security vulnerabilities.

Résumé : Dans le paysage numérique d'aujourd'hui, les entreprises dotées d'applications web sont confrontées à un défi constant de défense contre les cyberattaques. Des mesures proactives telles que les tests de pénétration jouent un rôle crucial dans l'identification des vulnérabilités avant leur exploitation. Nous avons souligné l'importance des tests de pénétration pour permettre aux entreprises de prendre des décisions éclairées lors de l'évaluation des faiblesses. Nous avons largement abordé les vulnérabilités de sécurité courantes dans les applications web, y compris les dix premières du projet OWASP. Grâce à une collaboration réussie avec la société, nous avons réalisé un test de pénétration complet en utilisant des tests manuels et automatisés avec des outils tels que Burp Suite et Metasploit. Notre objectif était d'aider l'entreprise à atténuer les risques et à rester en avance sur les menaces cybernétiques.

Mots-clés : Tests de pénétration, OWASP top 10, applications web, vulnérabilités de sécurité.