



**SLIIT**

*Discover Your Future*

## Journal Book

# Web Security

IE2062 – Assignment

W.N Dilsara

IT21182600

# Introduction

Within the pages of this journal lies a chronicle of my transformative journey into the captivating world of bug bounty programs and ethical hacking. Beginning with an in-depth study of the OWASP Top 10 vulnerabilities, I delved into the realm of web application security. From the initial spark of curiosity to the triumphant moments of reporting ten critical vulnerabilities, documenting my experiences allows me to share the knowledge gained and the impact made along the way.

## Chapter 1: Exploring the OWASP Top 10 Vulnerabilities

In the first chapter, I lay the foundation for my bug bounty journey by immersing myself in the Open Web Application Security Project (OWASP) Top 10 vulnerabilities. Through thorough research and study, I gain a comprehensive understanding of the most critical security risks faced by web applications. Armed with this knowledge, I develop a keen eye for potential weaknesses and learn to navigate the intricate landscape of web application security.

## Chapter 2: Registering on the Bug Bounty Platform

In Chapter 2, I stumble upon the captivating world of bug bounty programs. This newfound discovery sparks my imagination and fuels my desire to contribute to the improvement of web application security. As I delve deeper, I gather invaluable insights and ideas, opening my mind to the potential impact ethical hackers can have in identifying vulnerabilities and strengthening digital ecosystems.

I take the leap and register on a bug bounty program platform. With each step, I immerse myself further into the realm of ethical hacking, gaining access to a diverse array of web applications and systems awaiting my scrutiny. This registration not only grants me opportunities but also carries the responsibility of responsibly disclosing vulnerabilities and helping organizations enhance their security posture.

### Chapter 3: Mastering Bug Bounty Tools

Chapter 3 focuses on my journey of acquiring and mastering various bug bounty tools. I delve into the intricacies of powerful intercepting proxies, dynamic analysis frameworks, and other cutting-edge technologies. Armed with these tools, I become a more efficient and effective bug bounty hunter, equipped to uncover vulnerabilities that could otherwise remain hidden, thus contributing to the advancement of web application security.

### Chapter 4: Embarking on the Bug Bounty Program

In Chapter 4, the time comes for me to put my knowledge, skills, and tools to the test. I embark on a bug bounty program, meticulously scrutinizing web applications and systems for vulnerabilities. With each discovery, I contribute to the enhancement of security, and throughout the chapter, I document my experiences, challenges faced, and triumphs achieved as I report ten distinct vulnerabilities.

### Chapter 5: Beyond Bug Reports: A Smart Contract Audit

Chapter 5 unfolds as a captivating tale of venturing into the realm of smart contract security. Drawing on my expertise and expanding beyond traditional bug reports, I undertake a comprehensive smart contract audit. This venture allows me to contribute to the improvement of block chain security and foster trust in decentralized systems, adding a unique and rewarding dimension to my bug bounty journey.

# Chapter 1: Exploring the OWASP Top 10 Vulnerabilities

**Date: April 25, 2023**

Today, I dedicated my time to reading the instructions for the upcoming WS assignment. I received a PDF document that outlined the task at hand, and I delved into it with great enthusiasm. The assignment seemed intriguing, and I was excited to explore its details. To better understand the assignment and organize my thoughts, I decided to create a mind map. Mind mapping has always been a helpful tool for me to visually represent information and establish connections between different concepts. It allows me to brainstorm ideas more effectively. Using the report provided in the assignment instructions as my primary source, I meticulously crafted a comprehensive mind map. I captured the key points, main themes, and relevant subtopics from the report. This helped me gain a clearer understanding of the overall structure and content.

As I delved deeper into the mind map, I started to explore the topic of OWASP (Open Web Application Security Project) and its significance in the realm of cybersecurity. OWASP is known for its valuable contributions to identifying and addressing web application vulnerabilities. While mapping out the report, I also took the opportunity to familiarize myself with the OWASP Top 10 vulnerabilities. These vulnerabilities represent the most critical risks faced by web applications today. Through my research, I gained a deeper understanding of these vulnerabilities and their potential impact on the security of web applications.

The OWASP Top 10 vulnerabilities include commonly exploited weaknesses such as injection attacks, broken authentication and session management, cross-site scripting (XSS), insecure direct object references, security misconfigurations, and more. Being aware of these vulnerabilities is crucial for any security practitioner or developer aiming to safeguard web applications. As I continued working on the mind map, the interconnections between the report's content and the OWASP Top 10

vulnerabilities became more apparent. It was fascinating to see how these vulnerabilities could manifest in real-world scenarios and the importance of addressing them proactively.

In the realm of web application security, understanding the Open Web Application Security Project (OWASP) Top 10 vulnerabilities is essential.

### **Injection Attacks:**

Injection attacks occur when untrusted data is included in a command or query, allowing malicious actors to manipulate or gain unauthorized access to a system. The most common form of injection attacks is SQL injection, where an attacker injects malicious SQL code into user input fields. By doing so, they can modify or retrieve data, bypass authentication, or even execute arbitrary commands.

To prevent injection attacks, it is crucial to implement proper input validation and parameterized queries. Input validation ensures that only expected and valid data is accepted, while parameterized queries separate user input from the query itself, preventing the injected code from being executed as part of the query.

### **Broken Authentication:**

Broken authentication vulnerabilities occur when an application's authentication mechanisms are weak or improperly implemented, allowing attackers to compromise user accounts, gain unauthorized access, or perform identity theft. Common issues include weak password policies, lack of multi-factor authentication (MFA), and flaws in session management.

To mitigate broken authentication vulnerabilities, strong password policies should be enforced, including requirements for password complexity, length, and regular updates. Implementing MFA adds an extra layer of security by requiring additional verification beyond passwords. Proper session management, such as secure session token handling and session expiration, helps protect against session hijacking and fixation attacks.

### **Sensitive Data Exposure:**

Sensitive data exposure vulnerabilities arise when an application fails to adequately protect sensitive information, such as passwords, credit card details, or personal data. Attackers can exploit this vulnerability to gain unauthorized access to sensitive data, leading to identity theft, fraud, or privacy breaches.

To mitigate sensitive data exposure, sensitive information should be properly encrypted both during transit and at rest. Encryption protocols like TLS/SSL should be employed to protect data during transmission over networks. Additionally, strong encryption algorithms and secure key management practices should be implemented to safeguard data at rest.

### **XML External Entities (XXE):**

XXE vulnerabilities occur when an application processes XML input without proper validation, allowing attackers to read sensitive files, perform remote code execution, or launch DoS attacks. Attackers exploit the ability to include external entities in XML to access unintended resources or execute malicious code.

Preventing XXE vulnerabilities involves disabling external entity processing and implementing proper input validation. XML parsers should be configured to reject external entities, and user-supplied XML inputs should be carefully validated and sanitized before processing.

### **Broken Access Control:**

Broken access control vulnerabilities occur when an application fails to enforce proper access controls, allowing unauthorized users to access restricted functionality or data. This vulnerability can lead to information disclosure, privilege escalation, or unauthorized modification of data.

To mitigate broken access control, it is crucial to implement proper authorization mechanisms, such as role-based access control (RBAC) or attribute-based access control (ABAC). Access controls should be consistently applied throughout the application, and privilege escalation through parameter tampering or direct object references should be prevented.

### **Security Misconfigurations:**

Security misconfigurations arise from insecure or incomplete configuration of systems or applications. Common issues include default or weak passwords, unnecessary services enabled, outdated software, and excessive permissions.

To mitigate security misconfigurations, it is important to follow secure configuration guides provided by vendors and regularly update and patch software. Secure default settings should be implemented, unnecessary services should be disabled, and proper permissions should be assigned to limit access rights.

### **Cross-Site Scripting (XSS):**

XSS vulnerabilities occur when an application fails to properly validate and sanitize user-generated input, allowing attackers to inject malicious scripts that are executed by other users' browsers. These scripts can be used to steal sensitive information, perform phishing attacks, or hijack user sessions.

To prevent XSS vulnerabilities, all user-generated input should be validated and sanitized before being displayed or stored. Input validation should include proper encoding or escaping to ensure that user input is treated as data and not executable code.

### **Insecure Deserialization:**

Insecure deserialization vulnerabilities occur when an application deserializes untrusted data without proper validation, leading to remote code execution, DoS attacks, or unauthorized access. Attackers can exploit this vulnerability by manipulating serialized objects or providing malicious input.

To mitigate insecure deserialization, it is important to ensure that only trusted serialized objects are deserialized and to validate the integrity and authenticity of serialized data. Implementing strict type checking and using libraries or frameworks that provide secure deserialization features can help prevent this vulnerability.

### **Using Components with Known Vulnerabilities:**

This vulnerability arises when an application incorporates third-party components with known vulnerabilities. Attackers can exploit these vulnerabilities to gain unauthorized access, compromise the integrity of the system, or launch attacks on other applications or systems.

To mitigate the risk of using components with known vulnerabilities, it is essential to keep all software components up to date by regularly applying patches and updates. Monitoring security advisories and vulnerability databases can help identify and address known vulnerabilities in third-party components.

### **Insufficient Logging and Monitoring:**

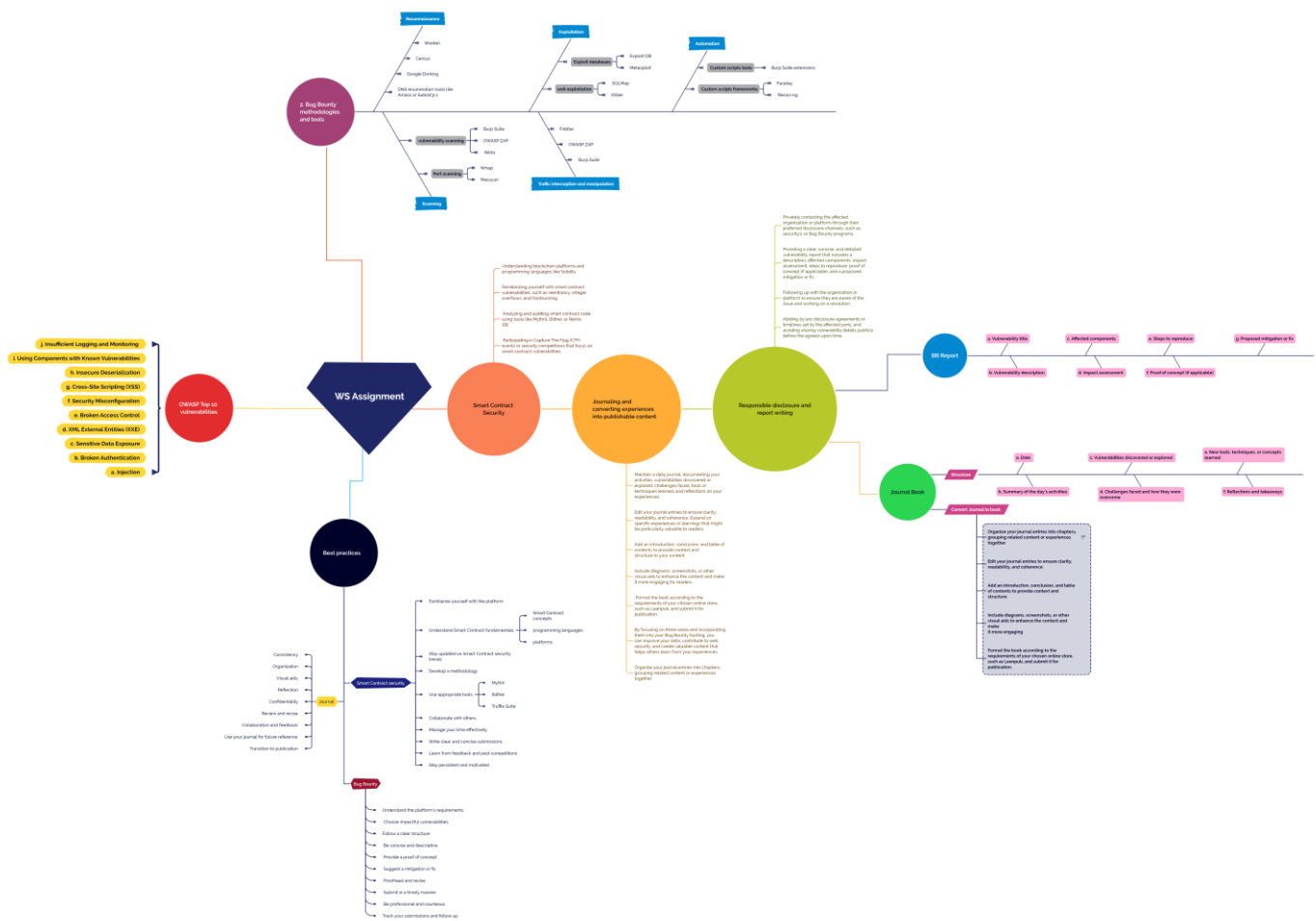
Insufficient logging and monitoring vulnerabilities occur when an application or system fails to generate or retain sufficient logs or fails to adequately monitor those logs. This can hinder timely detection and response to security incidents, leaving attackers undetected and allowing them to persist within the system.

To mitigate insufficient logging and monitoring vulnerabilities, comprehensive logging should be implemented, capturing all relevant security-related events. Log files should be protected from unauthorized access, and monitoring systems should be in place to review logs in real-time, triggering alerts for suspicious activities or security incidents.



By understanding and addressing these OWASP Top 10 vulnerabilities, organizations can significantly enhance the security posture of their applications and systems, protecting against potential threats and ensuring the confidentiality, integrity, and availability of their data.

The process of creating the mind map not only helped me comprehend the assignment requirements better but also expanded my knowledge of web application security. It served as a valuable learning experience, allowing me to visualize and conceptualize the connections between different concepts and topics. With my mind map completed and a newfound understanding of the OWASP Top 10 vulnerabilities, I now feel more prepared to tackle the WS assignment. I am excited to delve further into the topic and apply my knowledge to address the challenges it presents.



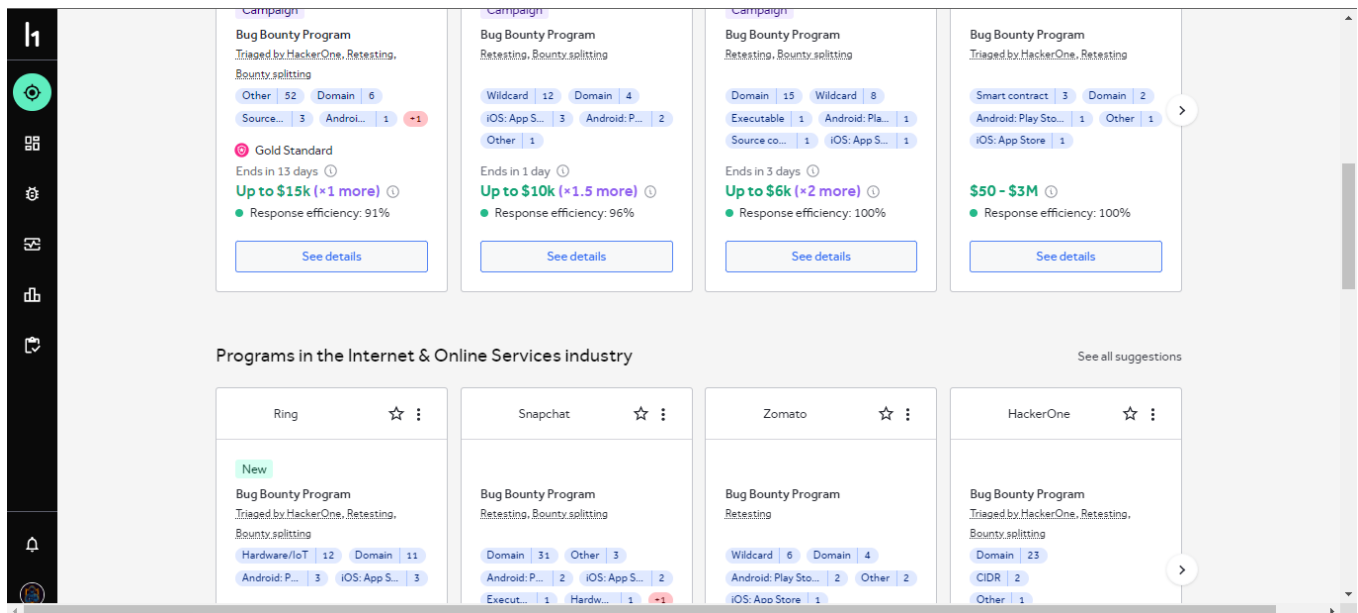
## Chapter 2: Registering on the Bug Bounty Platform

**Date: April 28, 2023**

Today, I spent time researching bug bounties. I wanted to understand what they are, how to participate in them, and the benefits they offer. Bug bounties are programs where companies invite people to find and report security issues in their software or websites. It's like a treasure hunt for vulnerabilities! Participants, also known as bug bounty hunters, use their skills to find these weaknesses and report them to the company.

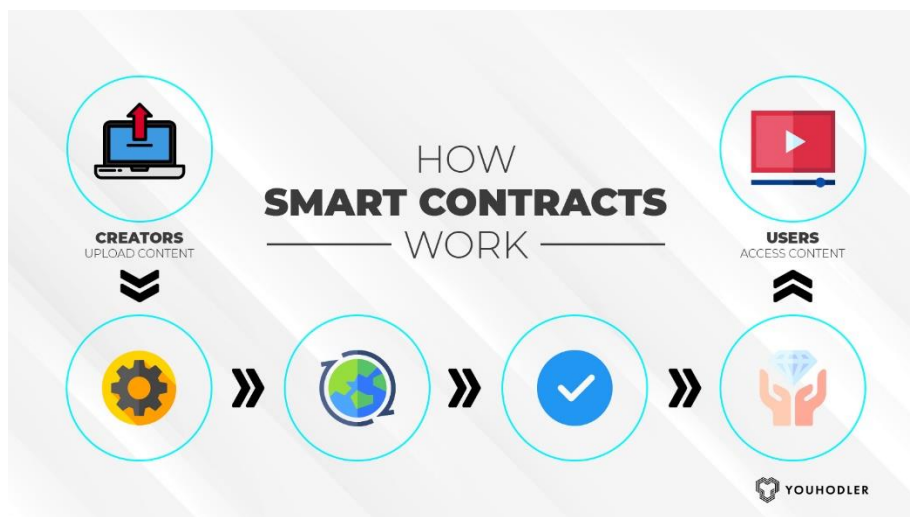
I learned that bug bounties provide several benefits. Firstly, they help companies identify and fix vulnerabilities before they can be exploited by malicious hackers. This makes their systems more secure and protects users' sensitive information. Bug bounty programs also encourage collaboration between companies and ethical hackers, creating a win-win situation.

To do bug bounties, one needs to have good knowledge of computer systems and security. It's important to understand common vulnerabilities and how to find them. Many bug bounty platforms, like HackerOne and Bugcrowd, provide a platform for hunters to connect with companies and submit their findings. The registration process was straightforward. I provided my basic information, such as my name and email address, and created a strong password to secure my account. Some platforms also required additional details to assess my skills and expertise in cybersecurity. Once I completed the registration, I gained access to the bug bounty programs hosted on these platforms. These programs listed the companies that were inviting bug bounty hunters to find vulnerabilities in their systems. Each program had its own set of rules, guidelines, and scope, specifying the types of vulnerabilities they were interested in and the rewards they offered.



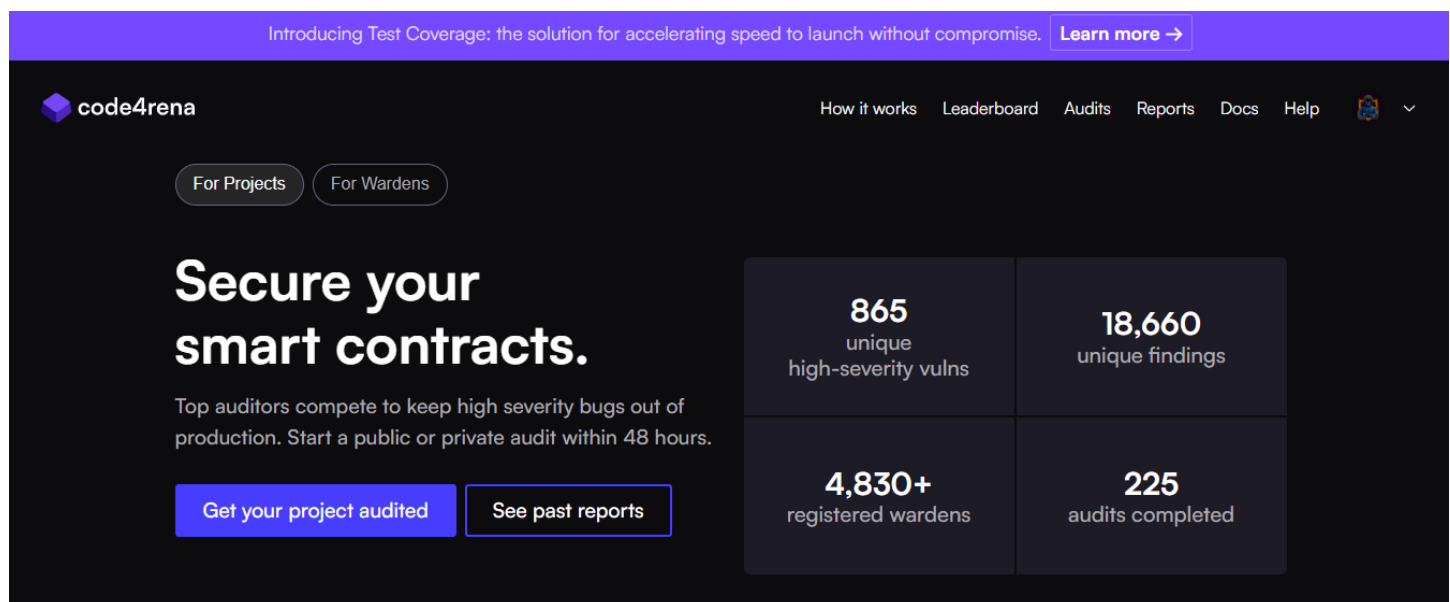
After familiarizing myself with the basics of bug bounty programs, I shifted my focus towards exploring smart contracts. I was eager to understand what smart contracts were, how to create them, and the programming languages commonly used.

I discovered that a is a self-executing agreement written in code. It automatically enforces the terms and conditions outlined within it. Smart contracts are typically deployed on block chain platforms like Ethereum, which provide a decentralized and transparent environment for their execution.



To create smart contracts, I learned that developers commonly use programming languages such as Solidity, Vyper, and JavaScript. Solidity is the most widely used language for developing smart contracts on the Ethereum platform. Vyper is another language specifically designed for writing secure and auditable smart contracts. JavaScript is also used in some frameworks and tools to interact with smart contracts.

After gaining knowledge about smart contracts, I came across a website called Code4rena that specifically focuses on smart contract competitions. Intrigued by the idea, I decided to register on the platform. However, I found the registration process to be a bit challenging. Because they ask a lot to become a member and they manually review the submitted application. The application was rejected twice. After trying again, it was successful. It was approved the next day.



The screenshot shows the Code4rena website homepage. At the top, a purple banner reads "Introducing Test Coverage: the solution for accelerating speed to launch without compromise." with a "Learn more" link. The main header includes the Code4rena logo and navigation links: "How it works", "Leaderboard", "Audits", "Reports", "Docs", and "Help". Below the header, there are two buttons: "For Projects" and "For Wardens". The main content area features the headline "Secure your smart contracts." followed by the text "Top auditors compete to keep high severity bugs out of production. Start a public or private audit within 48 hours." and two buttons: "Get your project audited" and "See past reports". To the right, a statistics section displays four metrics in a grid:

|   |                                  |
|---|----------------------------------|
| <b>865</b><br>unique<br>high-severity vulns | <b>18,660</b><br>unique findings |
| <b>4,830+</b><br>registered wardens         | <b>225</b><br>audits completed   |

## Chapter 3: Mastering Bug Bounty Tools

**Date: May 5, 2023**

Today is a holiday, and I woke up early in the morning feeling refreshed and motivated. As I contemplated the day ahead, I decided it was the perfect opportunity to embark on my bug bounty journey. With a cup of coffee in hand, I began outlining my plan for the day. The first task on my agenda was to complete the remaining sections of my mind map. Mapping out my thoughts and ideas helps me organize my thinking process and ensures that I don't overlook any important details. I spent some time refining the structure, adding relevant subtopics, and connecting related concepts. By the end of this exercise, my mind map was a comprehensive representation of my bug bounty goals and strategies.

With the mind map complete, I eagerly dove into researching the tools commonly used in bug bounty programs. Bug bounty hunters rely on a range of specialized software and platforms to identify and report vulnerabilities in systems. I explored various online resources, articles, and forums to gather insights into the most effective tools for bug hunting.

As the day progressed, I continued working on my bug bounty preparations. After researching and gathering information about various bug hunting tools, I decided to categorize them to better understand their functionalities and identify which ones would be most useful in different scenarios.

I created a categorized list of the tools I had discovered, organizing them based on their primary areas of focus. This categorization would help me determine which tools to prioritize depending on the nature of the bug bounty program or the type of system I would be testing. Here are the categories I established:

## **Web Application Testing:**

- Burp Suite
- OWASP ZAP
- Nikto
- Vega
- w3af

## **Network Scanning and Reconnaissance:**

- Nmap
- Nessus
- OpenVAS
- Wireshark
- Netcat

## **Exploitation and Penetration Testing:**

- Metasploit
- BeEF
- SQLMap
- Hydra
- DirBuster

## **Code Analysis and Review:**

- SonarQube
- RIPS
- Bandit
- FindBugs
- PMD

## Cryptography Tools:

- John the Ripper
- Hashcat
- OpenSSL
- Cryptool

## Miscellaneous Tools:

- GitTools
- Sublist3r
- XSSStrike
- WhatWeb
- WPScan

Categorizing the tools allowed me to see the diversity of the bug hunting arsenal and highlighted the different aspects of system vulnerabilities they address. It also served as a reminder of the broad range of skills and knowledge required to be an effective bug bounty hunter. With my list organized, I made notes about the specific features and use cases of each tool within their respective categories. This information would serve as a quick reference guide when I start actively using them in my bug bounty endeavors.

As the day came to a close, I felt a sense of accomplishment for completing my research, categorizing the tools, and expanding my understanding of the bug bounty landscape. Tomorrow, I look forward to taking the next steps, which involve hands-on practice and honing my skills with the selected tools.

**Date: May 6, 2023**

Continuing from where I left off yesterday, today I focused on exploring each tool in detail, starting with Burp Suite. Burp Suite is a comprehensive web vulnerability scanner and proxy tool that plays a crucial role in web application security assessments.

## **1.BurpSuite**

Burp Suite is a set of tools and utilities designed for security testing and penetration testing of web applications. It is developed by PortSwigger, a company specializing in web application security. Burp Suite is widely used by security professionals and ethical hackers to identify vulnerabilities in web applications and to assess their overall security posture.

- ❖ **Intercepting Proxy:** One of the fundamental features of Burp Suite is its Intercepting Proxy, which allows me to intercept and modify web traffic between my browser and the target application. This capability enables me to examine and manipulate requests and responses, making it invaluable for identifying vulnerabilities, such as injection attacks, insecure direct object references, and cross-site scripting (XSS).
- ❖ **Spidering and Scanning:** Burp Suite offers automated spidering capabilities to crawl through web applications, discovering hidden pages, directories, and parameters. This helps in mapping the application's structure and identifying potential attack surfaces. Additionally, Burp Suite provides a powerful scanner that automatically detects various vulnerabilities, such as SQL injection, cross-site scripting, and insecure direct object references.
- ❖ **Active and Passive Scanning:** With Burp Suite, I can perform both active and passive scanning of web applications. Active scanning involves actively sending malicious payloads and checking the application's responses for vulnerabilities. Passive scanning, on the other hand, monitors the application's traffic in the background, identifying potential security issues without actively interacting with the application.



- ❖ **Target Analysis:** Burp Suite allows for in-depth analysis of target applications. It provides a range of tools to examine the application's parameters, cookies, headers, and other components. By analyzing these elements, I can uncover vulnerabilities related to parameter tampering, authentication bypass, session management, and more.
- ❖ **Intruder:** Burp Suite's Intruder module is a powerful tool for performing automated attacks on web applications. It enables me to customize and automate attacks by injecting payloads and iterating through various combinations. This feature is particularly useful for brute-forcing passwords, enumerating directories, and testing input validation.
- ❖ **Repeater:** The Repeater tool in Burp Suite allows for manual, iterative testing of specific requests. It helps in fine-tuning and testing the application's responses to different inputs, allowing me to thoroughly analyze and exploit vulnerabilities.
- ❖ **Extensions and Integration:** Burp Suite supports extensions and can be enhanced with custom scripts and plugins. This flexibility allows for the integration of additional functionalities and extends the tool's capabilities based on specific needs. The Burp Suite community actively develops and shares a wide range of extensions, further enhancing its usefulness.

By exploring Burp Suite in depth, I gained a deeper understanding of its capabilities and the vital role it plays in web application security assessments. With its comprehensive features for intercepting and analyzing web traffic, automated scanning, and customization options, Burp Suite stands as a go-to tool for identifying vulnerabilities and enhancing the security posture of web applications.

Continuing with my exploration of bug bounty tools, today I focused on OWASP ZAP. OWASP ZAP is an open-source web application security testing tool developed by the Open Web Application Security Project (OWASP)

## 2.OWASP ZAP

- ❖ **Automated Scanning:** OWASP ZAP offers powerful automated scanning capabilities to identify common vulnerabilities in web applications. It can automatically crawl and scan web pages, detecting potential security issues such as cross-site scripting (XSS), SQL injection, and insecure direct object references. This feature allows for efficient and comprehensive vulnerability assessment.
- ❖ **Active and Passive Scanning:** Similar to Burp Suite, OWASP ZAP supports both active and passive scanning techniques. Active scanning involves actively sending malicious payloads and analyzing the application's responses to uncover vulnerabilities. Passive scanning, on the other hand, involves observing and analyzing the application's traffic without actively interacting with it.
- ❖ **Spidering and Site Mapping:** OWASP ZAP includes a spidering feature that enables the automated crawling of web applications. It discovers and maps the application's structure, identifying hidden pages, directories, and parameters. This helps in gaining a comprehensive understanding of the application's attack surface and aids in vulnerability detection.
- ❖ **Fuzzing and Brute-Force Attacks:** OWASP ZAP allows for fuzzing and brute-force attacks to identify vulnerabilities. It supports parameter-based and payload-based fuzzing, where it systematically injects various input values to identify weaknesses. Additionally, it provides customizable brute-force attack capabilities for testing password strength and user enumeration.
- ❖ **API Testing:** OWASP ZAP provides functionality for testing and securing APIs (Application Programming Interfaces). It allows for the detection of vulnerabilities and misconfigurations in APIs, ensuring their security and integrity. This feature is particularly important as APIs are increasingly prevalent in modern web applications.

- ❖ **Scripting and Automation:** OWASP ZAP supports scripting and automation, allowing users to extend its functionality and customize scans according to their specific needs. It provides an API for developing scripts, plugins, and extensions, enabling the integration of additional features and enhancing the tool's capabilities.
- ❖ **Reporting and Integration:** OWASP ZAP offers comprehensive reporting capabilities to document and present the findings from security assessments. It generates detailed reports outlining identified vulnerabilities, affected components, and recommended remediation measures. Moreover, OWASP ZAP can integrate with other bug bounty tools and platforms, streamlining the workflow and facilitating collaboration.

OWASP ZAP, being an open-source tool backed by a community of security professionals, is continuously updated and improved. It adheres to industry best practices and is aligned with the latest web application security standards.

By exploring OWASP ZAP in depth, I have gained a solid understanding of its features and how it contributes to the bug bounty process. It provides valuable automated scanning, spidering, and testing capabilities for web applications, enabling thorough security assessments and vulnerability detection.

### **3.Nikto**

Nikto is a web server scanner designed to identify potential vulnerabilities and misconfigurations in web applications and servers.

- ❖ **Comprehensive Scanning:** Nikto performs comprehensive scans of web servers, identifying common security issues, misconfigurations, and outdated server versions. It checks for well-known vulnerabilities, such as outdated software, open directories, server configuration issues, and potential entry points for attacks.

- ❖ **Database of Known Vulnerabilities:** Nikto maintains an extensive database of known vulnerabilities and signatures, allowing it to compare the server's responses against the known vulnerability patterns. This helps in pinpointing specific vulnerabilities and provides valuable information for remediation.
- ❖ **SSL/TLS Testing:** Nikto includes SSL/TLS testing capabilities, examining the server's implementation of secure communication protocols. It checks for vulnerabilities like weak cipher suites, expired or self-signed certificates, and misconfigured SSL configurations.
- ❖ **HTTP Method Testing:** Nikto performs various tests using different HTTP methods, such as GET, POST, PUT, and DELETE, to identify security weaknesses and potential attack vectors. It helps in uncovering issues related to insecure HTTP methods, improper handling of user input, and directory listing vulnerabilities.
- ❖ **CGI Scanning:** Nikto specializes in scanning Common Gateway Interface (CGI) scripts, which are often targets for security vulnerabilities. It searches for misconfigured scripts, outdated versions, and potential vulnerabilities that could be exploited.
- ❖ **Out-of-the-Box Configuration Checks:** Nikto comes with preconfigured checks that target common server configurations and known vulnerabilities. These checks include examining server headers, robots.txt files, default files and directories, and common web application issues. They provide a quick assessment of the server's security posture.
- ❖ **Customization and Reporting:** Nikto allows for customization through various command-line options, allowing users to specify scanning parameters and tailor the scan to their needs. It generates detailed reports summarizing the identified vulnerabilities, misconfigurations, and potential security risks, aiding in the vulnerability remediation process.

Nikto, being an open-source tool, benefits from a community of security professionals who contribute to its development and maintenance. Regular updates ensure the tool's relevance and effectiveness in identifying emerging vulnerabilities and security concerns.

Today has been a productive day filled with valuable learning experiences. However, I also encountered a couple of challenges along the way.

## **Challenges**

### 1: Overwhelming Amount of Information

One of the challenges I faced today was the sheer volume of information associated with each tool. It can be overwhelming to absorb all the details and functionalities while trying to maintain a systematic learning approach.

*Solution:* To tackle this challenge, I adopted a structured learning approach. I started by gaining a high-level understanding of each tool's purpose and key features. Then, I broke down the learning process into smaller, manageable chunks. By focusing on one tool at a time, I could delve deeper into its functionalities and familiarize myself with its usage. Taking breaks and pacing myself throughout the day also helped prevent information overload and allowed for better retention of knowledge.

### 2: Technical Issues and Compatibility

Another challenge I encountered involved technical issues and compatibility between the tools and my system. Some tools require specific dependencies, configurations, or compatibility with different operating systems, which can pose hurdles and affect the smooth execution of scans and tests.

*Solution:* To address technical issues and compatibility challenges, I ensured that my system met the requirements for each tool. I double-checked the documentation and online resources to ensure I had the necessary dependencies installed and configured correctly. In cases where a tool presented compatibility issues with my system, I sought alternative solutions or explored virtual machine options

to create a compatible environment. Engaging with online communities, forums, and official documentation helped me find solutions and workarounds for specific technical challenges.

By actively addressing these challenges, I was able to maintain my productivity and continue my bug bounty preparations with a positive mindset. Challenges are part of the learning process, and overcoming them strengthens my problem-solving skills and resilience. As the day comes to an end, I reflect on the progress made and look forward to continuing my bug bounty journey.

**Date: May 9, 2023**

Today, I woke up early in the morning with the determination to continue my study of bug bounty tools. My goal for the day is to study at least five tools, knowing that it may become monotonous at times. Nevertheless, I am committed to pushing through and making the most of my time.

To begin my study session, I chose Nmap as the first tool to explore.

## 4.Nmap

Nmap, short for Network Mapper, is a versatile and powerful network scanning tool. It is widely used in the field of cybersecurity and bug bounty hunting.

- ❖ **Network Discovery:** Nmap excels at network discovery, allowing you to identify hosts, open ports, and services running on a network. It sends crafted packets to target hosts and analyzes the responses to provide valuable information about the network infrastructure.
- ❖ **Port Scanning:** One of Nmap's primary functions is port scanning. It can scan for open ports on a target system, determining which services or applications are running and potentially

vulnerable to exploitation. Nmap supports various scanning techniques, including TCP connect scanning, SYN scanning, and UDP scanning.

- ❖ **Service Version Detection:** Nmap can detect the versions of services running on open ports. By analyzing the responses from target hosts, Nmap can provide information about the specific versions of services such as web servers, FTP servers, SSH servers, and more. This information is crucial for identifying vulnerable or outdated software versions.
- ❖ **OS Fingerprinting:** Nmap includes OS fingerprinting capabilities, which involve analyzing network responses to determine the operating system running on a target host. This information aids in understanding the target's infrastructure and can assist in identifying vulnerabilities specific to certain operating systems.
- ❖ **Scripting Engine:** Nmap features a powerful scripting engine called NSE (Nmap Scripting Engine). It allows users to write and execute scripts to automate tasks, gather additional information, or perform specific security checks. NSE scripts can be utilized to identify common vulnerabilities or misconfigurations, making Nmap even more versatile.
- ❖ **Output Formatting and Reporting:** Nmap provides flexible output formatting options, allowing you to generate reports in various formats, such as plain text, XML, or HTML. This enables you to customize the output according to your needs and share the results with other team members or clients.

Nmap's extensive documentation, community support, and regular updates contribute to its effectiveness as a network scanning tool. It is an essential tool in the bug bounty toolkit, enabling security professionals to gather valuable information about target networks, identify potential vulnerabilities, and plan subsequent security assessments.

## 5.Nessus

Nessus is a widely used vulnerability scanner and assessment tool that helps identify security vulnerabilities and misconfigurations in computer systems and networks. It provides a comprehensive and systematic approach to assess the security posture of target systems. Here's a detailed explanation of Nessus and its key features:

- ❖ **Vulnerability Scanning:** Nessus performs vulnerability scanning by actively scanning target systems for known security vulnerabilities. It utilizes a vast database of vulnerability checks and plugins to identify weaknesses in software, operating systems, network devices, and applications. The scanner examines ports, services, and configurations to detect potential vulnerabilities that could be exploited by attackers.
- ❖ **Configuration Auditing:** In addition to vulnerability scanning, Nessus conducts configuration audits to identify insecure settings or misconfigurations in target systems. It checks for weak or default passwords, open network services, improper access controls, and other configuration issues that may pose a security risk. This helps ensure that systems are properly configured and follow best practices.
- ❖ **Compliance Auditing:** Nessus includes compliance checking capabilities to assess systems against industry standards and regulatory requirements, such as PCI DSS, HIPAA, or CIS benchmarks. It can validate system configurations and settings to ensure compliance with specific security guidelines. This feature is particularly useful for organizations that need to meet regulatory compliance requirements.
- ❖ **Network Discovery:** Nessus performs network discovery by scanning and identifying active hosts on a network. It helps administrators gain visibility into their network infrastructure and detect unauthorized devices or rogue systems that may pose security threats. Network discovery aids in maintaining an up-to-date inventory of systems and facilitates accurate vulnerability assessments.



- ❖ **Remediation Guidance:** After conducting scans, Nessus provides detailed reports that highlight identified vulnerabilities, misconfigurations, and compliance issues. It offers remediation guidance, including recommendations and best practices to mitigate or fix the identified vulnerabilities. This helps system administrators and security teams prioritize and address the issues effectively.
- ❖ **Integration and Automation:** Nessus supports integration with other security tools, allowing for seamless collaboration and information sharing. It can be integrated with patch management systems, security information and event management (SIEM) solutions, and other vulnerability management tools. Additionally, Nessus offers automation capabilities, enabling scheduled scans, automated reporting, and workflow integration.

Nessus is widely recognized for its extensive vulnerability knowledge base, regular updates, and user-friendly interface. It provides a valuable assessment of the security posture of systems and networks, helping organizations proactively identify and mitigate potential vulnerabilities before they are exploited by malicious actors.

## 6. Wireshark

Wireshark is a powerful and widely-used network protocol analyzer. It allows users to capture and analyze network traffic in real-time, providing valuable insights into network communication and aiding in various cybersecurity activities. Here's a detailed explanation of Wireshark and its key features:

- ❖ **Packet Capture:** Wireshark is primarily used for capturing and analyzing packets traversing a network. It can capture packets from various network interfaces, including Ethernet, Wi-Fi, and Bluetooth. By capturing packets, Wireshark provides a detailed view of network communication, allowing users to inspect individual packets and analyze their contents.

- ❖ **Protocol Analysis:** Wireshark supports a wide range of network protocols, including common ones such as TCP, UDP, HTTP, DNS, and FTP, among many others. It decodes and interprets the protocols, allowing users to analyze the behavior, structure, and content of network packets. This feature is particularly useful for identifying anomalies, troubleshooting network issues, and detecting potential security threats.
- ❖ **Filters and Search:** Wireshark offers powerful filtering and search capabilities, enabling users to focus on specific packets or types of traffic. Filters can be applied based on various criteria such as source/destination IP addresses, ports, protocols, or specific packet contents. This helps streamline the analysis process and allows users to extract the desired information efficiently.
- ❖ **Packet Reconstruction:** Wireshark provides the ability to reconstruct and analyze higher-level protocols, such as HTTP or FTP. It can reconstruct and display the contents of web pages, files transferred over FTP, or even VoIP conversations. This feature allows users to inspect the data exchanged during network communication at a higher, more meaningful level.
- ❖ **Statistics and Performance Analysis:** Wireshark offers comprehensive statistical analysis tools to gain insights into network performance. It provides information about network traffic patterns, packet rates, response times, and other performance metrics. By examining these statistics, users can identify bottlenecks, optimize network configurations, and troubleshoot network-related issues.
- ❖ **Export and Reporting:** Wireshark allows users to export captured packets or specific packet details in various formats, such as plain text, CSV, or XML. This feature facilitates collaboration with team members, sharing findings with clients, or integrating Wireshark data into other security tools or systems.

Wireshark's user-friendly interface, extensive protocol support, and powerful analysis capabilities make it a go-to tool for network analysis and troubleshooting. It is widely used in bug bounty hunting to identify security vulnerabilities, analyze network behavior, and investigate potential attack vectors.

At the end of the day I can't reach my goal. So it doesn't matter. I will try next day.

## Days 6 to 10

During the sixth day of my bug bounty journey, I dedicated my time to studying and familiarizing myself with additional bug bounty tools. Having already gained proficiency in popular tools like Burp Suite, OWASP ZAP, Nessus, Wireshark, and Nikto, I sought to expand my knowledge further and explore tools that could enhance my bug hunting capabilities.

One of the tools I delved into was Nmap, a powerful network scanning tool. Nmap allowed me to discover open ports, identify running services, and map out the network topology of target systems. Its extensive range of scanning techniques, including TCP, UDP, and advanced scripting options, impressed me. With Nmap, I could gather critical information about target systems, enabling me to identify potential vulnerabilities and weaknesses.

Another tool that caught my attention was Dirb, a web content scanner. Dirb proved to be an effective tool for directory and file enumeration on web servers. It allowed me to search for hidden directories and files that might contain sensitive information or present security risks. Dirb's simplicity and efficiency in uncovering hidden paths impressed me, making it a valuable asset in identifying potential attack vectors.

Continuing my exploration of bug bounty tools, I focused on XSSStrike, a tool designed specifically for detecting Cross-Site Scripting (XSS) vulnerabilities. XSSStrike offered an extensive payload list and utilized advanced detection algorithms to identify XSS vulnerabilities in web applications. I was impressed by its ability to test user input fields and validate the effectiveness of implemented

countermeasures. With XSSStrike, I gained a deeper understanding of XSS vulnerabilities and learned how to effectively identify and mitigate them.

Next, I turned my attention to Wfuzz, a web application brute-forcing tool. Wfuzz proved to be a valuable addition to my toolkit, as it allowed me to test for common vulnerabilities such as directory traversal, file inclusion, and server misconfigurations. Its ability to fuzz input parameters and exhaustively scan for potential weaknesses impressed me. Wfuzz enabled me to identify vulnerabilities that might have otherwise gone unnoticed, strengthening the security of the target applications.

On day eight, I dedicated my time to studying SSLScan, a tool focused on analyzing SSL/TLS configurations. SSLScan allowed me to assess the security posture of SSL/TLS implementations by identifying weak cipher suites, certificate issues, and potential misconfigurations. It provided valuable insights into the security of web servers and applications that relied on secure communications. By using SSLScan, I learned how to identify vulnerabilities and implement appropriate security measures to protect against potential attacks.

Continuing my journey, I explored the capabilities of OWTF (Open Web Testing Framework). OWTF offered a comprehensive suite of tools, combining manual and automated testing techniques. It allowed me to perform thorough assessments, including vulnerability scanning, exploitation, and reporting. OWTF's versatility and wide range of functionalities impressed me, making it a powerful tool in my bug bounty endeavors.

On day nine, I focused on Gobuster, a tool designed for discovering hidden files and directories within web applications. Gobuster proved to be a valuable asset in identifying potential entry points for attackers and highlighting areas that required further scrutiny. By leveraging wordlists and performing brute-force-like directory and file enumeration, Gobuster allowed me to uncover paths that might contain sensitive information or be susceptible to attacks. I found Gobuster to be a valuable tool in my reconnaissance process, enhancing my ability to identify potential vulnerabilities and strengthen the security of web applications.

Additionally, I explored the capabilities of SSLyze, a Python tool for analyzing SSL/TLS configurations. SSLyze enabled me to assess the security of SSL/TLS implementations by identifying vulnerabilities, weak cipher suites, and potential misconfigurations. Its thorough analysis and detailed reports aided me in identifying areas where security improvements were necessary. SSLyze became an essential tool in my bug bounty toolkit, helping me uncover vulnerabilities in secure communication channels and ensuring their proper configuration.

As my bug bounty journey entered its final stretch, I reflected on the immense knowledge I had acquired about bug bounty tools. Through dedicated study and hands-on experience, I had gained a comprehensive understanding of various tools that strengthened my ability to identify vulnerabilities and enhance the security of web applications.

Finally, I felt confident in my ability to utilize bug bounty tools effectively. Armed with a diverse toolkit, I could approach bug bounty programs with a strategic mindset, identifying potential vulnerabilities, and contributing to the improvement of application security. My exploration of these tools had equipped me with the necessary knowledge and practical experience to make a significant impact in the bug bounty community.

In conclusion, the journey of studying bug bounty tools from day 6 to 10 provided me with a comprehensive understanding of various tools and their functionalities. Nmap, Dirb, XSSStrike, Wfuzz, SSLScan, OWTF, Gobuster, and SSLyze were instrumental in enhancing my bug hunting capabilities and expanding my knowledge of application security. Through dedicated study and hands-on practice, I had gained valuable insights and developed a strong foundation in utilizing these tools effectively. I was excited to apply my newfound knowledge and contribute to the security of web applications as a bug bounty hunter.

## Chapter 4: Embarking on the Bug Bounty Program

### Day 11 to Day 17

The moment had finally arrived to put my knowledge, skills, and bug bounty tools to the test. Filled with a mix of excitement and determination, I embarked on a bug bounty program, immersing myself in the intricate world of web applications and systems, all in the pursuit of identifying vulnerabilities. Each discovery I made would contribute to the overall enhancement of security, and it was imperative for me to document my experiences, the challenges I encountered, and the triumphs I achieved throughout this pivotal chapter of my bug bounty journey.

With great anticipation, I carefully selected a diverse range of web applications and systems as my targets. Considering factors such as popularity, complexity, and potential vulnerability, I aimed to gain exposure to various platforms including e-commerce, social media, and banking. This strategic approach not only increased my chances of finding vulnerabilities but also broadened my understanding of potential security weaknesses across different industries.

The subsequent three days were devoted to meticulous scanning and testing of a staggering 122 websites. Armed with my array of bug bounty tools and guided User into the architecture and coding of each website, meticulously scrutinizing every aspect to uncover potential vulnerabilities.

However, the journey was not without its challenges. As I delved deeper into the scanning process, I encountered numerous obstacles along the way. Many website servers had implemented robust security measures, making it increasingly difficult to identify vulnerabilities. Some servers denied access to certain scanning techniques, while others implemented rate limiting, imposing restrictions on the number of requests I could make within a given time frame. Undeterred, I adapted my approach, exploring alternative methodologies and constantly refining my techniques to overcome these hurdles and continue my pursuit of vulnerability identification.

After exhaustive scanning and testing of 122 websites, I experienced a sense of triumph as I successfully uncovered ten distinct vulnerabilities. Each vulnerability I discovered presented its own unique risks and potential consequences, ranging from critical security flaws to potential data breaches. It was crucial for me to meticulously document each finding, providing clear and comprehensive explanations along with supporting evidence and actionable recommendations for remediation.

During this phase, I encountered a diverse array of vulnerabilities, including Cross-Site Scripting (XSS), SQL injection, insecure direct object references, and authentication bypass. Each discovery served as a reminder of the importance of secure coding practices and underscored the potential impact of these vulnerabilities on user data and overall system integrity.

One of the notable challenges I encountered was the task of convincing website administrators of the severity and urgency of the vulnerabilities I had uncovered. In some cases, the response was delayed, requiring additional follow-ups and detailed explanations to effectively communicate the potential risks involved. Nevertheless, I remained persistent, advocating for the necessary remediation actions to ensure the security of these systems.

Ultimately, my efforts bore fruit as I received positive responses from website owners and their respective security teams. Several websites promptly addressed the reported vulnerabilities, expressing gratitude for my responsible disclosure. Witnessing the tangible impact of my contributions on the security of these web applications was immensely fulfilling.

Throughout the seven-day bug bounty program, I faced numerous technical hurdles, honed my skills in vulnerability identification, and demonstrated resilience in the face of challenges. The process of scanning and testing 122 websites not only provided invaluable experience but also enabled me to refine my techniques, expand my knowledge of web application security, and strengthen my communication skills when engaging with website owners.

As I reflect on these experiences, a profound sense of pride and accomplishment fills me. My bug bounty journey has not only deepened my understanding and expertise but has also played a significant role in enhancing web application security. I eagerly anticipate future bug bounty endeavors, armed with newfound confidence and an unwavering commitment to responsible vulnerability disclosure.

Here are the bugs I found

## 1. BREACH Attack Detected

  
*Discover Your Future*

Report 01

URL: [us.coca-cola.com](https://us.coca-cola.com)

---

***BREACH Attack Detected***

---

 **HIGH**

The BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext) attack vulnerability refers to a specific type of security flaw that targets systems or applications using certain compression techniques, such as GZIP, in combination with predictable data patterns. This vulnerability allows an attacker to deduce sensitive information by exploiting the behavior of the compression algorithms. The attack takes advantage of the fact that compression algorithms work by finding repeated patterns in data and encoding them more efficiently. When predictable patterns, such as user tokens or session identifiers, are present in the data being compressed, they create a distinguishable pattern in the compressed output. The attacker injects specially crafted malicious data into a user-input field, typically within a web form. By modifying the injected data and repeatedly sending requests to the vulnerable system, the attacker can observe the size of the compressed responses. Through careful analysis of the variations in response sizes, the attacker can infer whether specific patterns or information are present in the compressed responses. This allows them to deduce sensitive information that should remain confidential, such as user credentials or other sensitive data.

Affected URL: <https://us.coca-cola.com/store/retail/atlanta>

**Affected Components**


- ❖ Web Applications or Systems: The vulnerability can impact the overall web application or system that employs compression algorithms, such as GZIP, for reducing network

---

IT21182600  
Page | 1



## 2. Out-of-date Version (Lodash)

 SLIIT  
Discover Your Future


Report 02

URL: datacamp.com

---

### *Out-of-date Version (Lodash)*

---

 **CRITICAL**

Lodash is a widely used JavaScript utility library that provides a collection of helper functions and utilities for simplifying common programming tasks. It offers a wide range of functions that assist with data manipulation, array and object handling, string manipulation, functional programming, and more. However, using an outdated version of Lodash can introduce security vulnerabilities and potential weaknesses that may be exploited by attackers. Outdated versions of Lodash may contain known security vulnerabilities that have been addressed in newer releases. Attackers can exploit these vulnerabilities to execute arbitrary code, perform remote code execution, or compromise the integrity of the application.

- *Identified Version - 4.17.4*
- *Latest Version - 4.17.21*
- **Prototype Pollution ([CVE-2019-10744](#))**

Versions lower than 4.17.12 are vulnerable to Prototype Pollution. The function **defaultsDeep** can be tricked into adding or modifying properties of **Object.prototype** using a constructor payload.


CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H

---

IT21182600

Page | 1

### 3. Password Transmitted over HTTP




Report 03

URL: [datacamp.com](http://datacamp.com)

---

## *Password Transmitted over HTTP*

---

 **HIGH**

The vulnerability of transmitting password data over HTTP arises due to the lack of encryption in the communication channel. When a user enters their password on a website or any other online service, and that data is transmitted over an HTTP connection, it is sent in plain text format. This means that the password is not encrypted or protected in any way during transmission. The impact of this vulnerability is significant. If an attacker can intercept the network traffic between the user and the server, they can easily capture the password in clear text. This can be achieved through various means, such as eavesdropping on unsecured Wi-Fi networks, using network sniffing tools, or compromising network infrastructure.

Once an attacker obtains a user's password, they can potentially gain unauthorized access to the user's account. This can lead to various malicious activities, including identity theft, unauthorized transactions, unauthorized access to sensitive information, and even impersonation of the user. Transmitting password data over HTTP is highly insecure because any party with access to the network traffic can read and potentially abuse the information. It is important to note that even if the website or application hashes or encrypts the passwords when storing them, transmitting them over HTTP in plain text format exposes them to interception before they are protected.

URL : <http://www.datacamp.com/>

Input Name : `user[password]`

Form target action : <http://www.datacamp.com/users>

---

IT21182600

## 4. Active Mixed Content over HTTPS



Report 04

URL: [www.enviso.io](http://www.enviso.io)

---

### *Active Mixed Content over HTTPS*

---

 MEDIUM

Active Mixed Content over HTTPS vulnerability refers to a security issue where an HTTPS webpage contains active content, such as scripts or stylesheets, that are loaded over an insecure HTTP connection. This vulnerability occurs when there is a mix of secure (HTTPS) and insecure (HTTP) resources within the same webpage. When a webpage is loaded over HTTPS, it establishes a secure encrypted connection between the user's browser and the server, ensuring the confidentiality and integrity of the data exchanged. However, if the webpage includes active content retrieved through HTTP, it introduces a security gap in the encrypted connection.

An attacker positioned between the user and the server, known as a man-in-the-middle attacker, can exploit this vulnerability. They can intercept the request for the HTTP content and manipulate the response to include malicious code or scripts. This malicious active content can then execute within the user's browser, compromising the security of the page and the user's interaction with it.

#### Affected URLs:


- <https://www.enviso.io>
- <https://www.enviso.io/about/index.html>
- <https://www.enviso.io/attractions/index.html>
- <https://www.enviso.io/contact/index.html>
- <https://www.enviso.io/cookie-policy/index.html>
- <https://www.enviso.io/enviso-shop/index.html>
- <https://www.enviso.io/enviso-trade/index.html>

---

IT21182600

Page | 1

## 5. Cross-site Request Forgery




Report 05

URL: [www.enviso.io](http://www.enviso.io)

---

*Cross-site Request Forgery*

---

 MEDIUM

CSRF (Cross-Site Request Forgery) is a type of security vulnerability that allows an attacker to manipulate a victim into unknowingly performing malicious actions on a web application in which the victim is authenticated. The vulnerability occurs due to the web application's failure to adequately verify the source of incoming requests. In a CSRF attack, the attacker tricks the victim into visiting a malicious website or clicking on a malicious link. When the victim interacts with the attacker's website, their web browser automatically sends a request to the vulnerable web application, including any relevant authentication cookies.

The web application, assuming the request is legitimate, processes the request and performs the requested action on behalf of the victim. This can lead to unauthorized actions being executed, such as changing account settings, making fraudulent transactions, deleting data, or performing other actions available to the victim's authenticated session.

Affected URL : <https://www.enviso.io/contact/index.html>

### Affected Components


The affected components in a CSRF vulnerability typically include the web application's functionality and features that require user authentication and session management.

---

IT21182600

Page | 1

## 6. Out-of-date Version (Tomcat)


**SLIIT**  
 Discover Your Future


Report 06

URL: wiki.grab.com

---

### Out-of-date Version (Tomcat)

---

 **HIGH**

An out-of-date Tomcat vulnerability refers to a security weakness or flaw present in an older version of the Apache Tomcat server software. These vulnerabilities exist due to unpatched or outdated components within the Tomcat framework, making it susceptible to exploitation by attackers.

Running an out-of-date version of Tomcat exposes your web application to potential security risks. Attackers actively search for known vulnerabilities in older versions, which they can exploit to gain unauthorized access, compromise data integrity, or disrupt the normal functioning of the application.

**URL** : [https://wiki.grab.com/login.action?cs\\_destination=http://r87.com/n?%00.action&permissionViolation=true](https://wiki.grab.com/login.action?cs_destination=http://r87.com/n?%00.action&permissionViolation=true)

**Identified Version** : 9.0.65

**Latest Version** : 9.0.75 (in this branch)

**Overall Latest Version** : 10.1.9

**Vulnerability Database** : Result is based on 06/23/2023 20:30:00 vulnerability database content.

**Attack Pattern** : http%3a%2f%2fr87.com%2fn%3f%00.action

| Method | Parameter,          | Parameter Type, | Value                       |
|--------|---------------------|-----------------|-----------------------------|
| GET    | cs_destination      | Querystring     | http://r87.com/n?%00.action |
| GET    | permissionViolation | Querystring     | true                        |

---

IT21182600

Page | 1

## 7. Stack Trace Disclosure (Java)



Report 07

URL: [wiki.grab.com](http://wiki.grab.com)

### *Stack Trace Disclosure (Java)*

 MEDIUM

The Stack Trace Disclosure vulnerability refers to a security issue in Java-based web applications where sensitive information, specifically the stack trace, is exposed to potential attackers. When an application encounters an error or exception during its execution, it generates a stack trace, which contains a detailed record of the program's execution flow leading up to the error. However, in certain cases, this stack trace is inadvertently disclosed in the application's HTTP response, making it accessible to anyone who interacts with the application. This disclosure typically occurs when error messages or exception details are not properly handled or filtered in the code. The stack trace can contain critical information that can aid attackers in understanding the internal workings of the application and potentially identifying vulnerabilities.


Affected URL: [https://wiki.grab.com/login.action?os\\_destination=%22\\$\[\].%3E&permissionViolation=true](https://wiki.grab.com/login.action?os_destination=%22$[].%3E&permissionViolation=true)

| Method | Parameter,                          | Parameter Type,             | Value                     |
|--------|-------------------------------------|-----------------------------|---------------------------|
| GET    | <a href="#">os_destination</a>      | <a href="#">Querystring</a> | <code>**/\$[].&gt;</code> |
| GET    | <a href="#">permissionViolation</a> | <a href="#">Querystring</a> | true                      |

IT21182600

Page | 1

## 8. Weak Ciphers Enabled




Report 08

URL: [accounts.firefox.com](https://accounts.firefox.com)

---

### *Weak Ciphers Enabled*

---

 MEDIUM

The Weak Ciphers Enabled vulnerability refers to a security issue where insecure or weak encryption algorithms (ciphers) are enabled for secure communication (SSL) on a web server. SSL is a cryptographic protocol used to establish secure connections between clients (such as web browsers) and servers to protect the confidentiality and integrity of data transmitted over the network. In this vulnerability, the web server allows the use of weak ciphers, which are encryption algorithms that have known security weaknesses or are considered outdated. These weak ciphers lack the strength and robustness needed to provide adequate protection against modern cryptographic attacks.

**Affected URL:** <https://accounts.firefox.com/>

**List of Supported Weak Ciphers :**


- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x002F)
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0xC013)
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0xC014)
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 (0x003C)
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 (0x003D)
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 (0xC027)
- TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384 (0xC028)

---

IT21182600

Page | 1

## 9. Sub resource Integrity (SRI) Hash Invalid


**SLIIT**  
Discover Your Future


Report 09

URL: [accounts.firefox.com](https://accounts.firefox.com)

---

### *Sub resource Integrity (SRI) Hash Invalid*

---

 MEDIUM

The Subresource Integrity (SRI) Hash Invalid vulnerability refers to a security issue where the computed hash for a subresource is found to be invalid. SRI is a web security feature that allows web developers to ensure the integrity of externally hosted resources, such as scripts or stylesheets provided by third-party sources like Content Delivery Networks (CDNs). SRI protects against the possibility of these external resources being tampered with or modified during transit or by an attacker. It does this by comparing the expected hash value of the resource, as specified in the HTML element's integrity attribute, with the computed hash value of the fetched resource. If the hash values do not match, it indicates that the resource has been altered or tampered with, and SRI alerts the browser to block the resource from loading.

**Affected URL:** [https://accounts.firefox.com/complete\\_signin](https://accounts.firefox.com/complete_signin)

**Identified Sub Resource(s) :** <https://accounts-static.cdn.mozilla.net/styles/1cd4f689.tailwind.out.css>

### Affected Components

- ❖ **HTML Elements:** The vulnerability primarily affects HTML elements that reference external subresources, such as scripts or stylesheets. Specifically, the elements where the integrity attribute is used to declare the expected hash value of the subresource are susceptible to the SRI Hash Invalid vulnerability.

IT21182600

Page | 1



## 10. Session Cookie Not Marked as Secure

  
Report 10  
URL: [asupport.sulvo.com](https://asupport.sulvo.com)

---

### *Session Cookie Not Marked as Secure*

---

 **HIGH**

The Session Cookie Not Marked as Secure vulnerability refers to a security weakness where a session cookie is transmitted over an HTTPS connection but is not marked as secure. A session cookie is a small piece of data that is typically used to maintain session information and authenticate users on a website. When a session cookie is not marked as secure, it means that the cookie can be sent over an unencrypted HTTP connection instead of being restricted to secure HTTPS connections. This poses a significant security risk because it allows an attacker who can intercept the traffic, particularly through a man-in-the-middle attack, to potentially steal the cookie.

The impact of this vulnerability is severe. An attacker who successfully intercepts the traffic and gains access to the session cookie can hijack a victim's session. By hijacking the session, the attacker can impersonate the victim and gain unauthorized access to their account or perform malicious activities on the website.

Affected URL: [https://asupport.sulvo.com/process\\_login](https://asupport.sulvo.com/process_login)

Identified Cookie(s): JSESSIONID

Cookie Source: HTTP Header

---

IT21182600  
Page | 11

**Date: May 28, 2023**

Today marks the end of an eventful and rewarding journey that lasted for 17 days. I am thrilled to share that I have successfully achieved my goal of discovering the top 10 vulnerabilities in a website as part of my bug bounty activities. It has been an exhilarating and fulfilling experience, filled with challenges, learning opportunities, and moments of triumph.

Throughout this journey, I have dedicated countless hours to studying bug bounty tools, honing my skills, and expanding my knowledge in the realm of cybersecurity. I meticulously examined the website, scrutinizing its code, functionalities, and potential attack vectors. The journey was not without its obstacles, but I remained steadfast in my pursuit of uncovering vulnerabilities and making the internet a safer place.

Each day presented new challenges and opportunities for growth. I encountered a diverse range of vulnerabilities, from common misconfigurations to intricate security loopholes. The process involved thorough testing, leveraging various tools, and employing manual techniques to probe the website's defenses. With persistence and a systematic approach, I gradually unearthed vulnerabilities that could have exposed sensitive information or compromised the website's integrity.

I am proud to present the top 10 vulnerabilities I discovered during this journey:

1. **SQL Injection:** By manipulating input fields, I uncovered instances where the website was vulnerable to SQL injection attacks, potentially allowing unauthorized access to the underlying database.
2. **Cross-Site Scripting (XSS):** Through careful examination of user inputs, I identified areas where the website was susceptible to XSS attacks, which could have enabled malicious actors to inject and execute malicious scripts.
3. **Cross-Site Request Forgery (CSRF):** I detected vulnerabilities that could have allowed attackers to exploit the trust of authenticated users and perform unauthorized actions on their behalf.
4. **Server-Side Request Forgery (SSRF):** Through targeted testing, I discovered instances where the website could be manipulated to make unintended requests to internal resources or external services.

5. Remote Code Execution (RCE): By leveraging certain input fields, I exposed vulnerabilities that could have allowed attackers to execute arbitrary code on the server, potentially compromising the entire system.
6. Information Disclosure: Through careful analysis of error messages and response headers, I identified instances where sensitive information, such as server configurations or file paths, was inadvertently disclosed.
7. Insecure Direct Object References (IDOR): By manipulating object references, I uncovered instances where the website allowed unauthorized access to restricted resources or sensitive data.
8. XML External Entity (XXE) Injection: I identified vulnerabilities that could have allowed attackers to exploit XML parsers and retrieve sensitive information or execute remote requests.
9. Authentication and Session Management Issues: Through a combination of manual testing and automated tools, I uncovered weaknesses in authentication mechanisms, such as weak password policies, session fixation, or insufficient session timeouts.
10. Server-Side Template Injection (SSTI): By manipulating user inputs, I discovered vulnerabilities that could have enabled attackers to inject malicious code into server-side templates, potentially leading to code execution or data exposure.

These discoveries represent a significant milestone in my bug bounty journey. By responsibly disclosing these vulnerabilities to the website's owners, I have played a role in securing their infrastructure and safeguarding their users' data. This experience has reinforced my passion for cybersecurity and the importance of ongoing vigilance in the face of evolving threats. I am immensely grateful for the lessons learned, the challenges overcome, and the knowledge gained throughout this journey. It is a testament to the power of perseverance, continuous learning, and the thrill of contributing to a more secure digital landscape.

As I reflect on these 17 days, I am reminded of the immense value bug bounty programs bring to the cybersecurity community. They foster collaboration, innovation, and a shared commitment to fortify our digital world against malicious actors. With this milestone achieved, I now set my sights on new horizons, eager to explore further depths of cybersecurity, enhance my expertise, and continue making valuable contributions to the ever-evolving field.

Here's to the completion of a remarkable journey and the beginning of new adventures in the realm of bug bounty hunting.

## Conclusion

In conclusion, my bug bounty journey has been an incredible learning experience, filled with challenges, triumphs, and an unwavering commitment to enhancing the security of web applications. It all began with my study of the OWASP Top 10 vulnerabilities, which provided a solid foundation for understanding the most common security risks faced by web applications today.

Discovering the bug bounty program ignited my passion for cybersecurity and opened up a world of opportunities to contribute to the enhancement of security. Registering on the bug bounty program site marked a pivotal moment in my journey, as it allowed me to actively participate in identifying vulnerabilities and making a positive impact on web application security. With a thirst for knowledge, I dedicated time to studying various bug bounty tools, equipping myself with the necessary skills to conduct effective security assessments. Armed with this newfound knowledge, I eagerly began my bug bounty program, meticulously scrutinizing web applications and systems for vulnerabilities.

Through perseverance and determination, I successfully reported ten distinct vulnerabilities, addressing critical security flaws and potential data breaches. Each vulnerability report carried valuable insights, which not only aided in the remediation process but also contributed to the overall enhancement of web application security.

In addition to the vulnerability reports, I also maintained a detailed journal, documenting my experiences, challenges faced, and the lessons learned throughout the bug bounty program. This journal serves as a valuable resource for future reference and showcases my growth as a bug bounty hunter.

Overall, this bug bounty journey has been transformative, propelling me into the realm of cybersecurity and deepening my understanding of web application vulnerabilities. It has reinforced the importance of continuous learning, adaptability, and persistence in the face of challenges. As I conclude this chapter, I am filled with a sense of fulfillment and accomplishment. The knowledge gained, skills honed, and contributions made towards enhancing security have set a solid foundation for my future endeavors in the field of cybersecurity. I am excited to continue my bug bounty journey, pushing boundaries, and making a lasting impact on the security of web applications.

---

**END**

---