Cybersecurity

Penetration Test Report

Rekall Corporation

Penetration Test Report

GoodCorp, LLC

Confidentiality Statement

This document contains confidential and privileged information from Rekall Inc. (henceforth known as Rekall). The information contained in this document is confidential and may constitute inside or non-public information under international, federal, or state laws. Unauthorized forwarding, printing, copying, distribution, or use of such information is strictly prohibited and may be unlawful. If you are not the intended recipient, be aware that any disclosure, copying, or distribution of this document or its parts is prohibited.

Table of Contents

Confidentiality Statement 2
Contact Information 4
Document History 4
Introduction 5
  Assessment Objective 5
Penetration Testing Methodology 6
  Reconnaissance 6
  Identification of Vulnerabilities and Services 6
  Vulnerability Exploitation 6
  Reporting 6
Scope 7
Executive Summary of Findings 8
  Grading Methodology 8
  Summary of Strengths 9
  Summary of Weaknesses 9
Executive Summary Narrative 10
Summary Vulnerability Overview 11
Vulnerability Findings 13

Contact Information

Company Name	GoodCorp, LLC
Contact Name	Anthony Cirillo
Contact Title	Penetration Tester

Document History

Version	Date	Author(s)	Comments
001	04/04/2023	Anthony Cirillo	First draft
002	11/04/2023	Anthony Cirillo	Second draft
003	16/04/2023	Anthony Cirillo	Final

Introduction

In accordance with Rekall policies, our organization conducts external and internal penetration tests of its networks and systems throughout the year. The purpose of this engagement was to assess the networks' and systems' security and identify potential security flaws by utilizing industry-accepted testing methodology and best practices.

For the testing, we focused on the following:

- Attempting to determine what system-level vulnerabilities could be discovered and exploited with no prior knowledge of the environment or notification to administrators.
- Attempting to exploit vulnerabilities found and access confidential information that may be stored on systems.
- Documenting and reporting on all findings.

All tests took into consideration the actual business processes implemented by the systems and their potential threats; therefore, the results of this assessment reflect a realistic picture of the actual exposure levels to online hackers. This document contains the results of that assessment.

Assessment Objective

The primary goal of this assessment was to provide an analysis of security flaws present in Rekall's web applications, networks, and systems. This assessment was conducted to identify exploitable vulnerabilities and provide actionable recommendations on how to remediate the vulnerabilities to provide a greater level of security for the environment.

We used our proven vulnerability testing methodology to assess all relevant web applications, networks, and systems in scope.

Rekall has outlined the following objectives:

Table 1: Defined Objectives

Objective
Find and exfiltrate any sensitive information within the domain.
Escalate privileges.
Compromise several machines.

Penetration Testing Methodology

Reconnaissance

We begin assessments by checking for any passive (open source) data that may assist the assessors with their tasks. If internal, the assessment team will perform active recon using tools such as Nmap and Bloodhound.

Identification of Vulnerabilities and Services

We use custom, private, and public tools such as Metasploit, hashcat, and Nmap to gain perspective of the network security from a hacker's point of view. These methods provide Rekall with an understanding of the risks that threaten its information, and also the strengths and weaknesses of the current controls protecting those systems. The results were achieved by mapping the network architecture, identifying hosts and services, enumerating network and system-level vulnerabilities, attempting to discover unexpected hosts within the environment, and eliminating false positives that might have arisen from scanning.

Vulnerability Exploitation

Our normal process is to both manually test each identified vulnerability and use automated tools to exploit these issues. Exploitation of a vulnerability is defined as any action we perform that gives us unauthorized access to the system or the sensitive data.

Reporting

Once exploitation is completed and the assessors have completed their objectives, or have done everything possible within the allotted time, the assessment team writes the report, which is the final deliverable to the customer.

Scope

Prior to any assessment activities, Rekall and the assessment team will identify targeted systems with a defined range or list of network IP addresses. The assessment team will work directly with the Rekall POC to determine which network ranges are in-scope for the scheduled assessment.

It is Rekall's responsibility to ensure that IP addresses identified as in-scope are actually controlled by Rekall and are hosted in Rekall-owned facilities (i.e., are not hosted by an external organization). In-scope and excluded IP addresses and ranges are listed below.

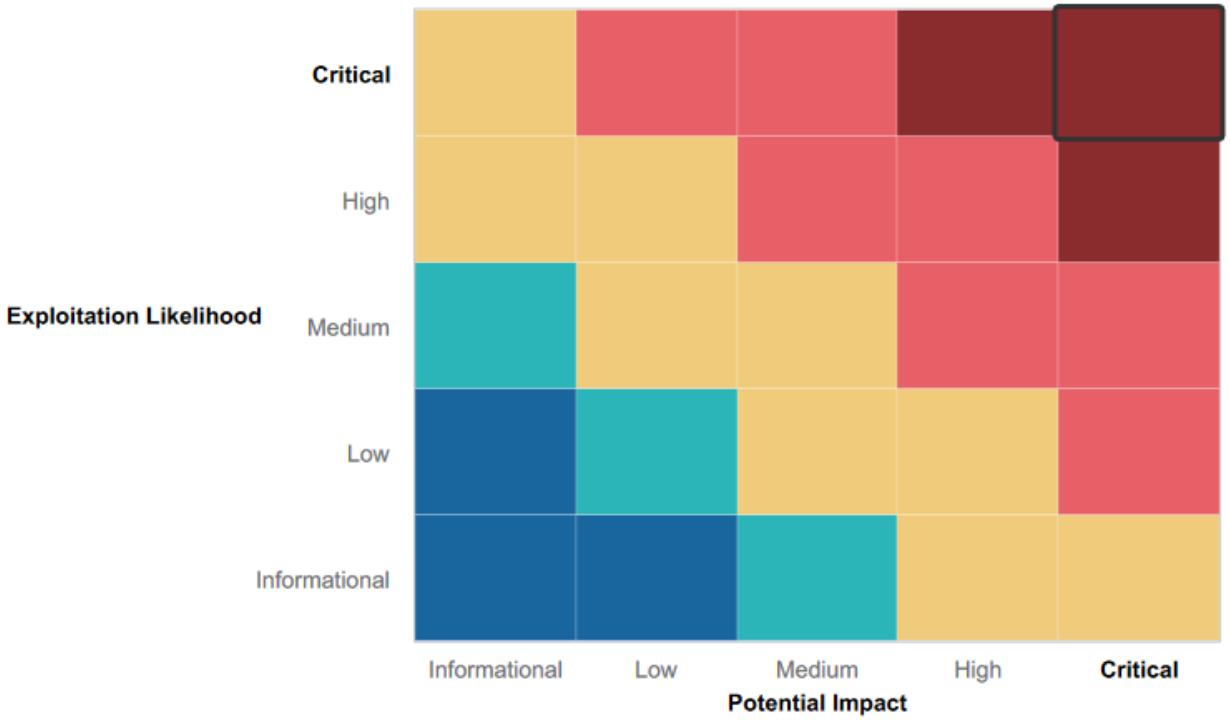
Executive Summary of Findings

Grading Methodology

Each finding was classified according to its severity, reflecting the risk each such vulnerability may pose to the business processes implemented by the application, based on the following criteria:

Critical: Immediate threat to key business processes.
High: Indirect threat to key business processes/threat to secondary business processes.
Medium: Indirect or partial threat to business processes.
Low: No direct threat exists; vulnerability may be leveraged with other vulnerabilities.
Informational: No threat; however, it is data that may be used in a future attack.

As the following grid shows, each threat is assessed in terms of both its potential impact on the business and the likelihood of exploitation:



Summary of Strengths

While the assessment team was successful in finding several vulnerabilities, the team also recognized several strengths within Rekall's environment. These positives highlight the effective countermeasures and defenses that successfully prevented, detected, or denied an attack technique or tactic from occurring.

- The company has not exposed highly sensitive data on publicly accessible servers or directories, indicating a level of caution in their security posture.
- The implementation of network segmentation or access control measures is evident as lateral movement was required to move between different parts of the network, which is a positive sign for network security.
- The fact that some form of password hashing was in place, as evidenced by the need to attempt to crack the NTDS.dit file, indicates a basic level of password security.
- The company's engagement in a pen test shows a commitment and desire to have a secure network, which is an encouraging sign for overall security posture.
- There were attempts to secure the web app and servers with procedures in place, such as limiting access to some directories only to root users.

Summary of Weaknesses

We successfully found several critical vulnerabilities that should be immediately addressed in order to prevent an adversary from compromising the network. These findings are not specific to a software version but are more general and systemic vulnerabilities.

Web Application Security:

- Several Cross-Site Scripting (XSS) vulnerabilities found on different pages
- Local File Inclusion (LFI) vulnerability on the memory-planner.php page
- Sensitive data exposure vulnerability
- Login credentials embedded in the page's source code

Network Security:

- Critical vulnerability related to the version of Apache Struts running on a host
- Exploitation of Tomcat Remote Code Execution (RCE) and Apache mod_cgi Bash environment variable code injection (Shellshock)
- Exploitation of a Drupal Web Services RCE module
- Password guessing techniques to access a host
- Open FTP port allowing anonymous login
- Vulnerable SLmail smtpd running on an open SMTP port

Privilege Escalation & Lateral Movement:

- CVE-2019-14287 vulnerability used to escalate privileges to root
- Cracking of password hashes using John software
- Metasploit windows smb psexec module used to move laterally into the domain controller
- kiwi tool could be used to retrieve the NTLM hash of the Administrator

Overall, these weaknesses highlight the importance of securing web applications, conducting regular vulnerability assessments on network infrastructure, and implementing strong access controls and privilege escalation prevention measures.

Executive Summary

During security testing, we discovered several Cross-Site Scripting (XSS) vulnerabilities on the website. A reflected XSS vulnerability was found in the "Put your name here" field on the welcome.php page. An XSS vulnerability was also found on the memory-planner.php page, but input validation prevented our initial payload from working. We bypassed this protection by using several filter evasion techniques. Additionally, we found a stored XSS vulnerability on the comments.php page. We identified a Local File Inclusion (LFI) vulnerability on the memory-planner.php page, and successfully executed a file containing malicious PHP code using a double extension. We discovered a sensitive data exposure vulnerability and were able to find sensitive data in the response headers by using FoxyProxy with Burp Suite. We utilized an SQL query to bypass the login screen on the login.php page, and also discovered login credentials embedded in the page's source code.

We accessed the robots.txt file and discovered a new page, souvenirs.php, that was not previously known. We also found a command injection vulnerability on the admin networking tools page, which allowed us to list the available files and retrieve the contents of the "vendors.txt" file. After conducting a brute force attack on the account of user Melina, we gained access to a page containing top secret legal data. We then used Burp Suite to try various session IDs to obtain access to a locked page, eventually finding session 87 to reveal the data.

Our initial Nmap ping sweep scan revealed five online hosts. During our basic network scan using Nessus, we identified a critical vulnerability on host 192.168.13.12, related to the version of Apache Struts running on the host. The vulnerability allowed us to execute remote code on the host using an apache struts Jakarta multipart parser OGNL injection module on Metasploit. We were able to extract valuable information from the host, including a noteworthy 7z file located in the root folder. During our aggressive Nmap scan, we discovered that IP address 192.168.13.10 had two open ports running Apache JServ and Apache Tomcat/Coyote JSP engine 1.1. We exploited a Tomcat Remote Code Execution (RCE) via JSP Upload Bypass module on Metasploit to gain access to the host and obtain valuable information.We also discovered an open port 80 on IP address 192.168.13.11 running Apache HTTPS 2.4.7, which we exploited using an Apache mod_cgi Bash environment variable code injection (Shellshock) module. The successful exploitation provided us access to the server's data, including the sudoers and passwd file.

On 192.168.13.13, we discovered an open port 80 running Drupal 8, which we exploited using a Drupal Web Services RCE module on Metasploit. We gained access to the host and gathered valuable insights. Then, on 192.168.13.14, we leveraged password guessing techniques to successfully access the host as Alice with the credentials 'alice:alice'. We escalated privileges to root using the CVE-2019-14287 vulnerability, which allowed us to gain full access to the host and uncover critical information.

We found a sensitive file containing a username and password hash in the GitHub repository belonging to totalrekall. We were able to decipher the password using the John software, gaining access to the 'trivera' account. With these login credentials, we successfully logged into a website on the internal network and gained access to confidential information.We also performed an aggressive nmap scan, which revealed an open FTP port on IP address 172.22.117.20 that allowed anonymous login. We used the 'ftp' command to extract a file from the server to our local system, which we were then able to read. Further scanning of the Windows network led to the discovery of an open SMTP port on the same IP address, which had a vulnerable SLmail smtpd running. We used a relevant module on Metasploit to exploit this vulnerability, providing us with a meterpreter session for the host. This enabled us to gain unauthorized access to sensitive data, including the mscash2 hash of a user named 'ADMBob,' which we cracked using John to reveal their password. We then leveraged our newly found credentials to move laterally into the domain controller using a Metasploit windows smb psexec module. In addition, we utilized kiwi in our meterpreter session on the domain controller with dcsync_ntlm to retrieve the NTLM hash of the Administrator, thereby officially compromising the admin.

Summary Vulnerability Overview

Vulnerability	Severity
Multiple instances of XSS vulnerabilities discovered on website	High
LFI and File Upload Vulnerability with Input Validation Bypass	Medium
Sensitive Data Exposure through HTTP Traffic Interception	High
SQL injection vulnerability on login page allowing bypass of authentication	High
Sensitive Credentials Disclosure via HTML Source Code on Login Page	High
Sensitive Data Exposure via Web Application's Robots.txt File	Low
Command Injection Vulnerability in Admin Networking Tools Page	Critical
Brute Force Attacks on Web Application Login Credentials/Session Management	High
PHP Injection Vulnerability in Web Application	Medium
Directory Traversal Vulnerability in Web Application	High
Network Scanning Reveals Exposed Hosts	Low
Remote Code Execution Vulnerability in Apache Struts Jakarta Multipart Parser	Critical
Tomcat RCE via JSP Upload Bypass vulnerability	High
Apache Bash Environment Variable Code Injection (Shellshock) Vulnerability	High
Remote Code Execution (RCE) via Drupal Web Services	High
Privilege Escalation via Sudo Vulnerability (CVE-2019-14287)	Critical
Potential Data Breach due to Sensitive Information on GitHub Repository	High
Unsecured Services and Weak Authentication Lead to Unauthorized Access	High
Unsecured FTP Service with Anonymous Login Allows Unauthorized Access	Medium
Unauthorized Access to Sensitive Files through SLmail SMTPd Vulnerability	High
Weak Passwords and Lateral Movement, allowing for DC compromise	Critical

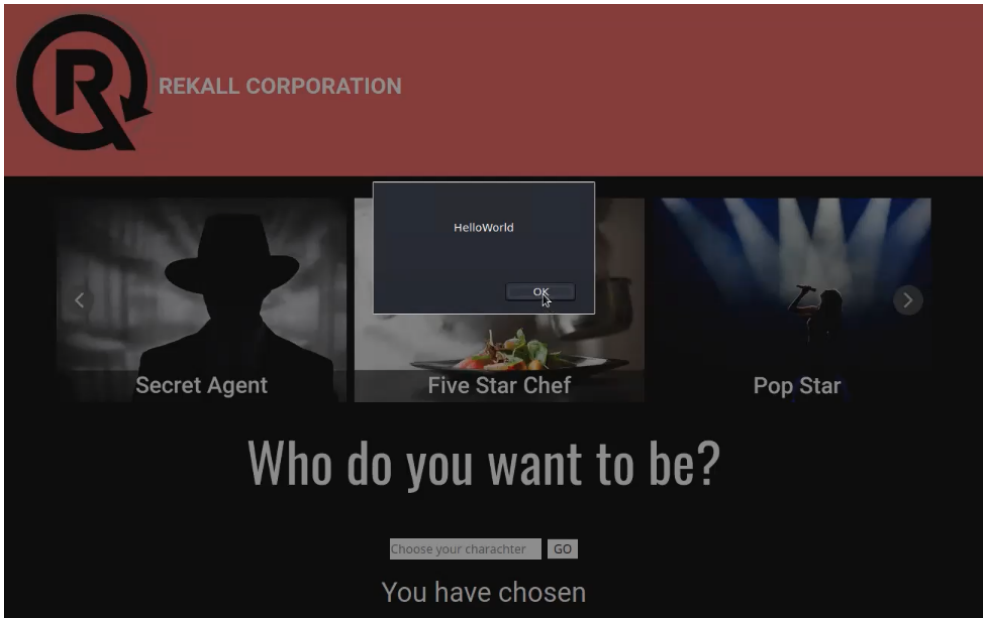
The following summary tables represent an overview of the assessment findings for this penetration test:

Scan Type	Total
Hosts	192.168.14.35 192.168.13.10 192.168.13.11 192.168.13.12 192.168.13.13 192.168.13.14 172.22.117.10 172.22.117.20
Ports	Port 22 (SSH) Port 80 (HTTP) Port 443 (HTTPS) Port 25 (SMTP) Port 21 (FTP) Port 8009 (HTTP) Port 8080 (HTTP)

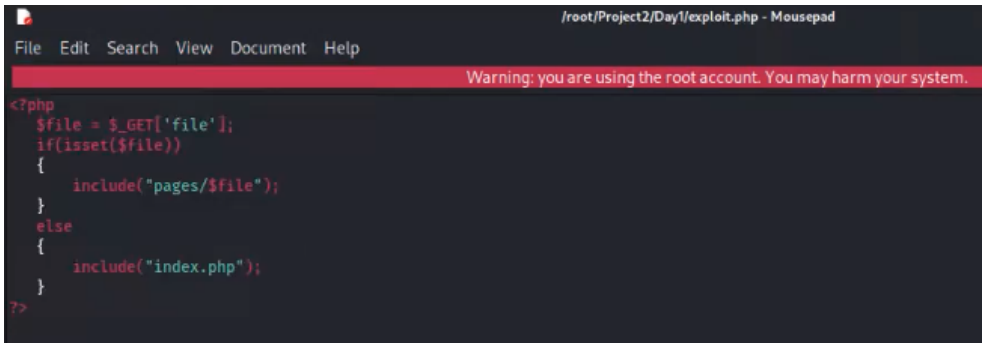
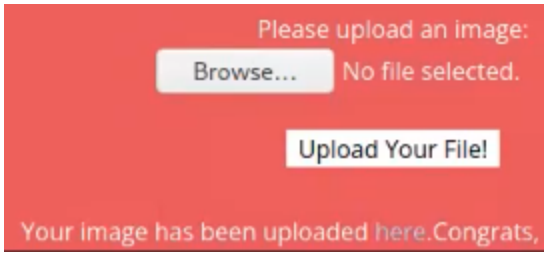
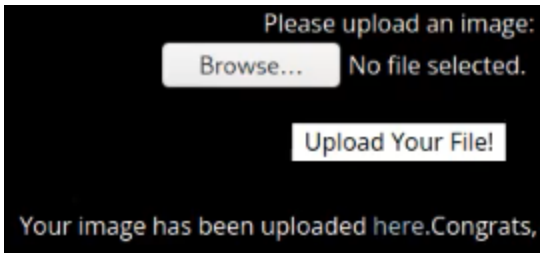
Exploitation Risk	Total
Critical	4
High	12
Medium	3
Low	2

Vulnerability Findings

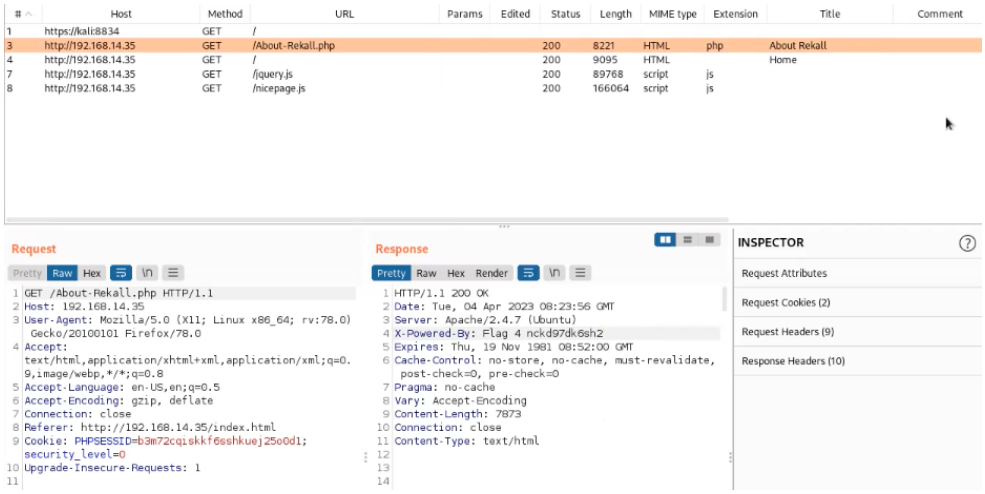
Vulnerability 1	Findings
Title	Multiple instances of XSS vulnerabilities discovered on website
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**High**
Description	During our security testing, we discovered several instances of Cross-Site Scripting (XSS) vulnerabilities on the website. We were able to exploit these vulnerabilities to inject malicious code and trigger pop-ups on certain pages. On the welcome.php page, we found a reflected XSS vulnerability in the "Put your name here" field. By entering a payload of "<script>alert('HelloWorld')</script>", we were able to trigger a pop-up. We also found an XSS vulnerability on the memory-planner.php page, specifically in the "Choose your character" field. However, we found that input validation was in place and our original payload did not work. To bypass this protection, we used several XSS filter evasion techniques, such as entering "<SCRIPscriptT>alert('HelloWorld')</SCRIPscriptT>", which successfully triggered a pop-up. Finally, we discovered a stored XSS vulnerability on the comments.php page. By entering a payload of "<script>alert('imhacking')</script>", we were able to store the malicious code and trigger a pop-up whenever someone views the comments section.

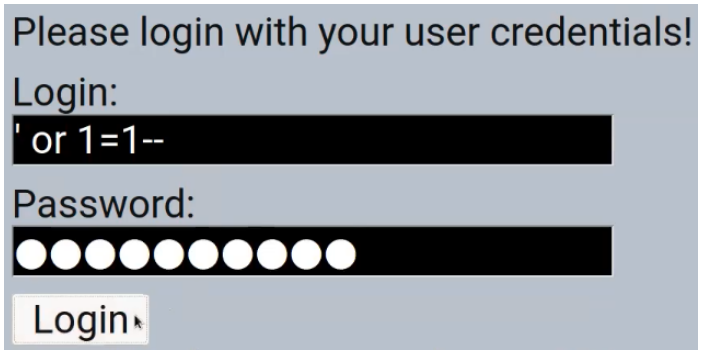
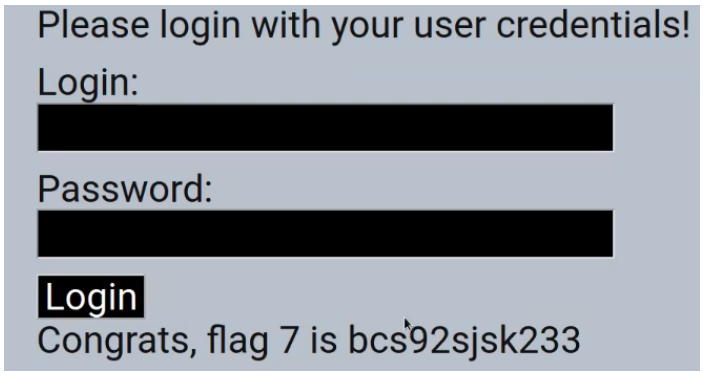
Images	 

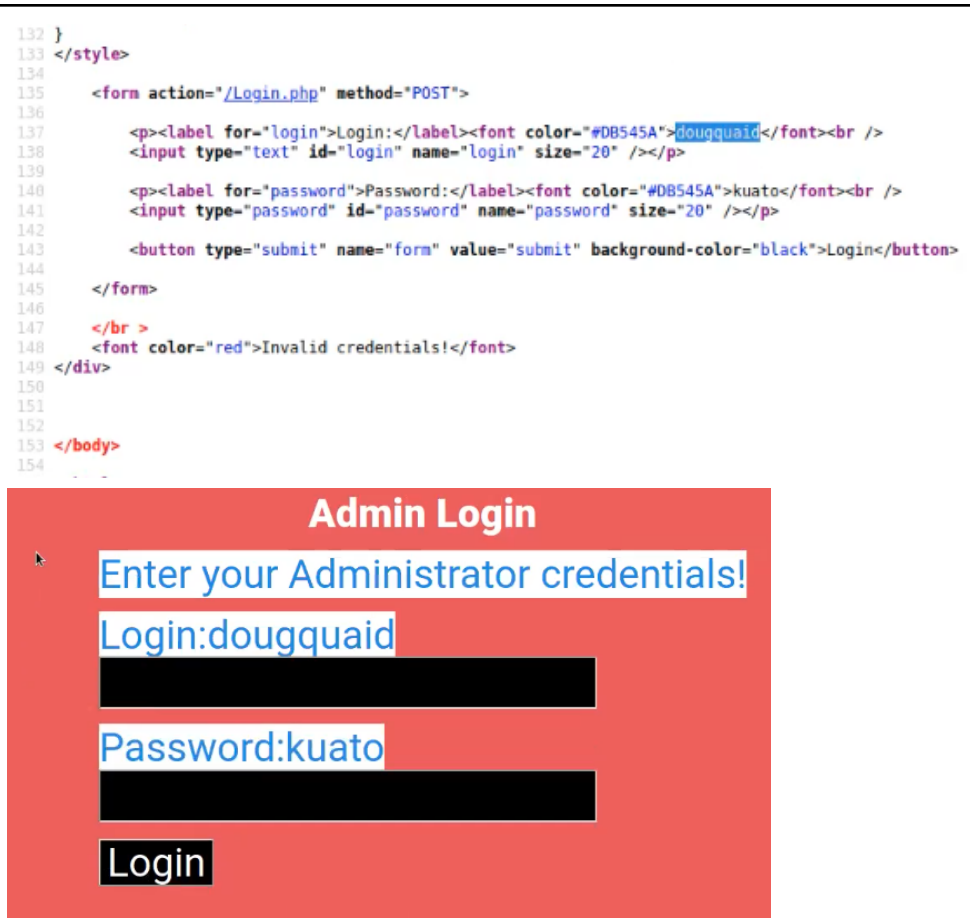
	
Affected Hosts	192.168.14.35
Remediation	1. Input Validation: Implement input validation on all user input fields, including client-side and server-side validation, to ensure that only expected data is accepted and processed. 2. Sanitization: Implement a sanitization process that cleanses any user-supplied data of potentially dangerous characters or scripts. This process can include encoding or stripping special characters and tags. 3. Content Security Policy (CSP): Implement a CSP that restricts the types of content that can be loaded on the website, such as scripts, images, and stylesheets, to prevent unauthorized execution of malicious code. 4. Output Encoding: Implement output encoding to encode all user-supplied data that is displayed on the website to prevent malicious scripts from executing. 5. Limiting User Input: Implement a limit on the length of input fields, to ensure that users cannot inject excessive amounts of data and potentially harmful scripts.

Vulnerability 2	Findings
Title	LFI and File Upload Vulnerability with Input Validation Bypass
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**Medium**
Description	During testing on the memory-planner.php page, a Local File Inclusion (LFI) vulnerability was discovered in the second input field. A file containing malicious PHP code was created using Mousepad and uploaded through the web application's file upload feature. The upload was successful, allowing the file to be executed on the server. However, in the other input field, input validation prevented the same file from being uploaded (only jpg files are allowed). To bypass this protection, a double extension "exploit.jpg.php" was used, which allowed the file to be uploaded and executed on the server. This technique was successful in exploiting the vulnerability and executing the malicious code.
Images	  
Affected Hosts	192.168.14.35
Remediation	1. Restrict file uploads: If file uploads are not essential, consider disabling them altogether. Otherwise, restrict file uploads to only trusted users and file types. This can help prevent malicious files from being uploaded to the local filesystem and executed on the server.

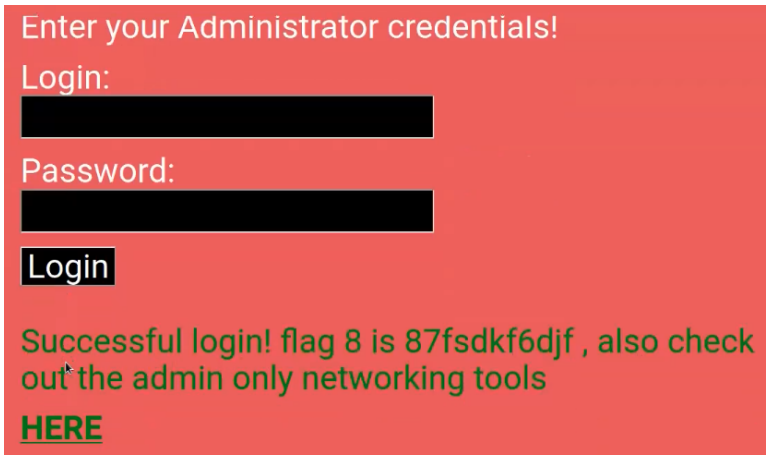
	2. Use allow listing: If file uploads are required, use allow listing to restrict the types of files that can be uploaded. Only allow the upload of specific file types, such as images or documents, and reject all others. 3. Implement input validation: Thoroughly validate user input, including file names and types, to ensure that they conform to expected formats and limits. Reject any input that does not meet the criteria to prevent unexpected behaviour. 4. Sanitize user input: Sanitize user input to remove any special characters or tags that could be used to inject malicious code into the application. 5. Apply file system permissions: Set appropriate file system permissions to restrict access to sensitive files and directories, and prevent the execution of arbitrary code. 6. Use Content Security Policy (CSP): Implement a CSP that restricts the types of content that can be loaded on the website, such as scripts, images, and stylesheets, to prevent unauthorized execution of malicious code.

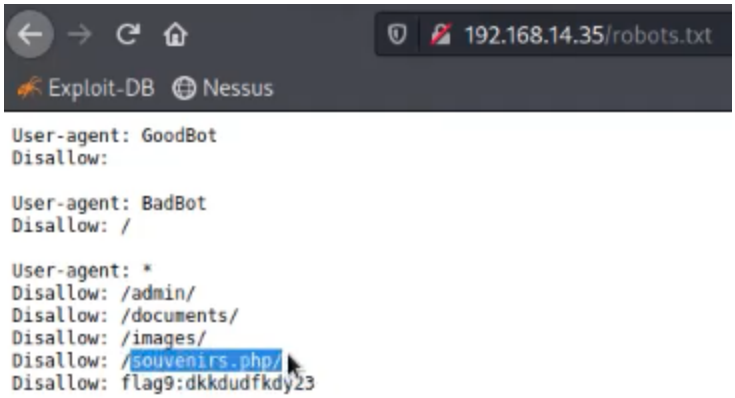
Vulnerability 3	Findings
Title	Sensitive Data Exposure through HTTP Traffic Interception
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**High**
Description	During a security assessment, we discovered a vulnerability related to sensitive data exposure. To exploit this vulnerability, we downloaded and configured Foxy Proxy on the browser to work with Burp Suite. After turning on Burp Suite, we intercepted HTTP traffic and navigated through the website to identify any sensitive data within the traffic history. As a result of our investigation, we were able to find sensitive data in the response headers.
Images	
Affected Hosts	192.168.14.35
Remediation	1. Implement encryption for sensitive data: Implementing encryption for sensitive data can prevent it from being exposed even if an attacker intercepts the traffic. 2. Use HTTPS: Using HTTPS instead of HTTP can help prevent sensitive data from being exposed through network traffic interception. 3. Implement access controls: Implementing access controls such as authentication and authorization can help prevent unauthorized access to sensitive data.

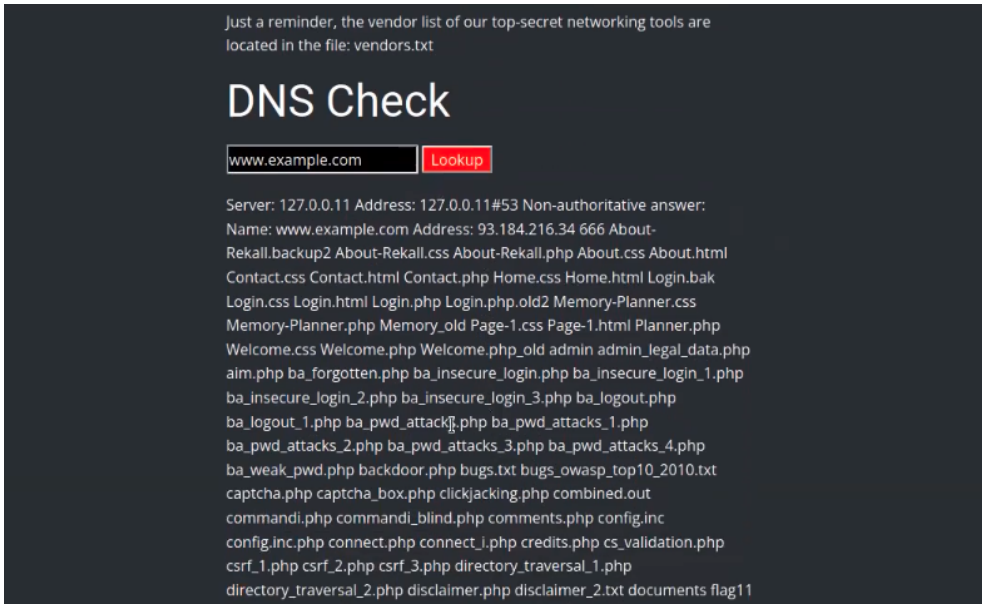
Vulnerability 4	Findings
Title	SQL injection vulnerability on login page allowing bypass of authentication
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	High
Description	During our security assessment, we discovered an SQL injection vulnerability on the login.php page. To bypass the login screen, we utilized an SQL query by entering " ' or 1=1 -- ". The double hyphen used at the end of the query indicates to ignore the rest of the text on that line.
Images	 
Affected Hosts	192.168.14.35
Remediation	1. Implementing server-side input validation: This involves validating input data on the server-side to ensure that it meets the expected format and type. 2. Using prepared statements: This involves using parameterized queries instead of concatenating user input with SQL queries. 3. Implementing client-side input validation: This involves validating input data on the client-side using JavaScript or other scripting languages.

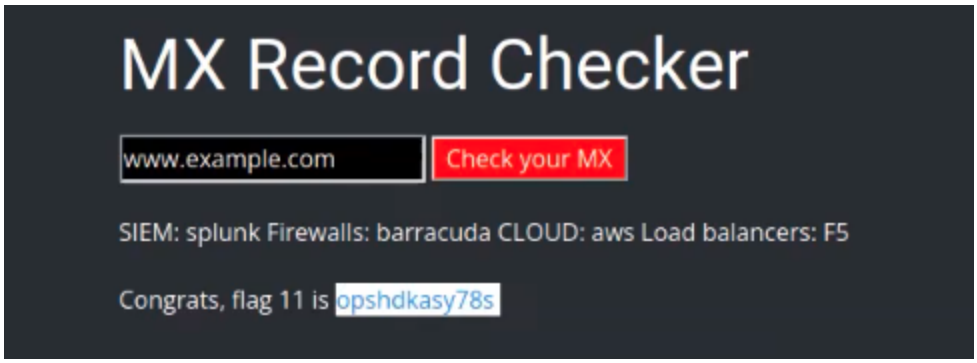
Vulnerability 5	Findings
Title	Sensitive Credentials Disclosure via HTML Source Code on Login Page
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**High**
Description	During our assessment of the login.php page, we decided to view its HTML source code to gain further insights. Upon examining the source code, we discovered user login credentials embedded in it. We observed that the credentials were set to the same colour as the page's background, making them only visible when highlighted. These credentials were identified as "dougquaid:kuato". With this information, we were able to gain access to an admin-only networking tools page using the discovered login credentials.
Images	

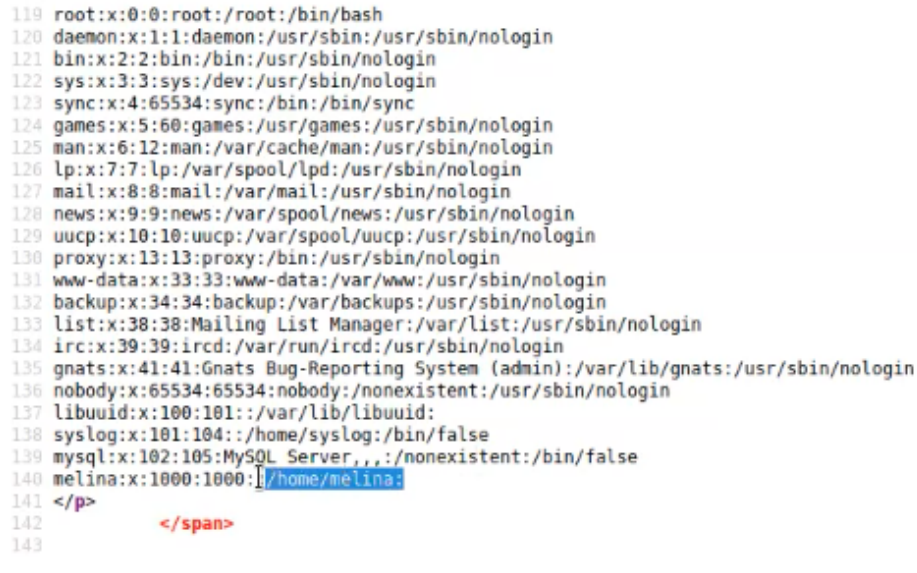
```
132 }
133 </style>
134
135     <form action="/Login.php" method="POST">
136
137         <p><label for="login">Login:</label><font color="#DB545A">dougquaid</font><br />
138         <input type="text" id="login" name="login" size="20" /></p>
139
140         <p><label for="password">Password:</label><font color="#DB545A">kuato</font><br />
141         <input type="password" id="password" name="password" size="20" /></p>
142
143         <button type="submit" name="form" value="submit" background-color="black">Login</button>
144
145     </form>
146
147     </br >
148     <font color="red">Invalid credentials!</font>
149 </div>
150
151
152
153 </body>
154
```

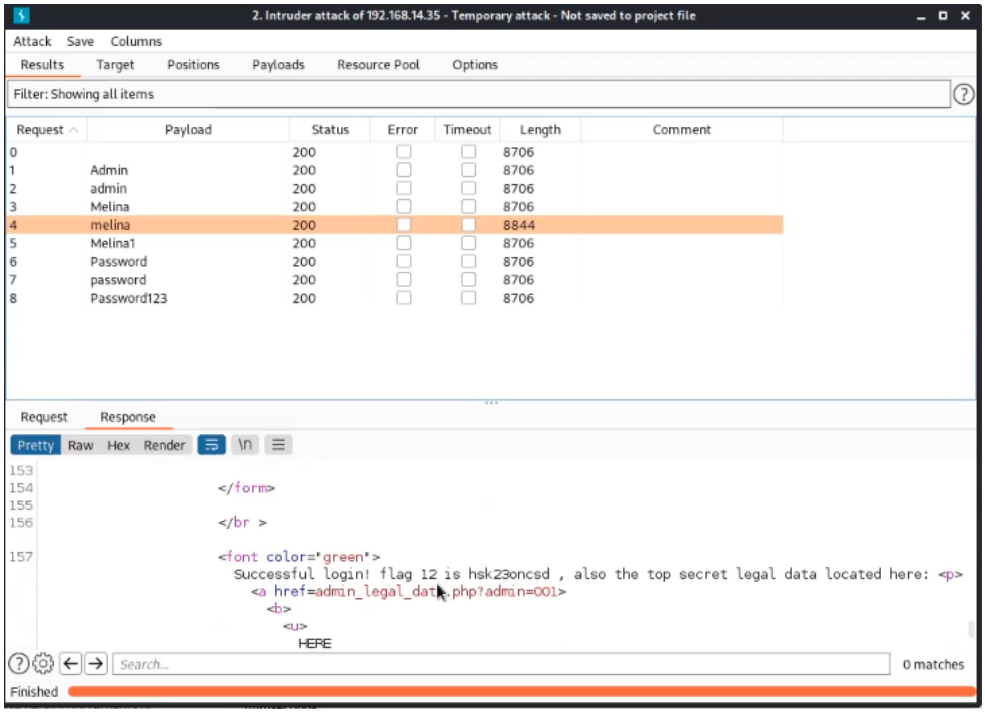
	
Affected Hosts	192.168.14.35
Remediation	1. Avoid embedding sensitive information such as usernames, passwords, or API keys in the HTML source code of web pages. 2. Store sensitive data securely on the server-side and use server-side scripting to retrieve and display the data as needed. 3. Use secure communication protocols such as HTTPS to encrypt sensitive data during transmission between the client and server. 4. Implement user authentication mechanisms such as multi-factor authentication to prevent unauthorized access to sensitive data. 5. Educate developers and administrators on secure coding practices

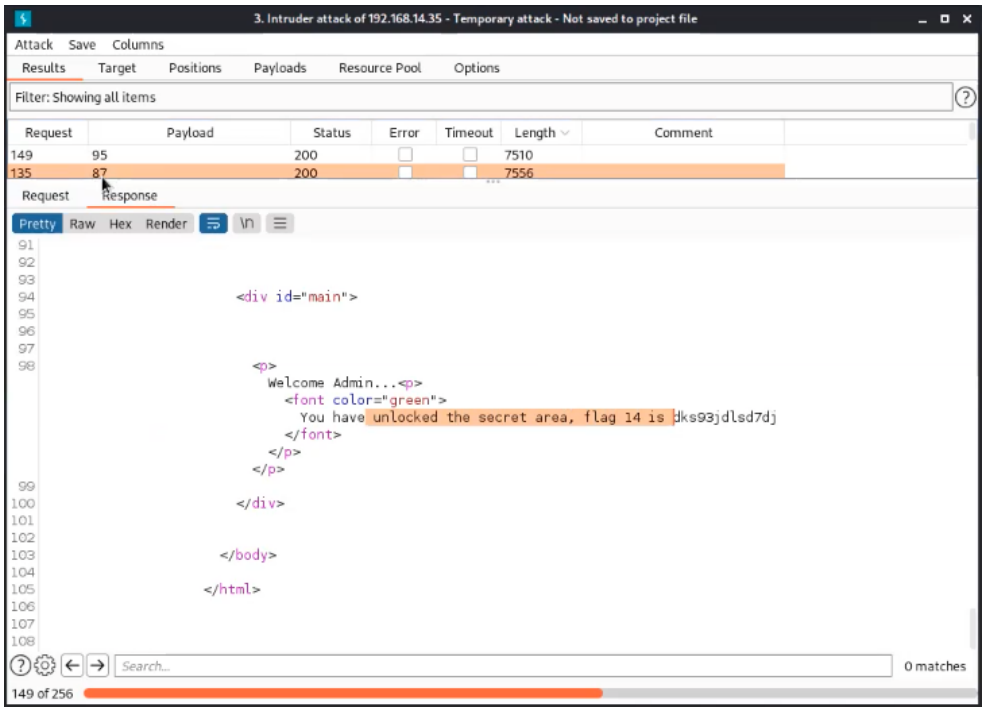
Vulnerability 6	Findings
Title	Sensitive Data Exposure via Web Application's Robots.txt File
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**Low**
Description	During our assessment, we identified a potential vulnerability of sensitive data exposure, which was related to the standard used to communicate with web crawlers and web robots. To investigate this, we entered "/robots.txt" in the URL to access the file. This file revealed pages of the web application that should be excluded from being crawled, including a new page called "souvenirs.php" that we had not previously encountered. This discovery raised the possibility that there may be sensitive data on the "souvenirs.php" page that was not intended to be publicly accessible.
Images	
Affected Hosts	192.168.14.35
Remediation	1. Review and update the robots.txt file regularly to ensure that it only includes pages that should be publicly accessible and exclude those that should not. 2. Avoid including sensitive information in the robots.txt file, such as login pages, confidential directories, or private information.

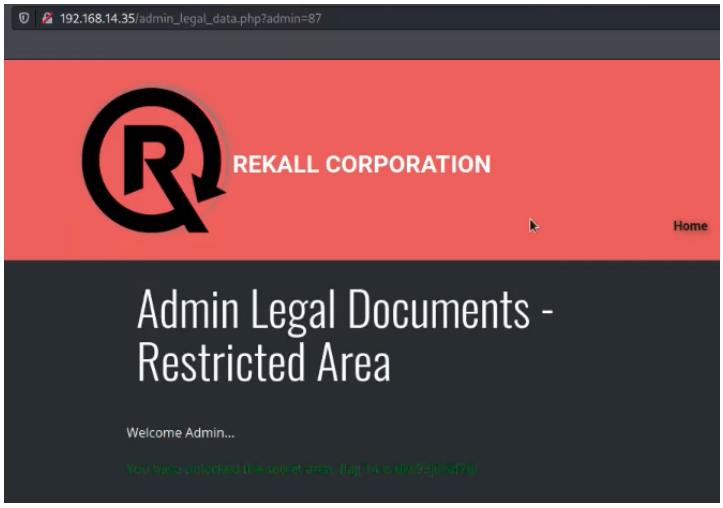
Vulnerability 7	Findings
Title	Command Injection Vulnerability in Admin Networking Tools Page
Type (Web app / Linux OS / Wlndows OS)	Web App
Risk Rating	**Critical**
Description	During our assessment, we accessed the admin networking tools page and identified a command injection vulnerability in the first field designed for DNS checking. After discovering that the "top secret networking tools" were located in the file "vendors.txt", as noted on the page, we tested the vulnerability by submitting "www.example.com; ls" in the field. This command listed the files and directories available on the server. We then submitted "www.example.com; cat vendors.txt" to retrieve the contents of the "vendors.txt" file. In addition, we attempted the same command injection on the second field on the page, which was designed for checking MX records. While the same command using the semicolon did not work, we were able to exploit the vulnerability using a pipe symbol instead, submitting "www.example.com | cat vendors.txt" to retrieve the contents of the file. Overall, these findings highlight the importance of properly sanitizing user input and implementing input validation to prevent command injection attacks.
Images	

	
Affected Hosts	192.168.14.35
Remediation	1. DNS and MX record checking fields should be designed to accept only valid input formats for their respective records, such as IP addresses, domain names, and hostnames. 2. Sanitize user input by removing or encoding special characters such as semicolons, pipes, and quotes that can be used to inject commands into the server. 3. Implement input validation to ensure that user input is properly formatted and meets specific criteria, such as length and character restrictions, to avoid any input that is not valid.

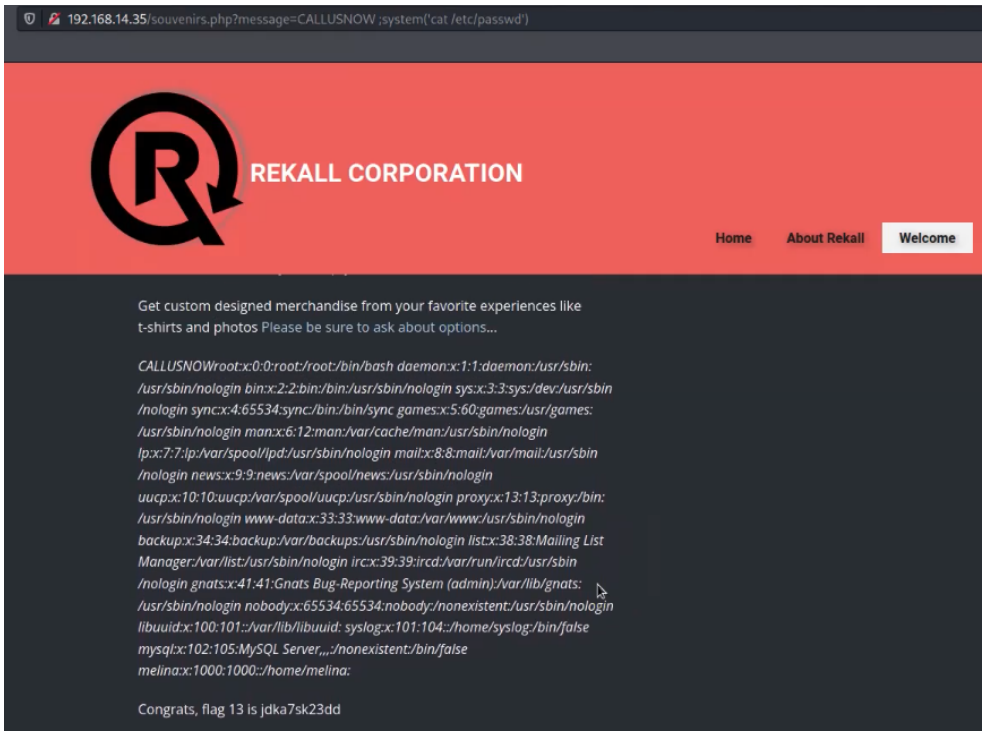
Vulnerability 8	Findings
Title	Brute Force Attacks on Web Application Login Credentials/Session Management
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**High**
Description	During a previous command injection attack, we were able to gain access to the passwd directory on a field designed for checking DNS records, where we discovered the existence of a user named Melina. Intrigued by this finding, we decided to conduct a brute force attack on Melina's account. To accomplish this, we utilized FoxyProxy with Burp Suite to intercept and analyze HTTP login traffic. With Burp Suite set up to try various common passwords as payload options for Melina's account, we launched the attack and eventually achieved success, discovering that the correct login credentials were "melina:melina". Upon logging in with these details, we discovered a page containing top secret legal data. After logging in as Melina and accessing the secret admin page, we noticed that the URL contained a session number. Although the page was locked and we couldn't see any sensitive material, we understood that we would need to obtain the correct session number to access this information. We followed a similar setup as our brute force login attack, using Burp Suite to try a variety of session IDs as payloads. After stating the attack, we discovered session 87 to be the one to reveal the data on the page.
Images	

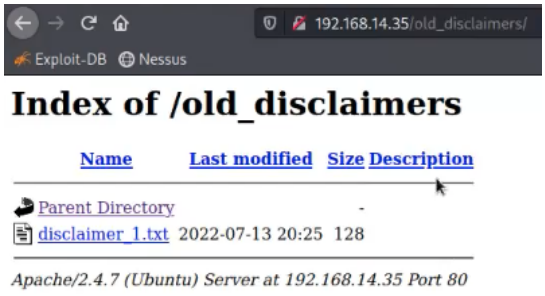


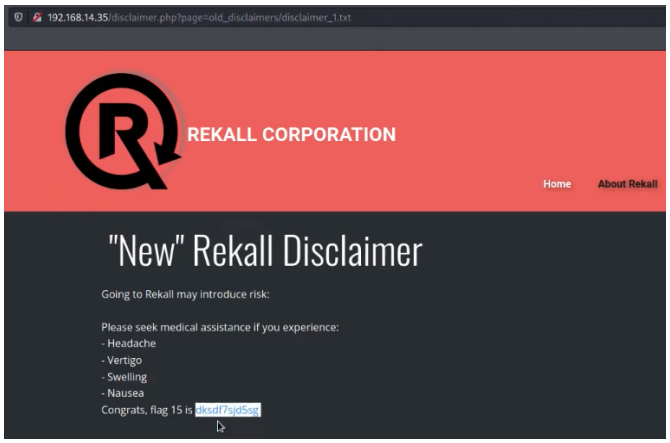




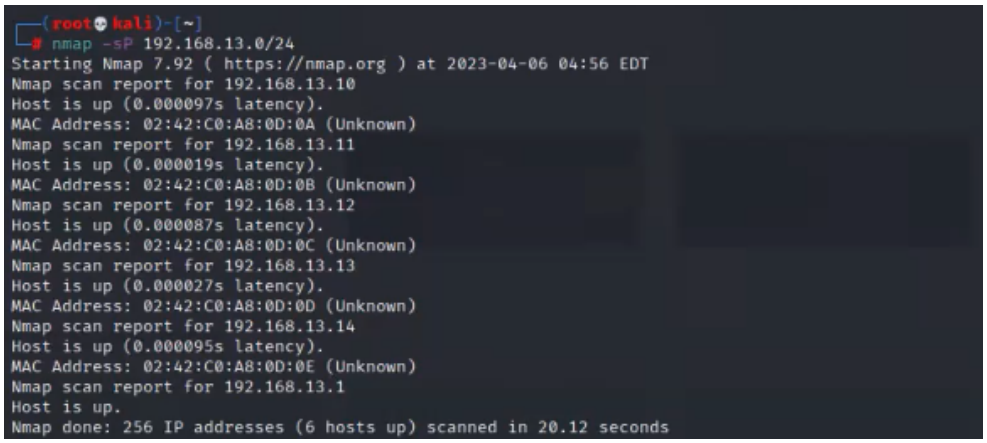
Affected Hosts	192.168.14.35
Remediation	1. Strong password policies: Implementing strong password policies such as mandatory password changes, password complexity requirements, and account lockout after multiple failed login attempts can prevent attackers from using brute force attacks to guess login credentials. 2. Captcha: Implementing captcha can prevent automated bots from attempting brute force attacks. 3. Rate limiting: Implementing rate limiting can prevent an attacker from trying too many login attempts within a short time period. 4. Session expiration: Implementing session expiration policies can ensure that sessions expire after a certain period of inactivity, reducing the likelihood of an attacker being able to use a stolen session ID. 5. Multifactor authentication: Implementing multifactor authentication can add an additional layer of security to prevent unauthorized access even if login credentials are compromised.

Vulnerability 9	Findings
Title	PHP Injection Vulnerability in Web Application
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**Medium**
Description	During an investigation, we discovered a page named souvenirs.php by accessing the robots.txt file. Upon navigating to this page, we observed that it was vulnerable to a PHP injection attack, as we were able to append a payload to the URL. We injected the payload ";system('cat /etc/passwd')" and were successful in executing the command, which allowed us to view sensitive data.
Images	
Affected Hosts	192.168.14.35
Remediation	1. Input validation: Implement input validation to ensure that user input is properly formatted and free from malicious code. 2. Escaping user input: Escape user input to prevent the server from interpreting it as code. 3. Disable dangerous functions: Disable dangerous PHP functions, such as "eval()", "system()", "exec()", and "passthru()". 4. Use web application firewalls: Implement web application firewalls to monitor and filter incoming traffic, blocking known malicious code.

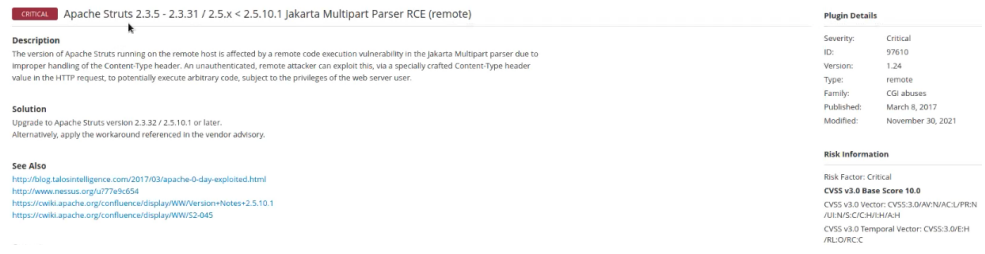
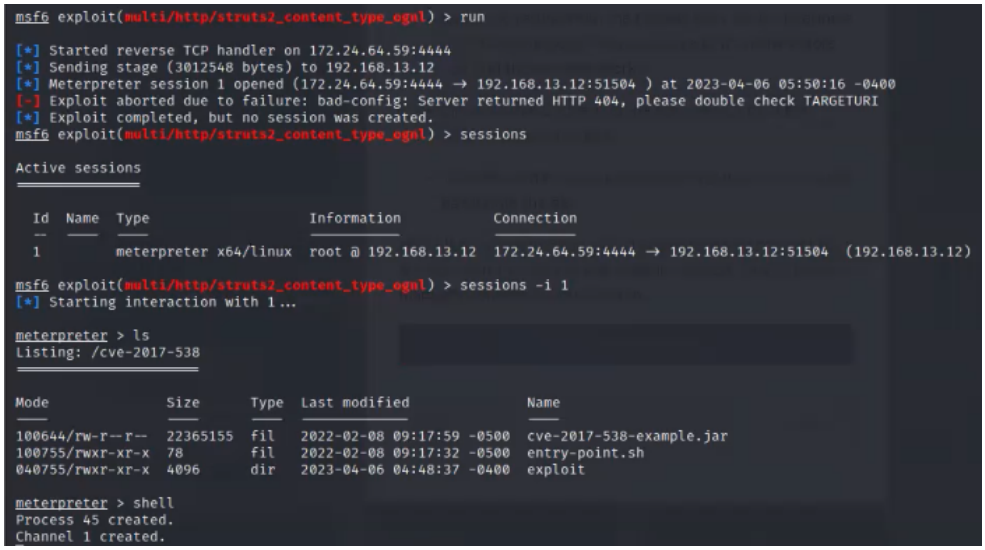
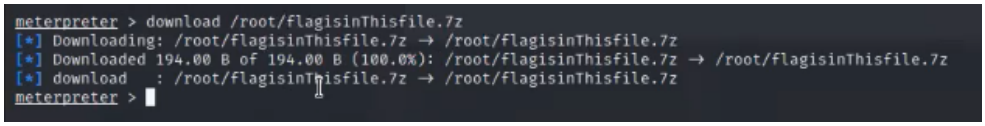
Vulnerability 10	Findings
Title	Directory Traversal Vulnerability in Web Application
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**High**
Description	During an investigation, we discovered a vulnerability in the web application that allowed us to perform a directory traversal attack. While conducting a command injection on the DNS checking field, we found an item named 'old_disclaimers' on the server. After navigating to this page, we identified a document named 'disclaimer_1.txt'. Using this information, we modified the URL on the disclaimer.php page to include '?page=old_disclaimers/disclaimer_1.txt', which enabled us to view the contents of the document and gain access to the sensitive information that we needed.
Images	

	
Affected Hosts	192.168.14.35
Remediation	1. Input validation: Implement input validation to ensure that user input is properly formatted and free from malicious code. 2. Whitelisting: Create a whitelist of allowed characters and only accept input that adheres to these specifications. 3. Limit file access: Use file permissions and access control mechanisms to limit access to files and directories to authorized users only. 4. Sanitize input: Sanitize user input by removing special characters, escape characters, and other potentially dangerous elements. 5. Disable directory listing: Disable directory listing on web servers to prevent attackers from gaining access to sensitive files and directories.

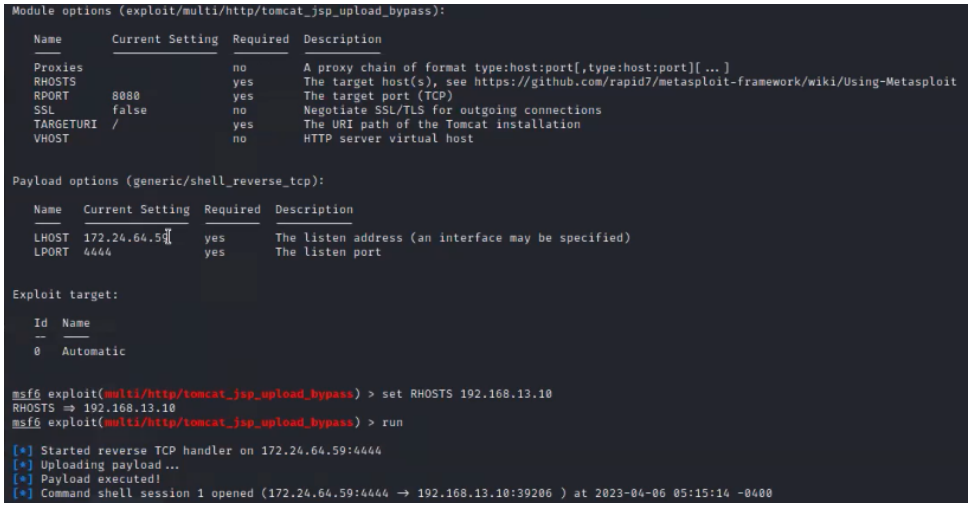
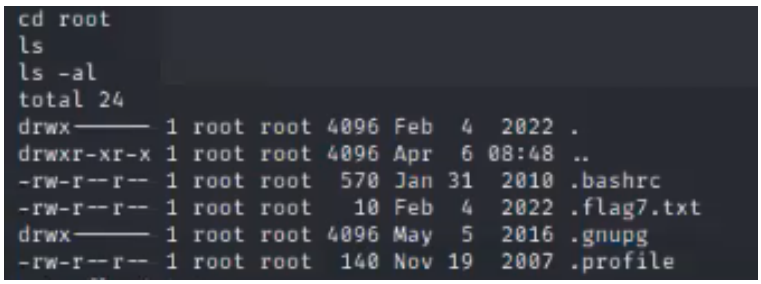
Vulnerability 11	Findings
Title	Network Scanning Reveals Exposed Hosts
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Low**
Description	We conducted an Nmap ping sweep scan on the network (192.168.13.0/24) to identify the number of online hosts, excluding the host from which the scan was initiated. The results showed that there were five online hosts, including 192.168.13.10, 192.168.13.11, 192.168.13.12, 192.168.13.13, and 192.168.13.14. This information provided us with a clearer understanding of the network topology, enabling us to perform further analysis and identify potential vulnerabilities.

Images	
Affected Hosts	192.168.13.10, 192.168.13.11, 192.168.13.12, 192.168.13.13, and 192.168.13.14
Remediation	1. Disable ICMP echo requests on hosts to prevent them from being detected during a ping scan. 2. Implement network segmentation and firewall rules to limit access to sensitive systems and resources and prevent unauthorized access. 3. Regularly scan the network for vulnerabilities and keep software up-to-date with the latest security patches and updates.

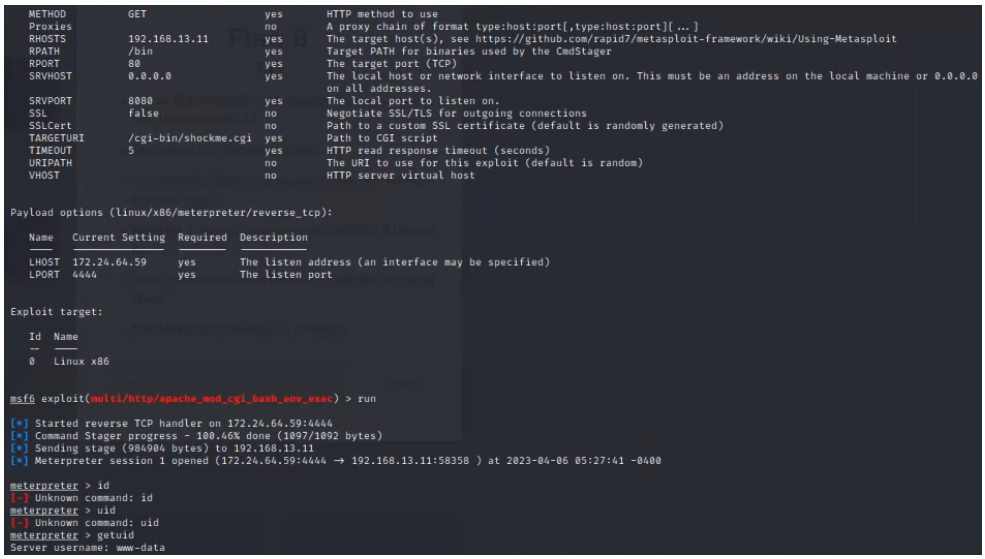
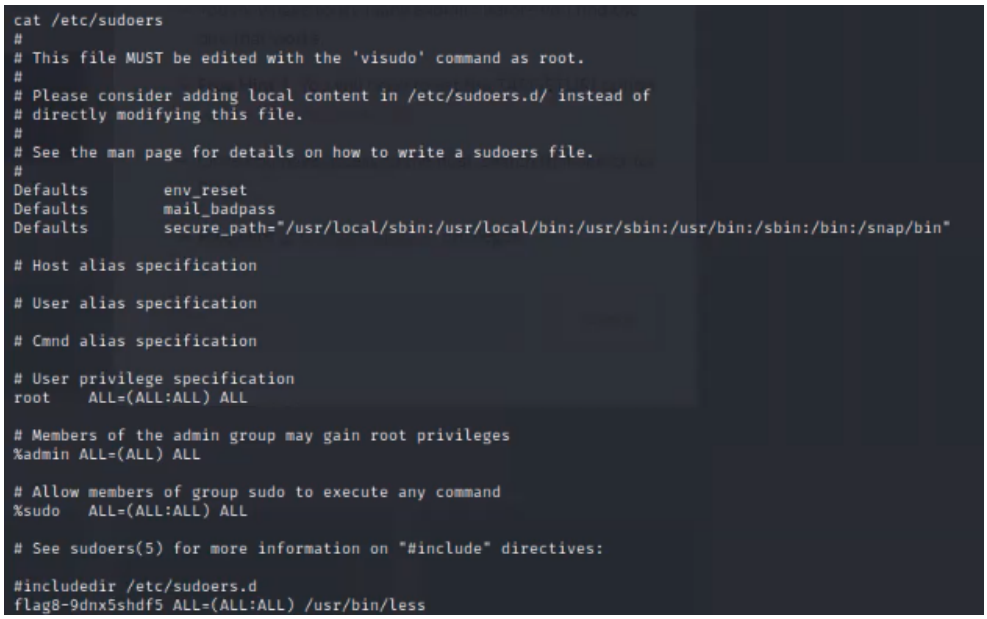
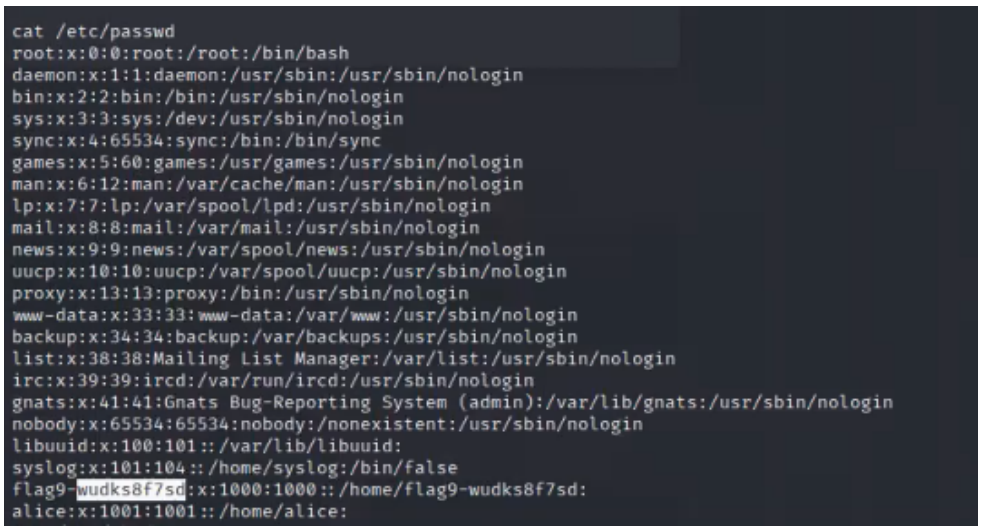
Vulnerability 12	Findings
Title	Remote Code Execution Vulnerability in Apache Struts Jakarta Multipart Parser
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	We conducted a basic network scan on the IP address 192.168.13.12 using Nessus. During the scan, we discovered a critical vulnerability related to the version of Apache Struts running on the host. Specifically, the host was found to be affected by a remote code execution vulnerability in the Jakarta Multipart parser. This vulnerability was caused by improper handling of the content-type header and was assigned the ID number 97610. With the knowledge that the host was vulnerable to this exploit, we utilized an

	apache struts Jakarta multipart parser OGNL injection module on metasploit. After successfully executing this module, we were able to gain access to the host and generate a shell. During our exploration of the host, we discovered a noteworthy 7z file located in the root folder. To investigate this file further, we downloaded it through our meterpreter session and were able to extract its contents on our local host. Overall, our successful exploitation of the vulnerability allowed us to uncover valuable information on the targeted host.
Images	   
Affected Hosts	192.168.13.12
Remediation	1. Upgrade to Apache Struts version 2.3.32 / 2.5.10.1 or later.

Vulnerability 13	Findings
Title	Tomcat RCE via JSP Upload Bypass vulnerability
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**High**
Description	During our aggressive nmap scan, we discovered that the IP address 192.168.13.10 had two ports open: port 8009, which was running Apache JServ, and port 8080, which was running Apache Tomcat/Coyote JSP engine 1.1. After identifying these services, we leveraged Metasploit to execute a Tomcat Remote Code Execution (RCE) via JSP Upload Bypass module. With this exploit, we were able to open a shell on the host, which gave us access to search the server for data, including the root folder. This successful exploitation provided us with valuable information about the compromised system.

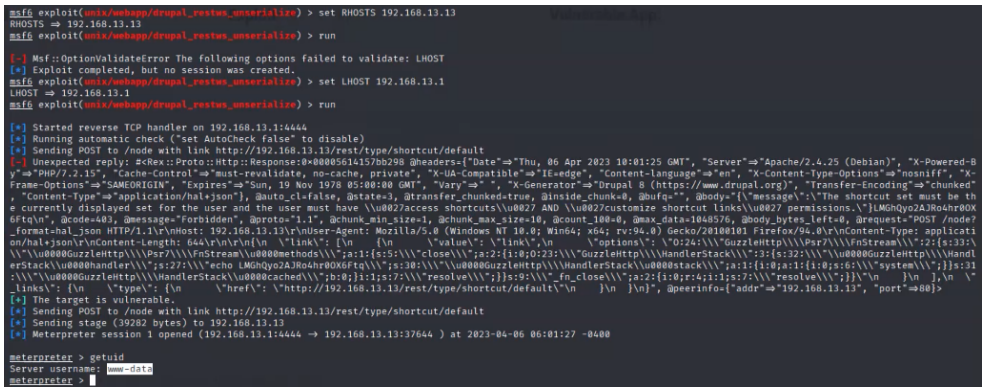
Images	 
Affected Hosts	192.168.13.10
Remediation	1. Apply the latest security patches and updates for Apache Tomcat, which may include patches for known vulnerabilities. 2. Disable unused services, ports, and protocols. In this case, disabling Apache Tomcat or configuring it securely can prevent this type of attack. 3. Use intrusion detection and prevention systems (IDPS) that can monitor network traffic for suspicious activity and block attacks in real-time.

Vulnerability 14	Findings
Title	Apache Bash Environment Variable Code Injection (Shellshock) Vulnerability
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**High**
Description	During our aggressive nmap scan, we discovered that the IP address 192.168.13.11 had port 80 open, running Apache HTTPS 2.4.7. We then leveraged Metasploit to execute an Apache mod_cgi Bash environment variable code injection (Shellshock) module. By exploiting this vulnerability, we were able to open a shell on the host, which provided us with access to the server's data, including the sudoers and passwd file. This successful exploitation allowed us to gain critical information about the compromised system.

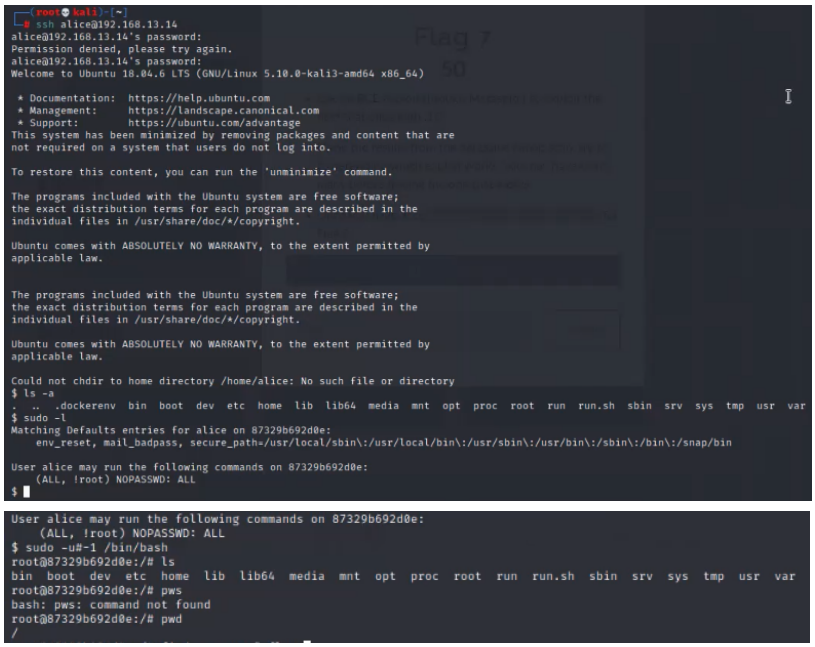
Images	  
Affected Hosts	192.168.13.11
Remediation	1. Update the Apache server to the latest version, which may include patches

	for known vulnerabilities. 2. Disable CGI modules and other non-essential features that are not required for the server to function properly. 3. Regularly monitor server logs for suspicious activity, including attempts to exploit the Shellshock vulnerability.

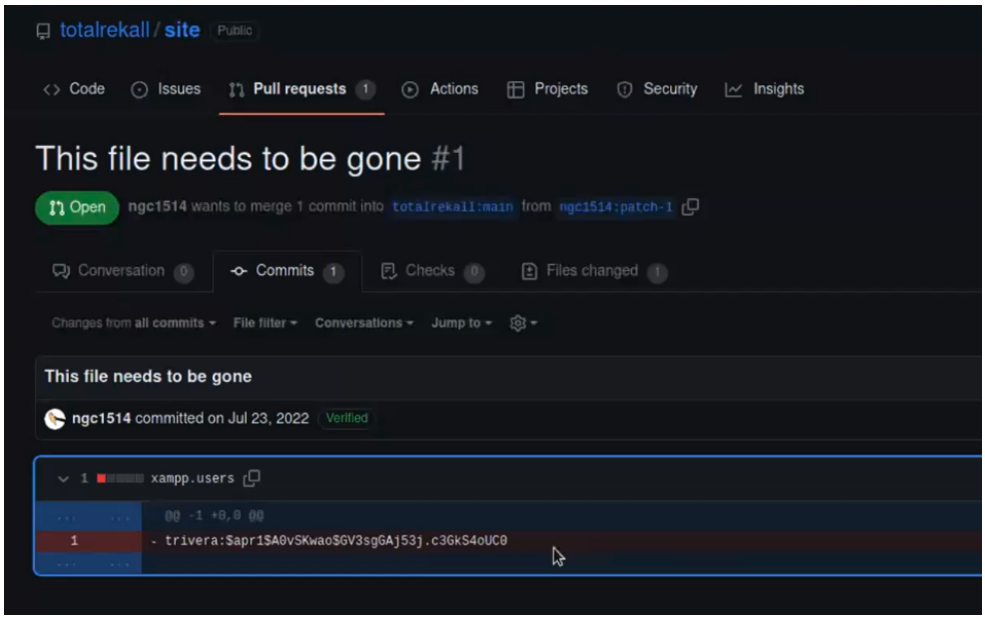
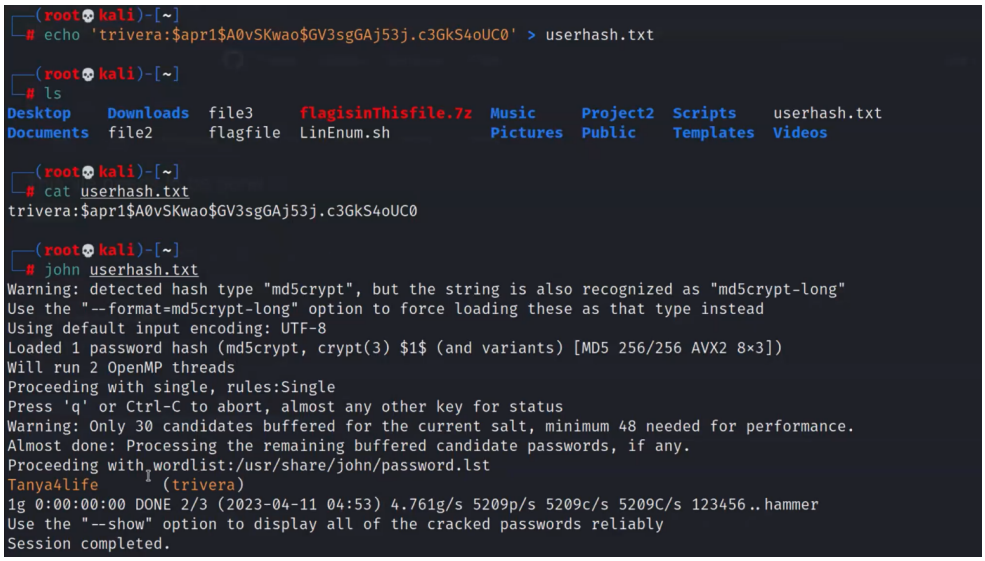
Vulnerability 15	Findings
Title	Remote Code Execution (RCE) via Drupal Web Services
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**High**
Description	Upon conducting an aggressive nmap scan, we observed that the IP address 192.168.13.13 had an open port 80, which was running Drupal 8. In order to further explore this vulnerability, we employed a Drupal Web Services RCE module on metasploit. This proved successful and allowed us to open a meterpreter session of the host. With our newly established access, we were able to investigate the targeted system and gather valuable insights.

Images	
Affected Hosts	192.168.13.13
Remediation	1. Patch Drupal: Install the latest security updates and patches provided by Drupal to address any known security vulnerabilities in the Web Services module. 2. Implement firewall rules: Configure the network firewall to block unauthorized traffic to ports and services that are not required for business operations. 3. Monitor network traffic: Use intrusion detection and prevention systems to monitor network traffic and identify potential security threats and attacks.

Vulnerability 16	Findings
Title	Privilege Escalation via Sudo Vulnerability (CVE-2019-14287)
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	During our aggressive nmap scan, we discovered that the IP address 192.168.13.14 had port 22 (ssh) open. Our earlier WHOIS domain query had provided us with information that 'Alice' was an sshUser on the system. Utilizing password guessing techniques, we were able to successfully access the host as Alice with the credentials 'alice:alice'. Upon gaining entry, we noted that Alice had the ability to run all commands except for those as root with no password

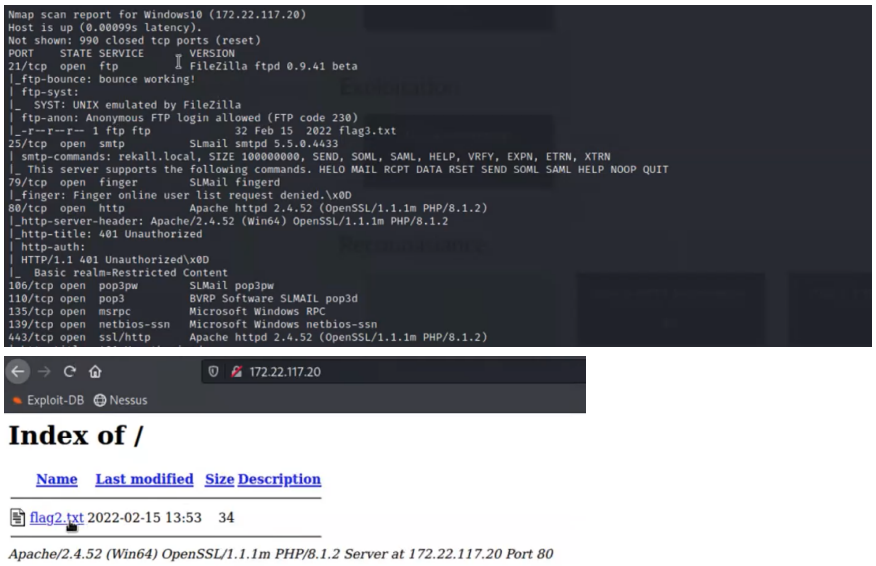
	requirement (ALL, !root) NOPASSWD: ALL. Leveraging the CVE-2019-14287 vulnerability, we were able to escalate privileges to root by executing the command "sudo -u#-1 /bin/bash". Our successful exploit allowed us to gain full access to the host and uncover valuable information.
Images	
Affected Hosts	192.168.13.14
Remediation	1. Update the system: Install all relevant patches and updates to ensure that the system is running the latest software and security fixes. 2. Review Sudoers file: Carefully review the Sudoers file to ensure that all users have the appropriate level of access and that the configuration does not contain any security vulnerabilities. 3. Implement strong password policies: Enforce strong password policies to prevent unauthorized access through password guessing or brute-force attacks. 4. Limit user privileges: Restrict user privileges to only those necessary for their role or job function to prevent unauthorized access and limit the impact of potential security breaches.

Vulnerability 17	Findings
Title	Potential Data Breach due to Sensitive Information on GitHub Repository
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	High
Description	As part of our investigation, we conducted a thorough search of the GitHub repositories belonging to totalrekall. During this process, we stumbled upon a file that immediately caught our attention as it appeared to contain sensitive information. Upon closer inspection, we found that the file contained a username,

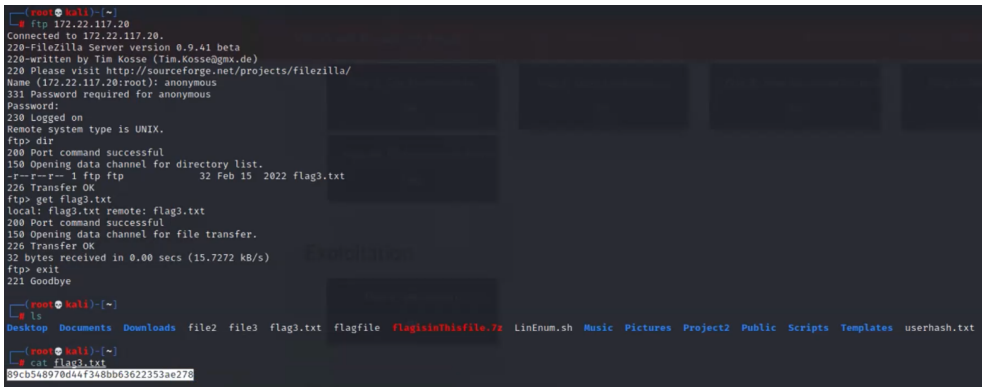
	'trivera,' along with their password hash. This discovery raised suspicions, leading us to believe that the file needed to be deleted to prevent any potential data breaches. With the aid of the John software, we were able to successfully decipher the password, revealing the login credentials for the 'trivera' account as 'Tanya4life.'
Images	 
Affected Hosts	172.22.117.20
Remediation	1. Remove the sensitive information: The first step is to remove the sensitive information from the file immediately. If the file is no longer required, it should be deleted. 2. Review the repository: Conduct a thorough review of the repository to ensure that there is no other sensitive information present that could lead to potential data breaches. 3. Change the password: Since the password for the 'trivera' account has been compromised, it is recommended that the account's password be changed immediately to prevent unauthorized access. 4. Educate users: Educate users on how to handle sensitive information to

	prevent similar incidents from occurring in the future.

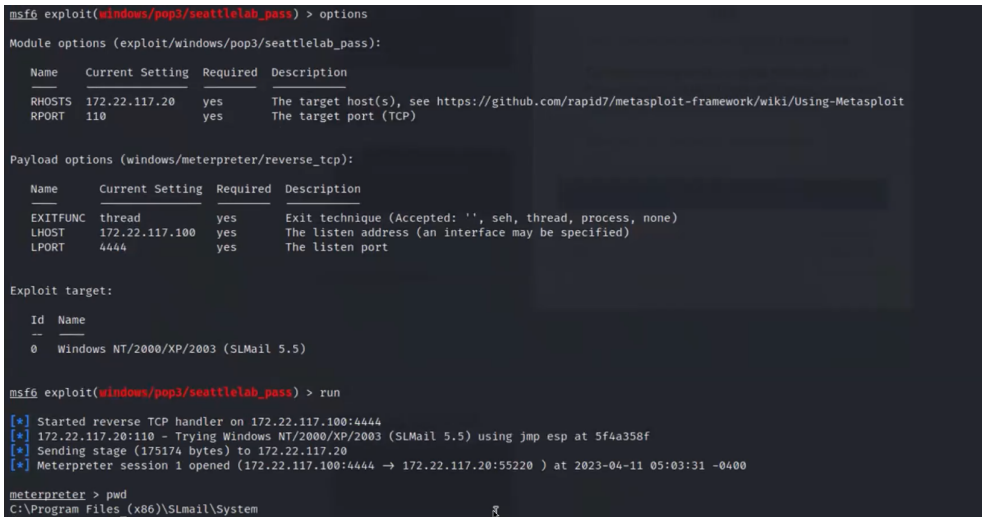
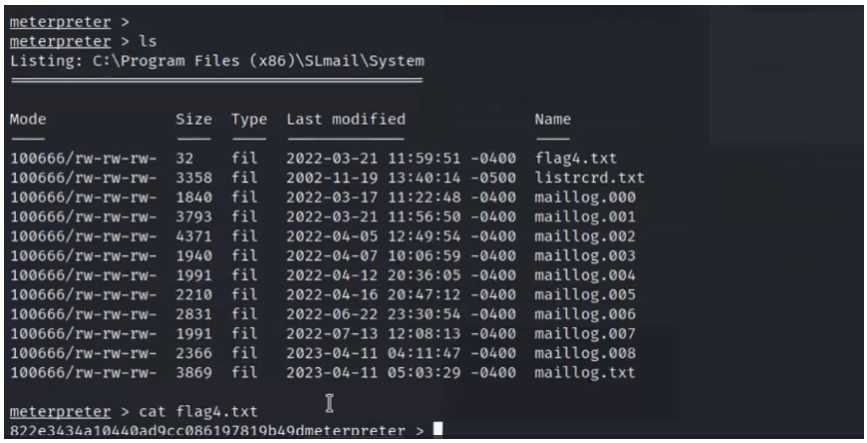
Vulnerability 18	Findings
Title	Unsecured Services and Weak Authentication Lead to Unauthorized Access
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**High**
Description	During our investigation, we employed a technique called http enumeration to gather information about the target network. We discovered that the Windows network had a subnet of 172.22.117.0/24 and proceeded to conduct an aggressive nmap scan. The scan revealed that IP address 172.22.117.20 had several ports open, including port 25 (smtp), port 21 (ftp), port 80 and port 443 (apache httpd 2.4.52). Using the login credentials obtained through the GitHub repository, we were able to successfully log into a website located on the internal network as 'trivera'. This allowed us to access information within a file on the website, providing us with further insights into the network.

Images	
Affected Hosts	172.22.117.20
Remediation	1. Secure network services: The open ports identified during the nmap scan need to be secured by implementing appropriate security measures. This could involve disabling unnecessary services, updating software and applying security patches, implementing firewalls, and monitoring network traffic. 2. Implement strong authentication: To prevent unauthorized access, strong authentication measures should be implemented, including the use of complex passwords, two-factor authentication, and limiting login attempts. 3. Conduct regular vulnerability scans: Conduct regular vulnerability scans and penetration tests to identify potential security weaknesses in the network and to address them promptly.

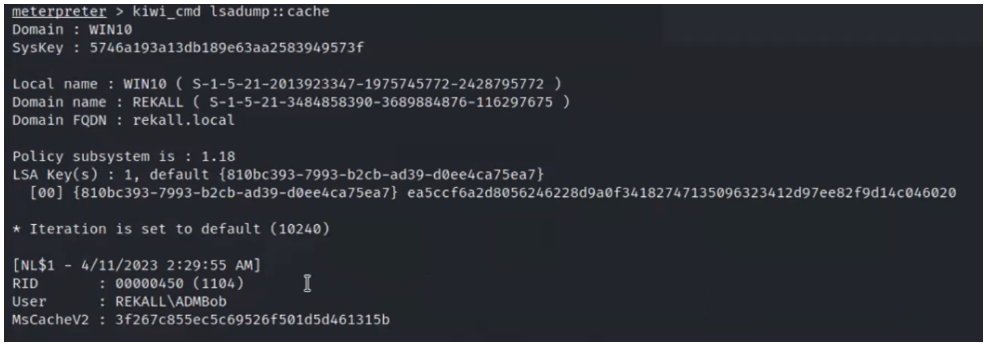
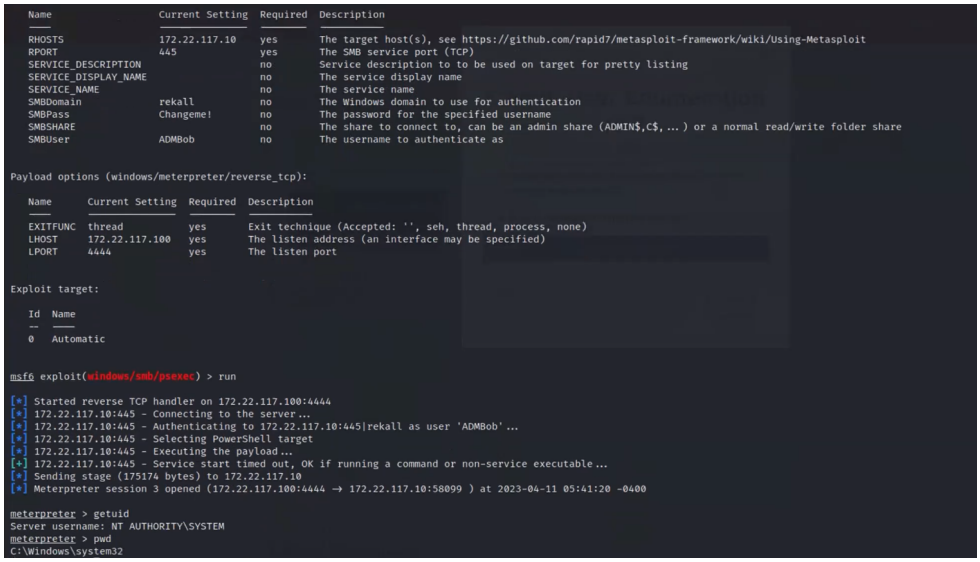
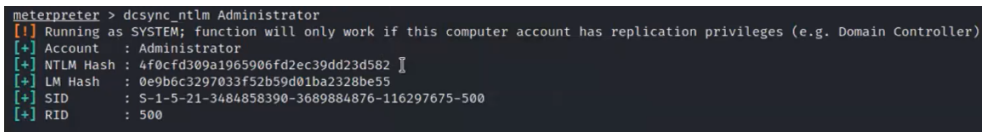
Vulnerability 19	Findings
Title	Unsecured FTP Service with Anonymous Login Allows Unauthorized Access
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**Medium**
Description	During our investigation, we performed an aggressive scan of the Windows network, which revealed that the IP address 172.22.117.20 had an open FTP port that allowed anonymous login. Leveraging this vulnerability, we logged in as anonymous using the 'ftp' command, without requiring a password. Once logged in, we used the 'get' command to extract a file from the server to our local system, which we were then able to read.

Images	
Affected Hosts	172.22.117.20
Remediation	1. Disable anonymous FTP login: Disable anonymous login for the FTP service to prevent unauthorized access. This can be done by configuring the FTP server to require valid user credentials before granting access. 2. Implement strong authentication: Implement strong authentication measures such as complex passwords, two-factor authentication, and limiting login attempts to prevent unauthorized access. 3. Monitor FTP activity: Monitor FTP activity closely to detect any unauthorized access or suspicious activity, and respond to it immediately. 4. Implement encryption: Implement encryption such as SSL/TLS to secure the FTP connection and protect data in transit.

Vulnerability 20	Findings
Title	Unauthorized Access to Sensitive Files through SLmail SMTPd Vulnerability
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	High
Description	During our investigation, we performed an aggressive scan of the Windows network and discovered that the IP address 172.22.117.20 had an open SMTP port with a vulnerable SLmail smtpd running. We then used a relevant SLmail module on Metasploit to exploit this vulnerability, which provided us with a meterpreter session for the host. This enabled us to view files containing sensitive data, thereby gaining unauthorized access to confidential information.

Images	 
Affected Hosts	172.22.117.20
Remediation	1. Ensure that SLmail smtpd is properly configured with appropriate security measures such as access controls and authentication mechanisms. 2. Disable unnecessary services or ports on the system to minimize the attack surface. 3. Regularly update the software to the latest version to ensure that any known vulnerabilities are patched.

Vulnerability 21	Findings
Title	Weak Passwords and Lateral Movement, allowing for DC compromise
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**Critical**
Description	During our exploit, we utilized the kiwi_cmd on the 172.22.117.20 meterpreter session to perform lsadump::cache. This operation revealed crucial information on a user named 'ADMBob', including their password mscash2 hash. We then utilized John and the applicable format to successfully crack the password, which turned out to be 'Changeme!'.

	Thanks to the nmap scan we conducted previously, we knew that the domain controller's IP address was 172.22.117.10. We then used a metasploit windows smb psexec module, armed with our newly found credentials, to move laterally into the domain controller. This granted us access to new and valuable data to explore further. In addition to this, we also leveraged kiwi in our meterpreter session on the domain controller with dcsync_ntlm. This allowed us to retrieve the NTLM hash of the Administrator, thereby officially compromising the admin.
Images	   
Affected Hosts	172.22.117.10, 172.22.117.20
Remediation	1. Implement strong password policies that require users to create complex passwords that are difficult to guess or crack.

	2. Consider using multi-factor authentication (MFA) to provide an additional layer of security to protect against stolen credentials. 3. Conduct regular vulnerability scans and penetration testing to identify and address weaknesses in the network and ensure that systems are up-to-date with the latest security patches. 4. Review and enforce access controls to limit lateral movement within the network. 5. Use network segmentation to isolate critical assets and implement firewalls to restrict access to vulnerable systems.