Cybersecurity

Penetration Test Report

Rekall Corporation

Penetration Test Report

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

Contact Information

Company Name	Zero Day Cyber Assurance, LLC
Contact Name	Paul Lindsay
Contact Title	Penetration Tester

Document History

Version	Date	Author(s)	Comments
001	2024-03-03	Paul Lindsay	PenTest

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:

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.

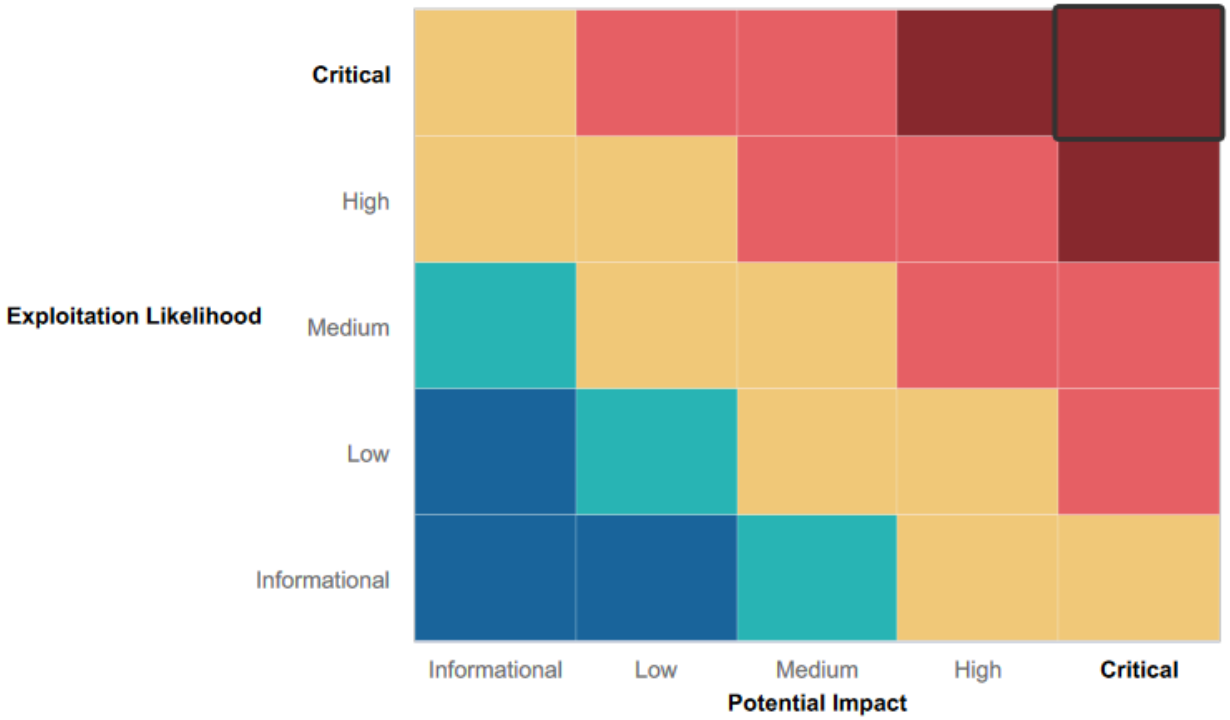
Executive Summary of Findings

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.

As we are always so focused on the vulnerabilities in these reports it is important to notice the good side of things, although our team was able to get around certain situations your security already had some input validation methods in place to protect against some of the attacks that were conducted. Although you see everything that was successful as that is what the report is meant to do behind the scenes there are multiple exploits that were tried but came unsuccessful indicating up that Total Rekall has implemented effective security configurations and patches for known vulnerabilities in certain aspects of there network and infrastructure.

As mentioned above, given the complexity needed to bypass the input validation in certain exploits, it's clear that Total Rekall has employed input validation measures for their web applications. The varieties of the systems and application under attack implies that Total Rekall have a layered security approach, we learn this by having to specifically tailor exploits to bypass security controls indicating the presence of multiple defensive layers.Some data exposure attempts were unsuccessful which would indicate to us that the use of encryption and secure data transmission practices protecting sensitive information are taking place during storage and in transit.

The strengths of this network and web application is displayed to us in the form of exploits not working in certain areas likely due to the commitment to regular patching in some services, effective incident detection and response, robust input validation and encryption practices, these strengths suggest that Total Rekall is aware of cybersecurity landscape and has taken steps in the right direction to take action and reduce the attack surface for any malicious actors.

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.

Exploit:XSS Cross Side Scripting
Host IP address:192.168.14.35
Port:WEBBASED
Service name:N/A
Service version: N/A
Input: <script>alert('XSS test alert!')</script>
Exploit outcome: Alert pops up on screen indicating that input was received as html

---

Exploit: Advanced XSS scripting avoiding input validation
Host IP address: 192.168.14.35/Memory-Planner.php
Port: N/A
Service name: N/A
Service version: N/A
Input: <SCRIPscriptT>alert("XSS test alert!")</SCRIPscripTt>
Exploit outcome: Alert created meaning input validation can be worked around in this instance.

Exploit: Stored XSS

Host IP address: 192.168.14.35/comments.php
Port: N/A
Service name: N/A
Service version: N/A
Input : <script>alert('XSS stored test')
Exploit outcome: executed a simpt XSS Stored in the comments section which returned an alert comments being stored are being read as raw html

---

Exploit: Sensitive Data Exposure (HTTP HEADERS via Curl / BURP)
Host IP address: 192.168.14.35/About-Rekall.php
Port: N/A
Service name: N/A
Service version: N/A
Exploit outcome: When using curl command against host it shows sensitive information.

Exploit: Local File Inclusion
Host IP address: 192.168.14.35/Memory-Planner.php
Input : script.php (upload)
Exploit outcome: Uploaded "script.php" to the section that is meant for adventure images

---

Exploit: Local File Inclusion (Input Validation)
Host IP address: 192.168.14.35/Memory-Planner.php
Input : script.jpg.php (upload)
Exploit outcome: Uploaded avoiding the input validation looking for ".jpg" still managing to get our file uploaded successfully

---

Exploit: Sensitive Data Exposure
Host IP address: 192.168.4.35/robots.txt
Input: 192.168.4.35/robots.txt
Exploit outcome: when inputting this into the address field we were able to view contents of robots.txt

---

Exploit: Command Injection (advanced)
Host IP address: 192.168.14.35/networking.php
Input: www.example.com | cat vendors.txt
Exploit outcome: When passing in a command in the MX Record Checker it returns us the output of the linux command.

Exploit: Brute Force Attack
Host IP address: 192.168.14.35/Login.php
Input: melina
Exploit outcome: using the above exploit we were able to view the /etc/passwd file showing a user "melina" this user has a weak password which we cracked.

Exploit: PHP Injection
Host IP address: 192.168.14.35/souvenirs.php
Input: http://192.168.13.35/souvenirs.php?message=""; system('cat /etc/passwd')
Exploit outcome: We recovered this hidden page from the earlier discovery of robots.txt When deploying PHP injection into the web browser we get access to view the /etc/passwd contents.

Exploit: Open Source Exposed Data
Host IP address: totalrekall.xyz
Input: totalrekall ====> https://centralops.net/co/DomainDossier.aspx
Exploit outcome: When running a whois on the domain totalrekall.xyz we are provided lots of exposed data.

Recon Method : nslookup scan
Host IP address: totalrekall.xyz
Input: nslookup -type=txt totalrecall.xyz
Exploit outcome:

Exploit: Open Source exposed data
Host IP address: totalrekall.xyz
Input: totalrekall.xyz ----> crt.sh
Exploit outcome: Data exposed using open source tool by simply typing in the domain.

Location: Scan results
Host IP address: 192.168.13.0/24
Input: nmap -sV 192.168.13.0/24
Outcome: After reviewing the nmap scan we can see clearly there are 6 hosts on the network.

Location: Scan Results
Host IP address: 192.168.13.0/24
Input: nmap -A 192.168.13.0/24

Outcome: Analyzing the aggressive nmap scan we can see that the hose running Drupal is 192.168.13.13

Location: Nessus Scan results
Host IP address: 192.168.13.12
Input:
Outcome: After reviewing the information we got from the Nessus scan we were able to see a critical vulnerability for Apache Struts //ID: 97610//

Exploit: Apache Tomcat Remote Code Execution
Host IP address: 192.168.13.10
Input: tomcat_jsp_upload_bypass
Exploit outcome: executed the payload via metasploit to gain a successful root shell as you can see below

---

Exploit: Shellshock
Host IP address: 192.168.13.11
Input: apache_mod_cgi_bash_env_exec
Exploit outcome: was able to establish a meterpreter session which I was then able to gain a shell to display the sudoers file and the passwd file from the /etc/ directory

---

Exploit:
Vulnerability: Struts - CVE-2017-5638
Host IP address: 192.168.13.12
Input:
Exploit outcome: was able to establish a meterpreter session then able to extract sensitive files back to our kali machine for analysis.

Vulnerability: Drupal - CVE-2019-6340
Host IP address: 192.168.13.13
Exploit: drupal_restws_unserialize
Exploit outcome: Was able to gain a meterpreter session on the 192.168.13.13 machine

---

Vulnerability: CVE-2019-14287
Exploit: BruteForce
Host IP address: 192.168.13.14
Input: password cracking methods
Exploit outcome: Cracked password resulting in unauthorized sign in to the user "Alice"

Vulnerability: Open source exposed data
Host IP address: Total Rekall Github repository directory
https://github.com/totalrekall/site/blob/main/xampp.users
Input: Total Rekall search ==> Public repository ===> xampp.users
Outcome: Sensitive information recovered. Then password hash cracked giving us valid credentials.

Input: nmap scan on host IP
Vulnerability: http Apache httpd 2.4.52 (OpenSSL/1.1.1m PHP/8.1.2)
Host IP address: 172.22.117.20
Outcome: Using this vulnerability we input 172.22.117.20 into the web browser giving us an Index page containing sensitive information.

Vulnerability: open ftp FileZilla ftpd 0.9.41 beta
Host IP address: 172.22.117.20
Input: ftp 172.22.117.20 ==> name=anonymous
Exploit outcome: Using nmap we discover 21/tcp port open on 172.22.117.20 machine, using this we are able to extract flag3.txt and then view on our kali machine.

Vulnerabitility: SLMail SMTP 25
Host IP address: 172.22.117.20
Exploit: windows/pop3/seattlelab_pass
Exploit outcome: Exploit completed, meterpreter session established, able to see and access sensitive information

---

Exploit: windows/pop3/seattlelab_pass
Host IP address: 172.22.117.20
Input: schtasks /query /TN flag5 /F0 list /v
Exploit outcome: After establishing a meterpreter session we were able to view all the scheduled tasks in the WIN10 machine

---

Exploit: windows/pop3/seattlelab_pass
Host IP address: 172.22.117.20
Exploit outcome: was able to gain a meterpreter session, using this was able to obtain user information in the form of a password hash, using john we were able to crack this hash giving us valid credentials to the system.

Exploit: windows/smb/psexec
Host IP address: 172.22.117.10
Exploit outcome: Using the credentials we obtained by cracking the hash we were now able to establish connection into the 2nd machine on the network.

Executive Summary

This penetration test was conducted to evaluate the security posture of Total Rekall's network and web application. ZDCA was able to identify a variety of vulnerabilities, these findings indicate there are multiple security risks that could potentially be exploited by attackers to gain unauthorized access to systems in the network and possibly extract sensitive information and compromise the integrity and availability of Total Rekall's systems and data.

Several instances of both reflected and stored XSS were identified across different web applications, enabling execution of javascript, this vulnerability was present in various forms, demonstrating the applications input validation and output encoding mechanisms were not working as intended. Sensitive data exposure was found to leak sensitive information due to improper configuration and security practices, this exposure includes HTTP headers and sensitive files accessible via simple HTTP requests.

The application endpoint 'Memory-Planner.php' was vulnerable to Local File Inclusion attacks allowing attackers to upload and execute system commands, providing attackers with unauthorized access to the system. Remote Code Execution (RCE) was also a point of weakness when carrying out our testing, exploits like Apache Tomcat Remote Code Execution and vulnerabilities in Apache Struts and Drupal were successfully exploited, granting control over the affected systems. These vulnerabilities were all exploited using known techniques which helps us highlight the need for regular patch management and security assessments.

Weak passwords were identified and cracked to help gain unauthorized access to user accounts and sensitive areas of the application. Through various scanning techniques we were able to identify misconfigured services and devices exposing unnecessary services to the network which could be leveraged for further attacks.

ZDCA strongly recommends that you regularly update all software, frameworks, and plugins to their latest versions to mitigate known vulnerabilities, ensure proper configuration of web servers, applications, and network devices to minimize the attack surface. Adopt comprehensive input validation and output encoding to prevent XSS and other injection attacks, secure file upload features by implementing strict validation on file types, sizes, names, and by segregating uploaded files from executable server directories.

Using strong password policies while implementing multi-factor authentication is a great and cost effective way to prevent unauthorized access. Regularly review and limit user permissions according to the principle of least privilege, this will make sure that any user does not have the ability to do anything or access anything that is not intended for that specific user. Conducting periodic security assessments and vulnerability scans is a safe way to keep up to date and secure in the ever evolving world of cyber secuity. Lastly, providing ongoing security training to developers and administrators will help foster a culture of security awareness, the vulnerabilities identified during this penetration test pose significant security risks. Immediate action is recommenced to address these issues to help protect Total Rekall's clients assets and maintain trust of users. Implementing the recommendations above will significantly enhance the security posture of Total Rekall's digital environment.

Summary Vulnerability Overview

Vulnerability	Severity
Stored XSS	Critical
Command Injection (advanced)	Critical
Brute Force Attack	Critical
PHP Injection	Critical
Open Source Exposed Data	Critical
Apache Tomcat Remote Code Execution	Critical
Shellshock	Critical
Struts - CVE-2017-5638	Critical
Drupal - CVE-2019-6340	Critical
CVE-2019-14287	Critical
Apache httpd 2.4.52	Critical
Windows/pop3/seattlelab_pass (Multiple occurrences)	Critical
Windows/smb/psexec	Critical
Local File Inclusion	High
Sensitive Data Exposure	High
Sensitive Data Exposure (HTTP HEADERS via Curl / BURP)	High
Windows/pop3/seattlelab_pass (Multiple occurrences)	High
Advanced XSS scripting avoiding input validation	Medium
Local File Inclusion (Input Validation)	Medium
Windows/pop3/seattlelab_pass	Medium
XSS Cross-Site Scripting (XSS)	Low
Open source exposed data (Multiple occurrences)	Low

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

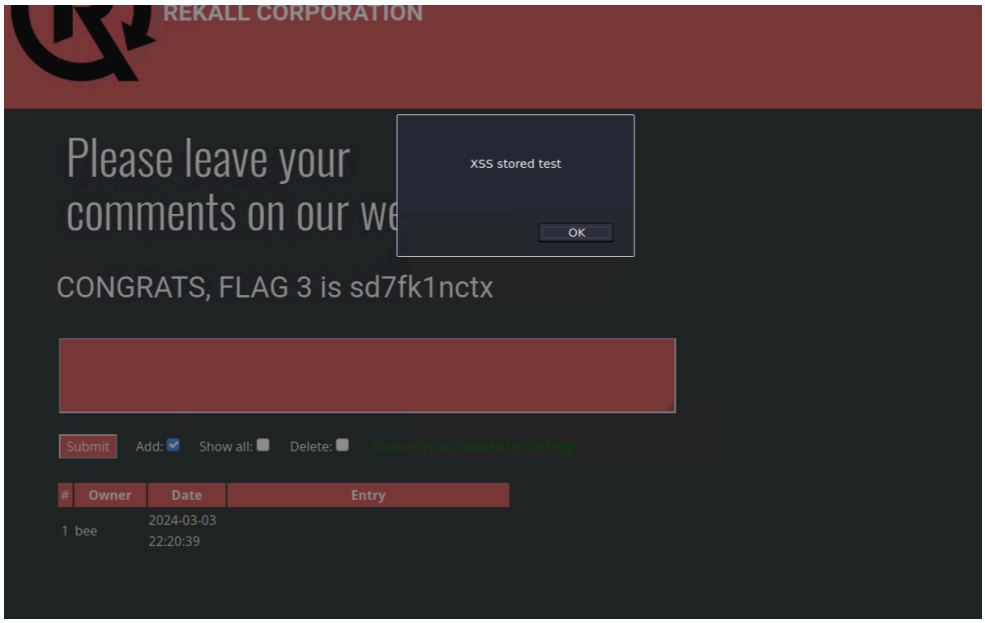
Scan Type	Total
nmap	7
Nessus	1
nslookup	1
Hosts	9
Ports	• 80/443 (HTTP/HTTPS) - Implied by web-based exploits and services.

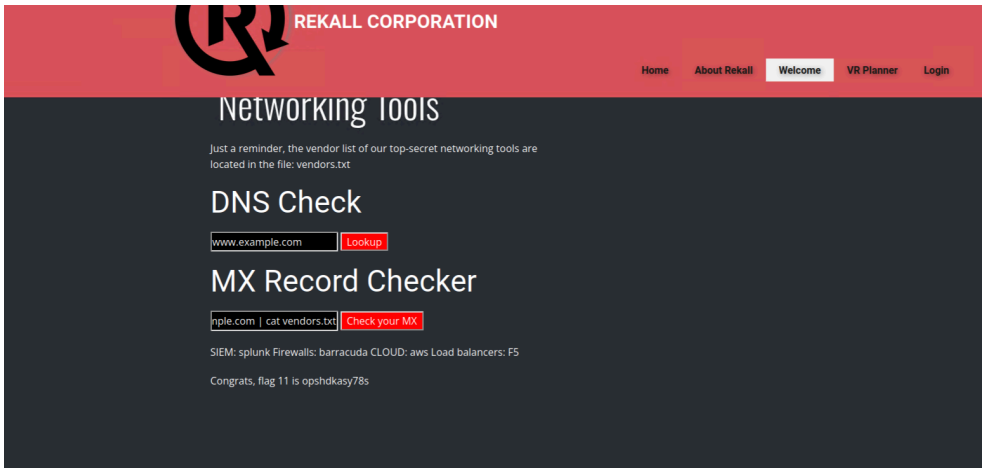
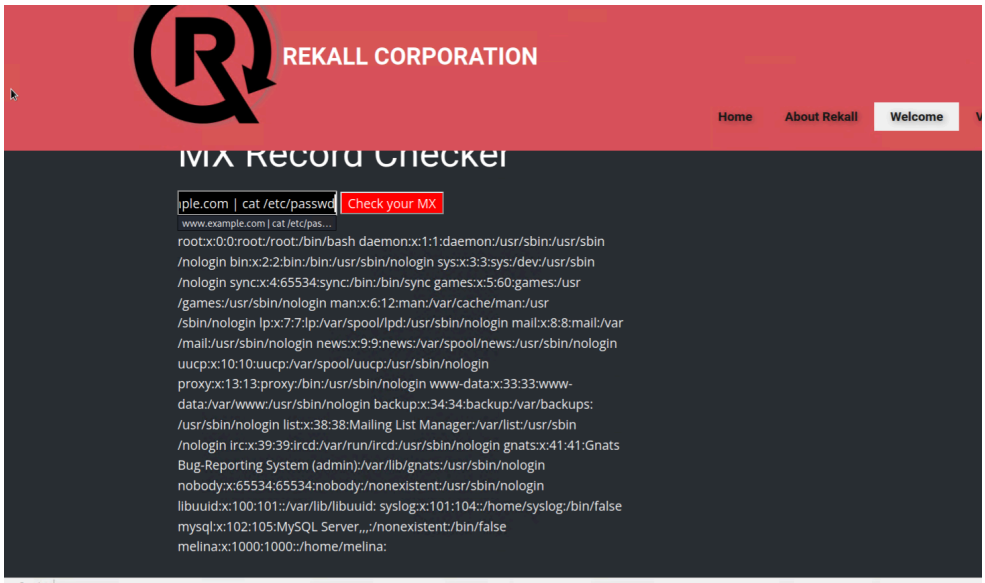
	• 21 (FTP) - Explicitly mentioned. • 25 (SMTP) - Inferred from the SLMail exploit. • 445 (SMB) - Implied by the use of a Windows SMB exploit. • 139 (SMB, if applicable) - Potentially involved in SMB communication.

Exploitation Risk	Total
Critical	13
High	4
Medium	3
Low	2

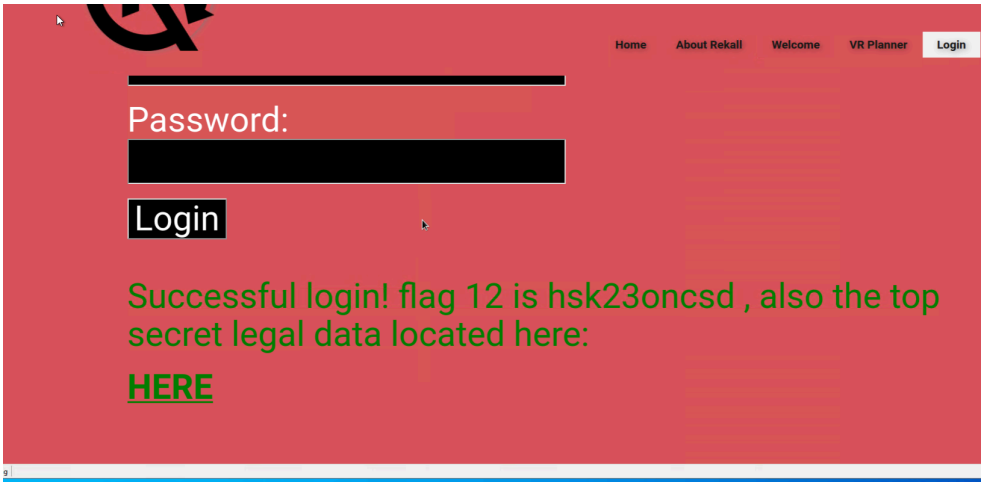
Vulnerability Findings

Vulnerability 1	Findings
Title	**Stored XSS**
Type (Web app / Linux OS / WIndows OS)	Web App
Risk Rating	**Critical**
Description	Executed a simple XSS Stored in the comments section which returned an alert comments being stored are being read as raw html

Images	
Affected Hosts	192.168.14.35/commnts.php
Remediation	Input Sanitization and Validation: Implement robust input sanitization and validation mechanisms on the server-side to filter and validate user inputs, especially in comment fields. This includes filtering out or escaping special characters such as <, >, &, and quotes (' and ") to prevent injection attacks. Content Security Policy (CSP): Implement a Content Security Policy (CSP) on the web server to mitigate the risk of XSS attacks. CSP allows web developers to declare which sources of content are considered legitimate for a web page, thereby reducing the risk of executing malicious scripts. Encoding Output: Encode user-generated content before rendering it in the browser. Use HTML entity encoding or other encoding mechanisms to ensure that user-supplied data is treated as plain text and not as executable script code. Regular Security Audits: Conduct regular security audits and code reviews of the application to identify and address potential vulnerabilities, including XSS vulnerabilities. Automated security scanning tools can be used to assist in identifying vulnerabilities. Educate Users: Educate users about the risks of XSS attacks and encourage them to use strong passwords and to avoid clicking on suspicious links or executing untrusted scripts. Security Headers: Implement security headers such as X-XSS-Protection, X-Content-Type-Options, and X-Frame-Options to provide an additional layer of defense against XSS attacks. Patch and Update: Ensure that all software components, including web servers, frameworks, and libraries, are kept up to date with the latest security patches and updates to address known vulnerabilities.

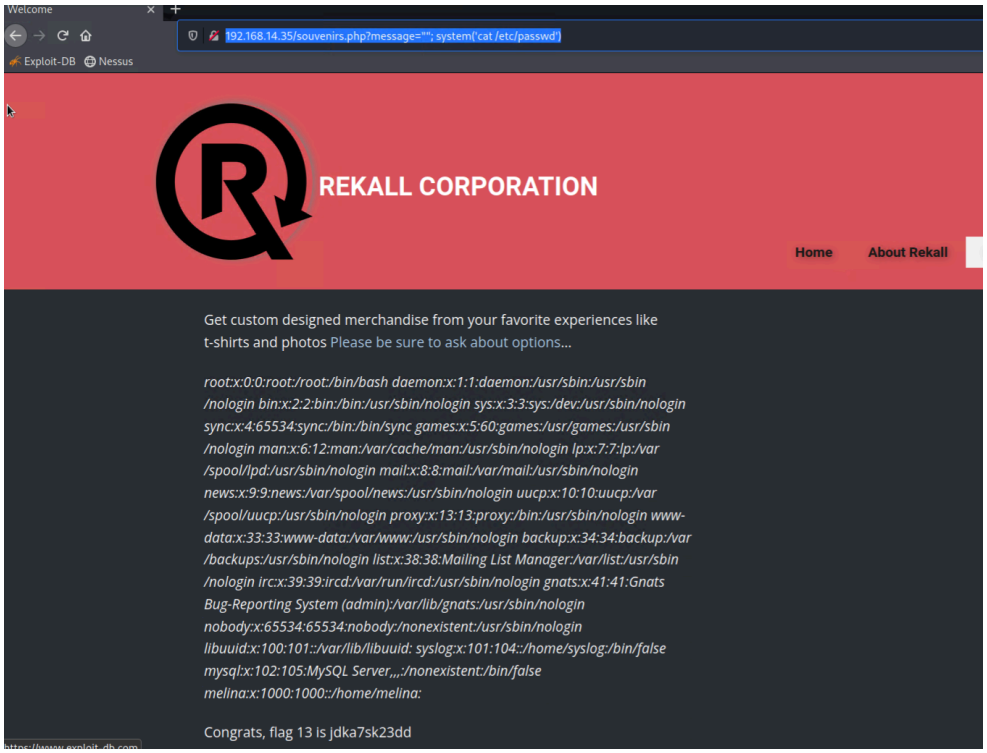
Vulnerability 2	Findings
Title	**Command Injection**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Critical**
Description	This exploit takes advantage of the vulnerability in the MS Record Checker, a feature that likely interacts with the system's command-line interface.
Images	 
Affected Hosts	192.168.14.35/networking.php
Remediation	Input Validation and Sanitization: Ensure that all user inputs are properly validated and sanitized before being processed by the server. Input validation involves checking user input to ensure it conforms to expected formats, while sanitization involves removing or escaping any potentially dangerous characters or sequences. Implement strict input validation and sanitize inputs to prevent

	attackers from injecting malicious commands. Use Safe APIs: Whenever possible, avoid using system calls or shell commands to process user input. Instead, use safe APIs provided by your programming language or framework to achieve the desired functionality. Safe APIs are designed to prevent command injection vulnerabilities. Parameterized Queries: If interacting with a database, use parameterized queries or prepared statements to handle user input safely. Parameterized queries separate SQL code from user input, preventing SQL injection vulnerabilities which can sometimes be related to command injection. Least Privilege Principle: Ensure that the web server and any associated processes run with minimal privileges necessary to perform their tasks. This limits the potential damage that can be caused if a vulnerability is exploited. Security Headers and Content Security Policy (CSP): Implement security headers like Content Security Policy (CSP) to restrict the sources from which resources (e.g., scripts, stylesheets, images) can be loaded. This can help prevent execution of malicious scripts injected through command injection vulnerabilities. Regular Security Audits and Penetration Testing: Conduct regular security audits and penetration testing to identify and address any vulnerabilities, including command injection vulnerabilities. This helps ensure that your system remains secure against evolving threats. Update and Patch Systems: Keep all software, including web servers, frameworks, and libraries, up to date with the latest security patches and updates. Vulnerabilities are often discovered and patched by software vendors, so staying current with updates is crucial for maintaining security. Security Awareness Training: Educate developers and system administrators about secure coding practices and common security vulnerabilities, including command injection. Security awareness training helps personnel recognize and mitigate security risks effectively. By implementing these remediation steps, you can significantly reduce the risk of Command Injection vulnerabilities and enhance the overall security posture of your web application or system.

Vulnerability 3	Findings
Title	**Brute Force Attack**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Critical**
Description	Using information from command injection exploit we were able to view the user melina, applying brute force techniques we were able to crack the password and gain access to melinas administrator accout.
Images	
Affected Hosts	192.168.14.35/Login.php
Remediation	Strong Password Policy: Enforce a strong password policy that requires users to create complex passwords containing a mix of uppercase and lowercase letters, numbers, and special characters. Additionally, encourage users to regularly update their passwords and avoid using easily guessable information. Account Lockout Mechanism: Implement an account lockout mechanism that temporarily locks user accounts after a certain number of failed login attempts. This prevents attackers from repeatedly attempting to guess passwords for a specific user account. Once locked, the account should only be unlocked by an administrator or after a specified period of time. CAPTCHA or Human Verification: Integrate CAPTCHA or other human verification mechanisms into the login process to differentiate between legitimate users and automated bots. This can help prevent automated brute force attacks by requiring users to solve challenges that are difficult for bots to pass. Rate Limiting:

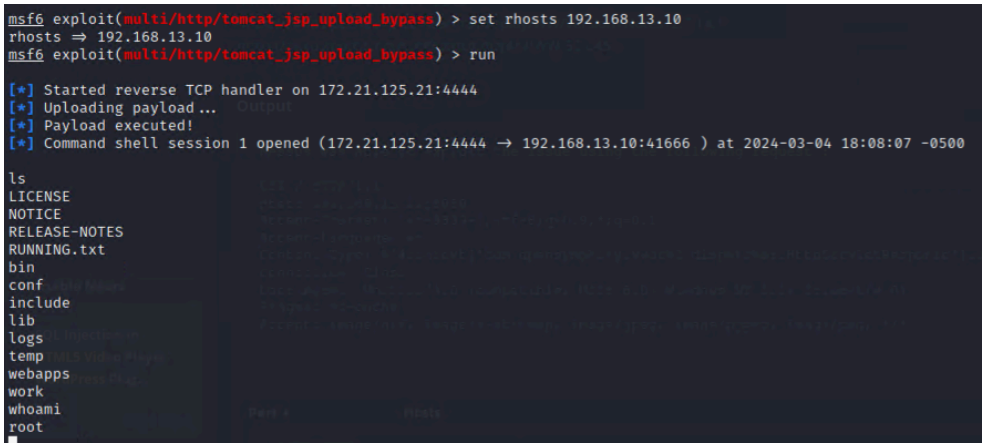
	Implement rate limiting on login attempts to restrict the number of login requests that can be made within a specific time frame from a single IP address or user account. This helps prevent brute force attacks by slowing down the rate at which login attempts can be made. Monitoring and Logging: Implement comprehensive logging and monitoring of login attempts, including successful and failed login events. Regularly review and analyze these logs to detect and respond to suspicious login patterns, such as multiple failed login attempts from the same IP address. Multi-factor Authentication (MFA): Implement multi-factor authentication (MFA) to add an additional layer of security to the login process. Require users to authenticate using multiple factors, such as a password combined with a one-time code sent to their mobile device or generated by an authenticator app. Security Headers and HTTPS: Use security headers like HTTP Strict Transport Security (HSTS) and implement HTTPS to encrypt communication between the client and server. This helps protect against man-in-the-middle attacks and ensures the confidentiality and integrity of login credentials. Regular Security Audits and Updates: Conduct regular security audits to identify and address vulnerabilities in the login functionality. Keep the server, web application, and any associated software up to date with the latest security patches and updates to mitigate known vulnerabilities.

Vulnerability 4	Findings
Title	**PHP Injection**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Critical**
Description	We recovered this hidden page from the discovery of robots.txt, when deploying PHP Injection into the web browser we get access to view the /etc/passwd contents

Images	
Affected Hosts	192.168.14.35/souvenirs.php
Remediation	Input Validation and Sanitization: Implement strict input validation and sanitization for all user-supplied data, including URL parameters, form inputs, and cookies. Validate input data against expected formats and sanitize it to remove or encode any potentially dangerous characters or sequences. Use whitelisting approaches whenever possible to only allow expected input patterns. Secure Coding Practices: Follow secure coding practices when developing PHP applications. Avoid using functions like `eval()` or `system()` that can execute arbitrary code provided by users. Instead, use safer alternatives or libraries that provide built-in protection against code injection vulnerabilities. Parameterized Queries: If your application interacts with a database, use parameterized queries or prepared statements to securely handle user input in SQL queries. Parameterized queries separate SQL code from user input, preventing SQL injection vulnerabilities which can sometimes be related to PHP injection. Disable Dangerous PHP Functions: Disable or restrict access to dangerous PHP functions that can be abused for code execution, such as `system()`, `exec()`, `shell_exec()`, `passthru()`, `popen()`, and `proc_open()`. This can be done by configuring PHP settings in the php.ini file or using functions like `disable_functions()` in your PHP code.

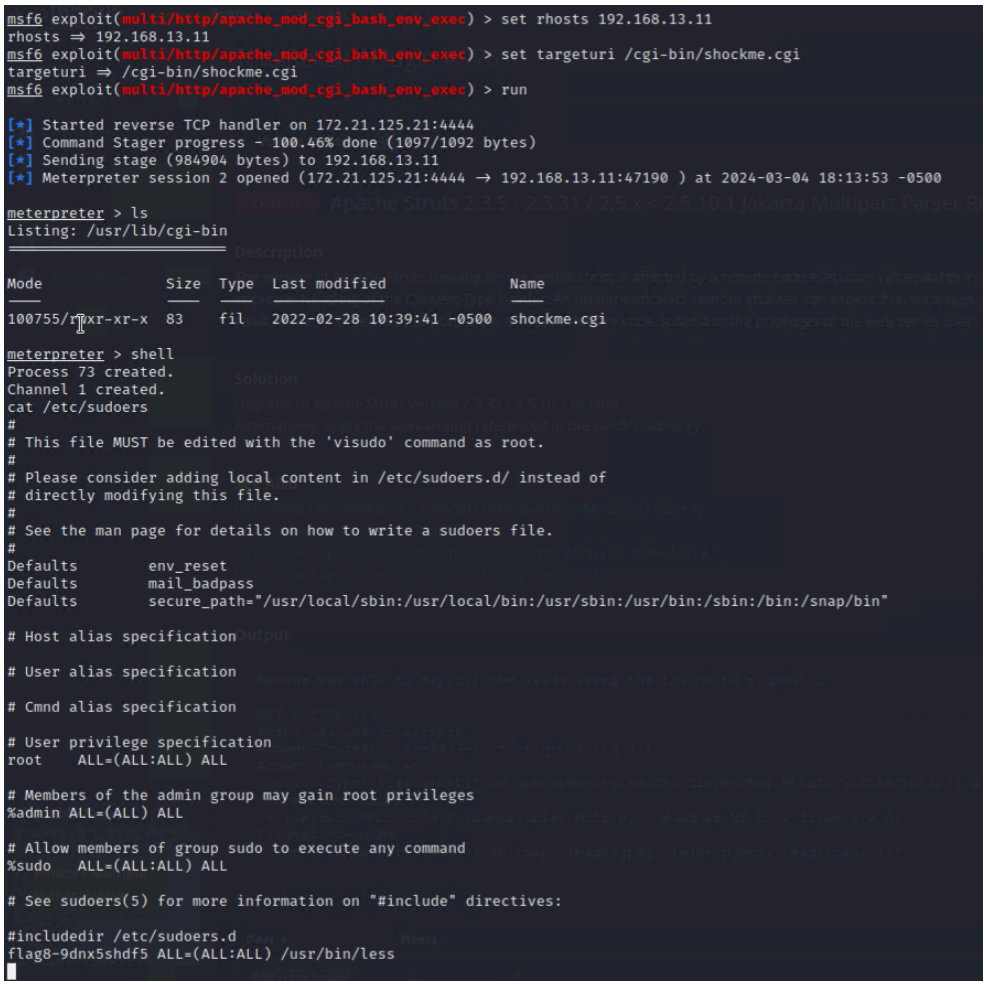
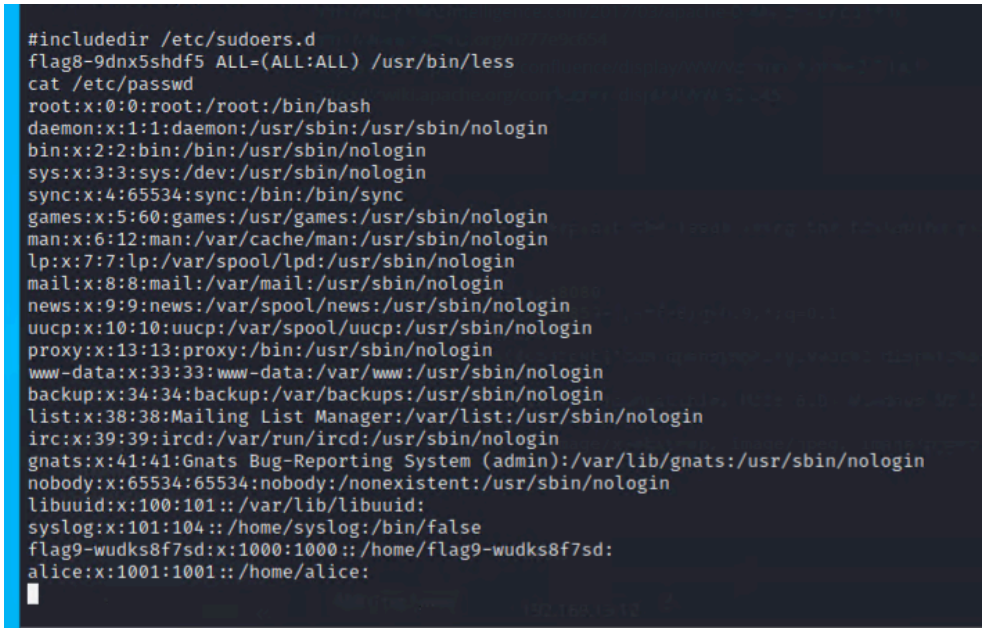
	File System Hardening: Restrict access permissions on sensitive files and directories, such as `/etc/passwd`, to prevent unauthorized access. Ensure that the web server process runs with minimal privileges necessary to perform its tasks and does not have unnecessary access to system files. Regular Security Audits and Code Reviews: Conduct regular security audits and code reviews of your PHP application to identify and address potential vulnerabilities, including PHP injection vulnerabilities. Look for insecure coding practices, unvalidated input handling, and potential code execution points. Web Application Firewall (WAF): Implement a web application firewall (WAF) that can detect and block malicious HTTP requests, including those attempting PHP injection attacks. Configure the WAF to filter and sanitize input data and block requests that contain suspicious or malicious patterns. Security Headers and Content Security Policy (CSP): Implement security headers like Content Security Policy (CSP) to restrict the sources from which resources (e.g., scripts, stylesheets, images) can be loaded. This can help prevent execution of injected scripts and enforce stricter security policies on the client-side.

Vulnerability 5	Findings
Title	**Apache Tomcat Remote Code Execution**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	Executed the payload tomcan_jsp_upload_bypass via metasploit to gain a successful root shell as you can see in the image below.

Images	
Affected Hosts	192.168.13.10
Remediation	Keep Apache Tomcat Updated: Regularly update Apache Tomcat to the latest version available. Developers frequently release security patches and updates to address vulnerabilities and improve the security of the software. Apply Security Patches: Apply security patches and updates provided by Apache Tomcat promptly after their release. Stay informed about security advisories and vulnerabilities related to Apache Tomcat and take necessary actions to address them. Limit Access: Restrict access to the Apache Tomcat server to only authorized users and applications. Implement strong authentication mechanisms, such as using strong passwords, multi-factor authentication (MFA), and restricting access based on IP addresses or network segments. Disable Unused Services and Applications: Disable or remove any unused services, applications, or components from the Apache Tomcat server to reduce the attack surface. Only enable the features and functionalities that are necessary for your specific use case. Implement Network Security Measures: Implement network security measures, such as firewalls, intrusion detection and prevention systems (IDPS), and network segmentation, to monitor and control traffic to and from the Apache Tomcat server. Configure firewalls to allow only necessary traffic and block unauthorized access attempts. Enable Secure Configuration: Configure Apache Tomcat securely by following best practices and recommendations provided by Apache Tomcat documentation and security guidelines. Ensure that default configurations are updated and

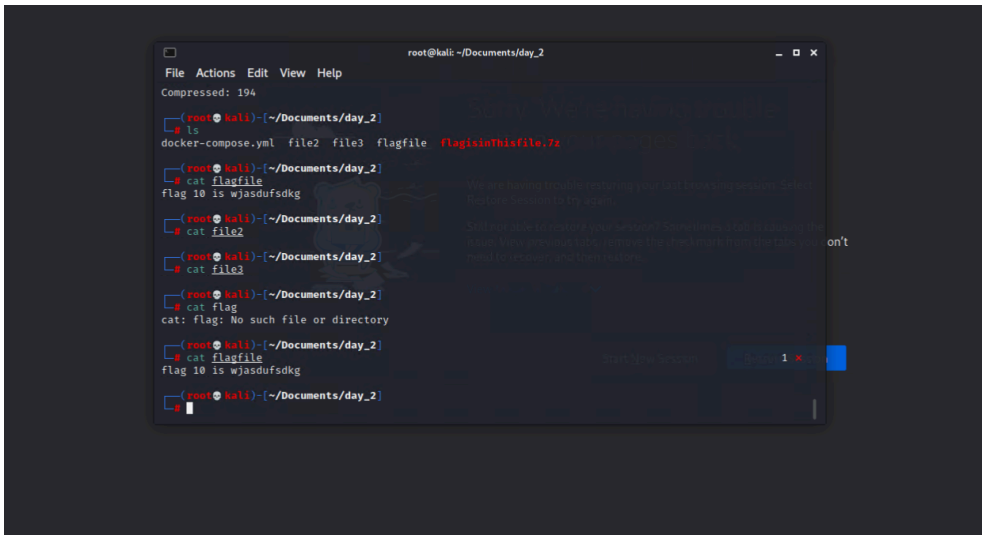
	hardened to prevent common security issues. Regular Security Audits and Vulnerability Scanning: Conduct regular security audits and vulnerability scanning of the Apache Tomcat server to identify and address potential security weaknesses and vulnerabilities. Perform penetration testing to evaluate the effectiveness of security controls and measures. Monitor Server Logs: Monitor Apache Tomcat server logs for any unusual or suspicious activities, such as unauthorized access attempts, unusual traffic patterns, and errors related to remote code execution. Enable logging and configure log management systems to centralize and analyze logs effectively. Security Awareness and Training: Provide security awareness training to system administrators, developers, and users to educate them about security best practices, common vulnerabilities, and the importance of maintaining a secure Apache Tomcat server environment.

Vulnerability 6	Findings
Title	**Shellshock**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	ZDCA was able to establish a meterpreter session using apache_mod_cgi_bash_env_exec which was then able to gain a shell to display the sudoers file and the paasswd file from the /etc/ directory

Images	 
Affected Hosts	192.168.13.11
Remediation	Update Bash: Ensure that Bash, the Unix shell interpreter, is updated to the latest version available. Vulnerabilities like Shellshock are often patched in newer

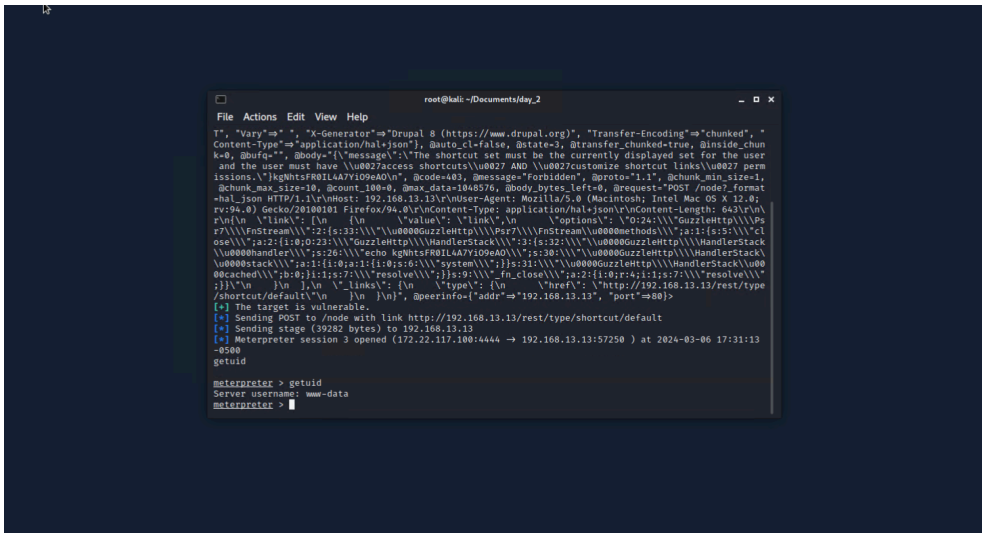
	versions of Bash. Regularly check for updates and apply them promptly. Patch Vulnerable Systems: Patch all vulnerable systems, including servers and workstations, to address the Shellshock vulnerability. Check with your operating system vendor or distribution provider for patches specific to your environment. Update CGI Scripts: If your web server uses CGI scripts, ensure that any Bash scripts invoked by CGI are updated to avoid Shellshock vulnerabilities. Review and update any custom or third-party CGI scripts that may be affected. Use ModSecurity Rules: Implement ModSecurity rules to detect and block HTTP requests attempting to exploit Shellshock. ModSecurity is an open-source web application firewall (WAF) that can help protect against various web-based attacks, including Shellshock. Filter Input: Filter and sanitize user-supplied input to prevent injection attacks. This is particularly important for applications and web services that accept input from external sources, as Shellshock exploits can occur through HTTP headers, cookies, and other input mechanisms. Implement Web Application Firewalls (WAF): Deploy a web application firewall (WAF) to monitor and filter HTTP traffic for suspicious activity, including attempts to exploit Shellshock. Configure the WAF to block requests that contain known attack patterns associated with Shellshock. Network Segmentation: Implement network segmentation to limit the impact of a successful Shellshock exploit. Segment your network into separate zones with access controls and firewall rules to restrict lateral movement by attackers. Monitor System Logs: Monitor system logs, including web server logs and system logs, for any signs of Shellshock exploitation attempts. Look for unusual or suspicious activity that may indicate an attempted exploit, such as unexpected commands being executed or abnormal HTTP requests. Security Awareness Training: Educate system administrators, developers, and users about the Shellshock vulnerability and its potential impact. Provide training on secure coding practices, vulnerability management, and incident response

	procedures to help prevent and respond to Shellshock exploits effectively.

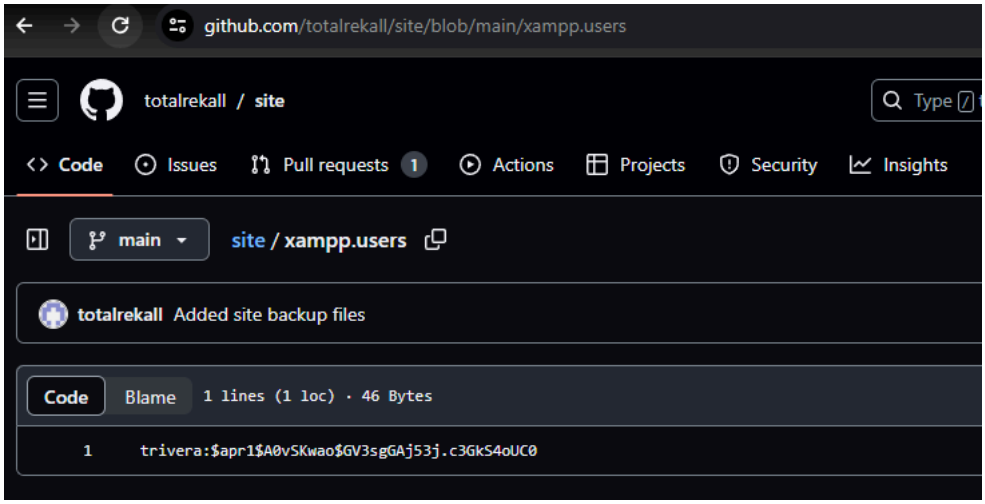
Vulnerability 7	Findings
Title	**Struts - CVE-2017-5638**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	ZDCA was able to establish a meterpreter session then able to extract sensitive files back to our kali machine for analysis.
Images	
Affected Hosts	192.168.13.12
Remediation	Patch Struts Framework: Update your Apache Struts framework to a patched version that addresses the CVE-2017-5638 vulnerability. You can download the latest patched version from the Apache Struts website or use your package manager to update the framework. Apply Vendor Patches: Check with your vendor or distribution provider for any patches or updates related to the CVE-2017-5638 vulnerability. Apply patches provided by the vendor to ensure that your system is protected against known vulnerabilities. Review and Update Dependencies: Review and update any third-party dependencies or libraries used by your application, including those related to Apache Struts. Ensure that all dependencies are updated to versions that do not contain vulnerabilities or

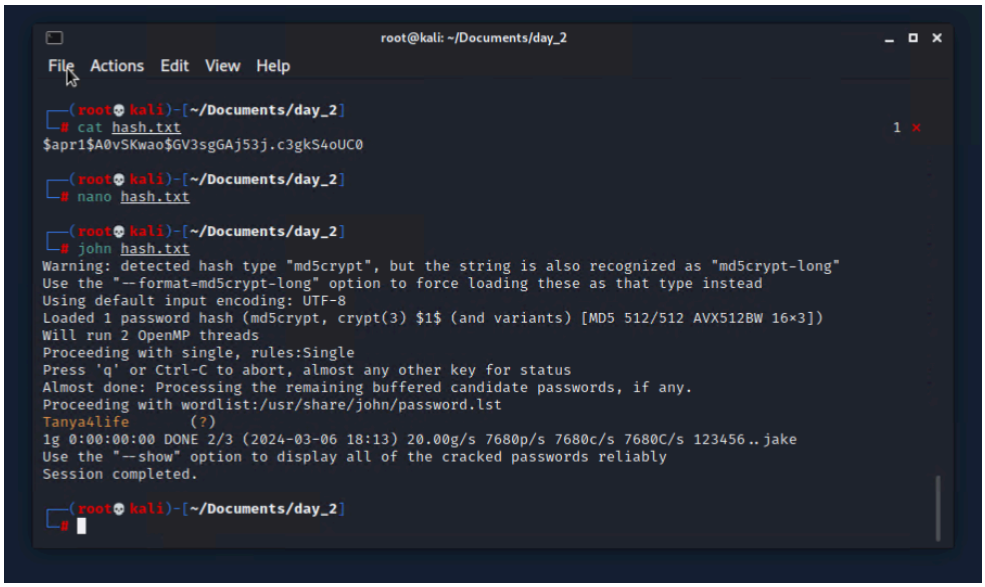
	are patched against known vulnerabilities. Implement Web Application Firewall (WAF): Deploy a web application firewall (WAF) to protect your web applications, including those built with Apache Struts, from common web-based attacks. Configure the WAF to filter and block malicious HTTP requests that exploit vulnerabilities like CVE-2017-5638. Network Segmentation: Implement network segmentation to limit the impact of a successful exploitation of the CVE-2017-5638 vulnerability. Segment your network into separate zones with access controls and firewall rules to restrict lateral movement by attackers. Monitor and Audit Logs: Enable logging and auditing features in your web server and application server to monitor and review user activities and system events. Regularly review logs for any suspicious or unauthorized access attempts and take appropriate actions if anomalies are detected. Security Awareness Training: Provide security awareness training to system administrators, developers, and users to educate them about the CVE-2017-5638 vulnerability and other potential security risks. Train them on best practices for securely managing and using web applications built with Apache Struts. Regular Security Audits: Conduct regular security audits and vulnerability assessments of your web applications built with Apache Struts to identify and address any new vulnerabilities or misconfigurations. Perform penetration testing to simulate real-world attack scenarios and assess the effectiveness of your security controls.

Vulnerability 8	Findings
Title	**Drupal - CVE-2019-6340**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	ZDCA was able to gain a meterpreter session on the 192.168.13.13 machine.

Images	
Affected Hosts	192.168.13.13
Remediation	Apply Security Updates: Update your Drupal installation to the latest patched version that addresses the CVE-2019-6340 vulnerability. You can download the latest release from the official Drupal website or use Drupal's built-in update mechanism to apply the patch. Update Contributed Modules: Update any contributed modules installed on your Drupal website to their latest versions. Some vulnerabilities may exist in third-party modules, so it's essential to keep them updated to mitigate potential risks. Disable or Limit Affected Functionality: If the vulnerable functionality is not essential to your website's operation, consider disabling it until a patch is available or until you can apply the necessary updates. This can help reduce the attack surface and mitigate the risk of exploitation. Implement Web Application Firewall (WAF): Deploy a web application firewall (WAF) to protect your Drupal website from common web-based attacks, including those targeting vulnerabilities like CVE-2019-6340. Configure the WAF to filter and block malicious HTTP requests that exploit known vulnerabilities. Network Segmentation: Implement network segmentation to limit the impact of a successful exploitation of the CVE-2019-6340 vulnerability. Segment your network into separate zones with access controls and firewall rules to restrict lateral movement by attackers. Monitor and Audit Logs:

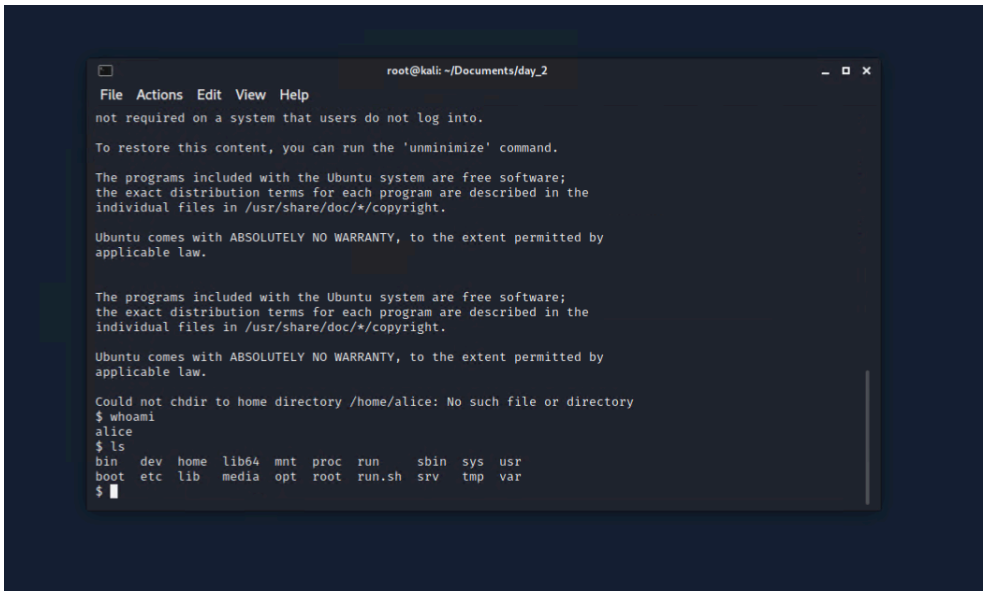
	Enable logging and auditing features in your web server and application server to monitor and review user activities and system events. Regularly review logs for any suspicious or unauthorized access attempts and take appropriate actions if anomalies are detected. Security Awareness Training: Provide security awareness training to system administrators, developers, and users to educate them about the CVE-2019-6340 vulnerability and other potential security risks. Train them on best practices for securely managing and using the Drupal website. Regular Security Audits: Conduct regular security audits and vulnerability assessments of your Drupal website to identify and address any new vulnerabilities or misconfigurations. Perform penetration testing to simulate real-world attack scenarios and assess the effectiveness of your security controls.

Vulnerability 9	Findings
Title	**Open Source Exposed Data (Multiple Occurances)**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Critical**
Description	ZDCA preformed a simple search for “aTotal Rekall”, when finding the github we saw that there was a public repository. With this information we investigated the repository and found sensitive information in the form of username and password giving us valid credentials to the system.
Images	

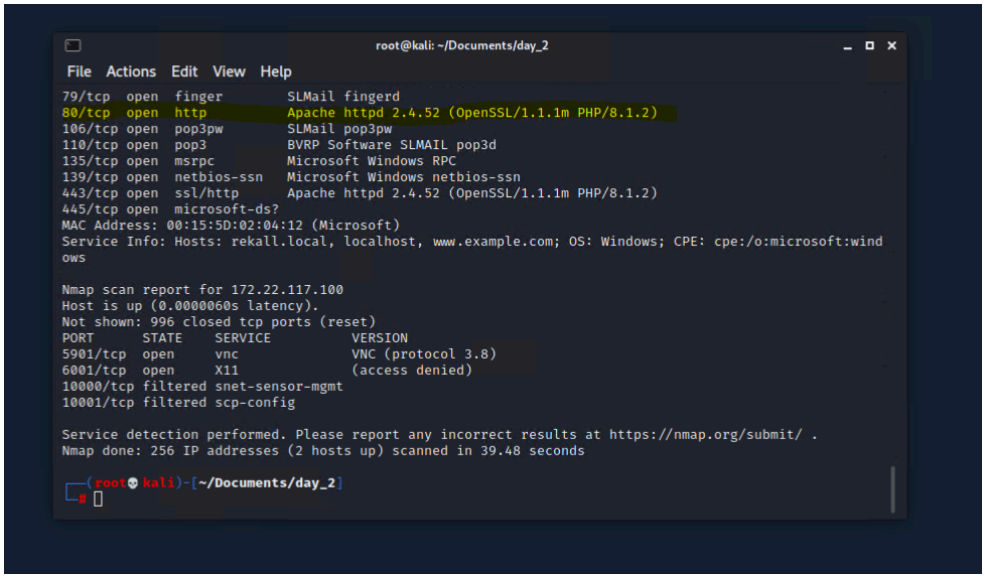
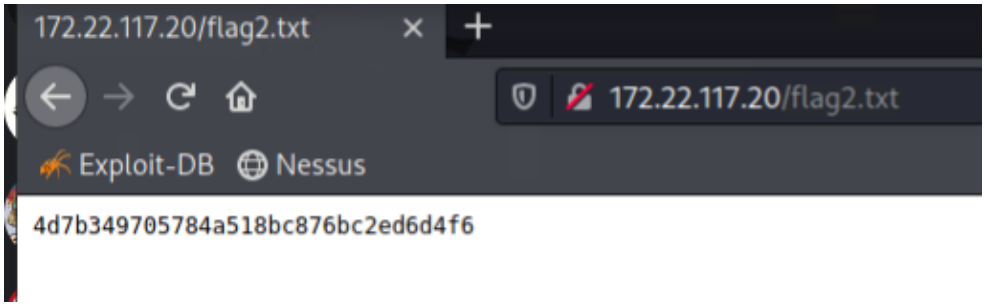


Affected Hosts	https://github.com/totalrekall/site/blob/main/xampp.users
Remediation	Immediate Removal: Quickly remove or restrict access to the specific file (`xampp.users`) containing sensitive information from the public repository. This can be done by deleting the file or moving it to a private repository. Password Resets: Since the password hashes have been exposed (and potentially cracked), it's critical to reset all affected passwords immediately. Notify all impacted users and require them to change their passwords as soon as possible. Ensure the new passwords are strong and unique. Review and Audit: Conduct a thorough review of the entire repository and any other public repositories under the Total Rekall organization for additional exposed sensitive information. Look for API keys, secret tokens, configuration files, and any other data that should not be publicly accessible. Implement Access Controls: Restrict access to sensitive information by implementing proper access controls. Use private repositories for storing any data that should not be publicly available. Leverage GitHub's branch protection rules to prevent unauthorized modifications to sensitive files. Use .gitignore: Utilize the `.gitignore` file to prevent accidental commits of sensitive files. Explicitly list files and directories that should never be uploaded to the repository, such as configuration files, backup files, and any files containing sensitive data. Scan for Sensitive Data: Use tools designed to scan repositories for sensitive information, such as GitGuardian, to identify and mitigate the risks of data exposure. Integrate these tools into your development process to catch issues before they are pushed to public repositories. Educate and Train:

	Educate your team about the risks of exposing sensitive data. Train them on best practices for handling sensitive information, secure coding practices, and the proper use of version control systems. Rotate Secrets: If any API keys, secret tokens, or other credentials were exposed, rotate them immediately. Update your applications and services with the new credentials as soon as possible. Incident Response Plan: Develop or update your incident response plan to include steps for dealing with exposed data in public repositories. This plan should include immediate actions, internal and external communication strategies, and preventative measures. Legal and Compliance Review: Consult with legal and compliance teams to understand any implications of the data exposure. There may be regulatory reporting requirements depending on the nature of the exposed data and the jurisdictions affected.

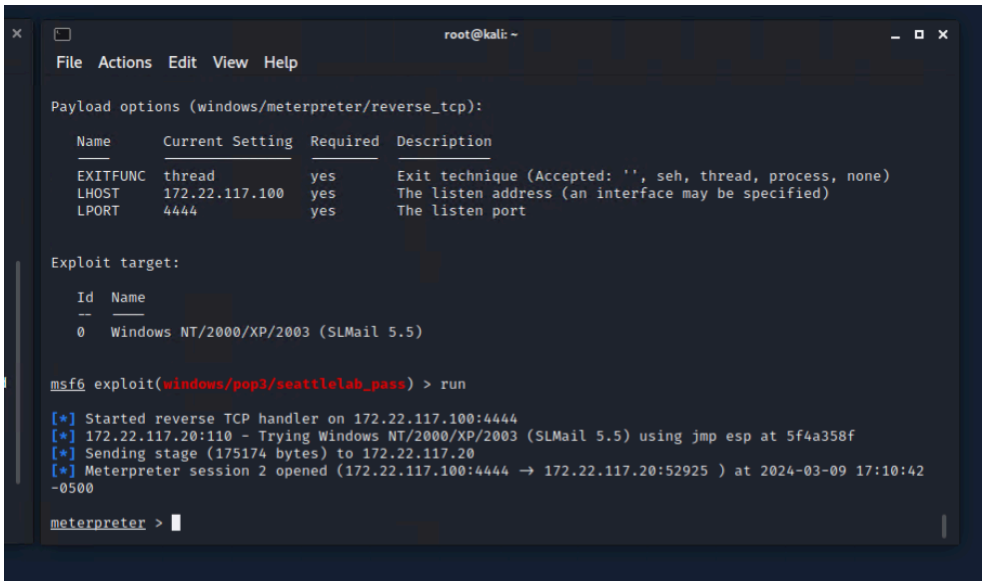
Vulnerability 10	Findings
Title	**CVE-2019-14287**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	ZDCA was able to successfully crack passwords resulting in unauthorized sign in to the use “Alice”
Images	
Affected Hosts	192.168.13.14

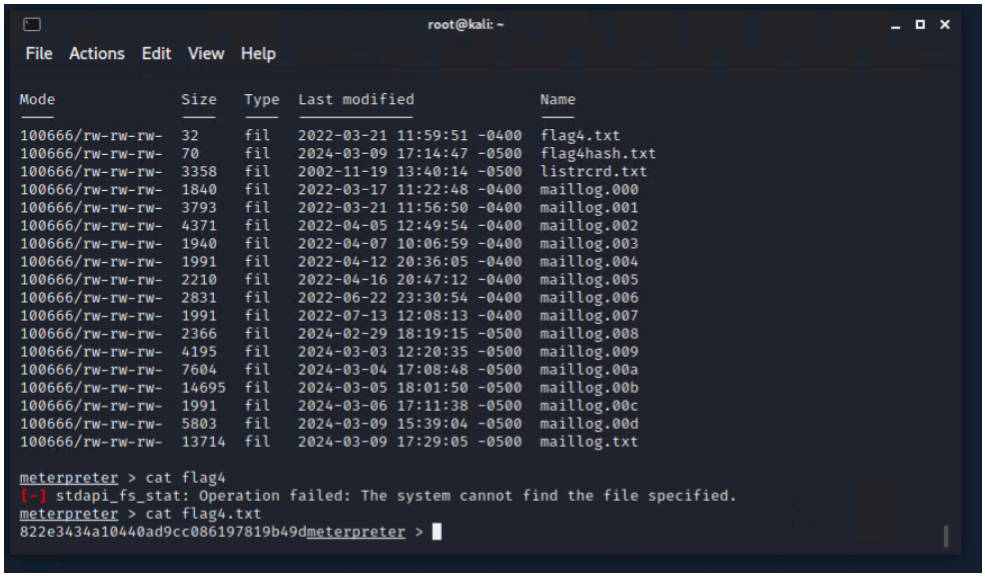
Remediation	Apply Security Updates: Ensure that your operating system and sudo utility are updated to the latest patched versions available. Check with your operating system vendor or distribution provider for patches specific to CVE-2019-14287 and apply them promptly. Review sudo Configuration: Review the sudo configuration files (usually located in `/etc/sudoers` or `/etc/sudoers.d/`) to ensure that privilege escalation rules are configured correctly and securely. Specifically, check for any wildcard entries or rules that may allow unauthorized users to escalate privileges. Remove Unnecessary sudo Access: Limit sudo access to only authorized users and commands necessary for their roles and responsibilities. Remove any unnecessary sudo privileges that could potentially be exploited for privilege escalation. Implement Least Privilege Principle: Follow the principle of least privilege when assigning sudo privileges. Grant users the minimum level of permissions required to perform their tasks effectively. Avoid granting unnecessary root or administrative privileges. Use Strong Passwords: Enforce strong password policies for all user accounts, including the root account and other accounts with sudo privileges. Encourage users to create complex passwords that are resistant to brute force attacks. Monitor sudo Logs: Enable sudo logging to monitor and audit sudo commands executed by users. Regularly review sudo logs for any suspicious or unauthorized sudo activity, such as repeated failed authentication attempts or unusual commands being executed. Implement Two-Factor Authentication (2FA): Implement two-factor authentication (2FA) for sudo access to add an extra layer of security. Require users to authenticate using a second factor, such as a one-time code sent to their mobile device, in addition to their password. Regular Security Audits: Conduct regular security audits and vulnerability assessments of your system to identify and address any potential vulnerabilities, including misconfigurations in sudo privileges. Perform penetration testing to assess the effectiveness of your security controls.

Vulnerability 11	Findings
Title	**Apache httpd 2.4.52**
Type (Web app / Linux OS / WIndows OS)	Linux OS
Risk Rating	**Critical**
Description	Using this vulnerability we were able to input the ip into the web browser giving us the Index page containing sensitive information.
Images	  

Affected Hosts	172.22.117.20
Remediation	Patch and Update Apache HTTP Server: Update your Apache HTTP server to the latest patched version available. Check the official Apache website or your operating system's package manager for updates. Patching ensures that known vulnerabilities are addressed, reducing the risk of exploitation. Security Configuration Review: Review the configuration settings of your Apache HTTP server to ensure that it follows security best practices. Disable or restrict access to sensitive directories and files. Use access controls, such as Apache's `.htaccess` files or server configuration directives, to restrict access appropriately. Implement Web Application Firewall (WAF): Deploy a web application firewall (WAF) in front of your Apache HTTP server to filter and block malicious HTTP requests. Configure the WAF to detect and block requests that exploit known vulnerabilities, including those targeting the Apache HTTP server. Regular Security Updates: Keep your operating system and software packages up to date with the latest security updates and patches. Vulnerabilities in other components, such as the OpenSSL library and PHP, can also impact the security of your Apache HTTP server. Remove Sensitive Information from Index Pages: Review your web server's index pages and ensure that they do not expose sensitive information. Customize the default index pages to display only necessary information and avoid disclosing sensitive details about your server configuration. Security Headers and Content Security Policy (CSP): Implement security headers, such as Content Security Policy (CSP), to restrict the sources from which resources (e.g., scripts, stylesheets, images) can be loaded. This can help prevent the execution of malicious scripts and mitigate the risk of attacks targeting your Apache HTTP server. Regular Security Audits and Vulnerability Scanning: Conduct regular security audits and vulnerability scanning of your Apache HTTP server to identify and address potential security weaknesses and misconfigurations. Perform penetration testing to assess the

	effectiveness of your security controls. Security Awareness Training: Provide security awareness training to system administrators and web developers responsible for managing and maintaining the Apache HTTP server. Educate them about security best practices and the importance of keeping the server secure.

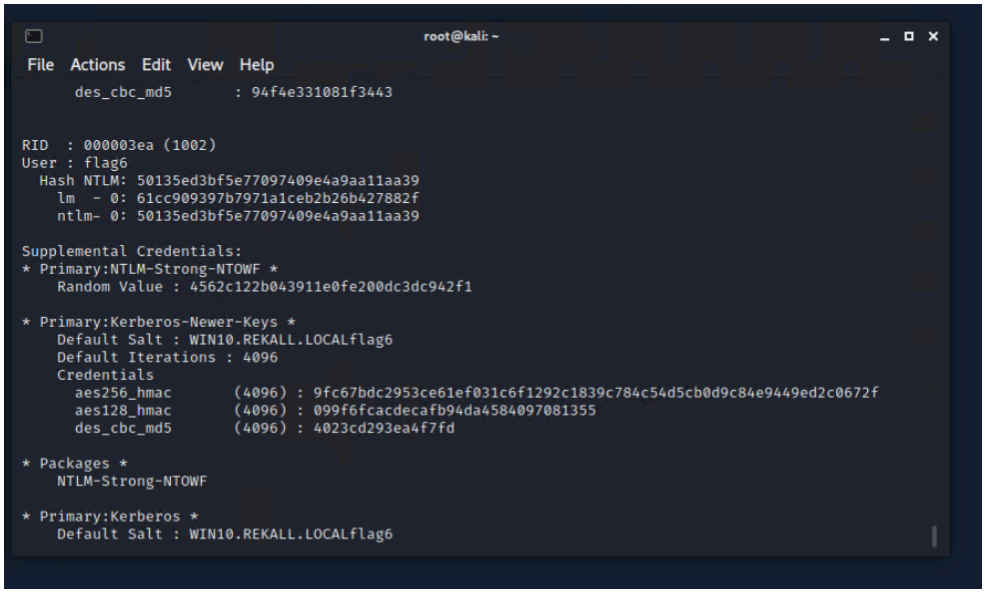
Vulnerability 12	Findings
Title	**Windows/pop3/seattlelab_pass (Multiple occurrences) SLMail SMTP 25**
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**Critical**
Description	ZDCA was about to successfully complete the exploit resulting in a meterpreter session being established. We were then able to see and access sensitive information.
Images	



Affected Hosts	172.22.117.20
Remediation	Patch and Update SLMail Server: Ensure that your SLMail server is updated to the latest patched version available. Check the official vendor website or contact the vendor for updates and patches that address known vulnerabilities. Patching the server will help fix security vulnerabilities and reduce the risk of exploitation. Implement Access Controls: Configure access controls on your SLMail server to restrict access to authorized users only. Implement strong authentication mechanisms, such as username and password authentication, and consider implementing multi-factor authentication (MFA) for added security. Network Segmentation: Implement network segmentation to limit the impact of a successful exploitation of the SLMail vulnerability. Segment your network into separate zones with access controls and firewall rules to restrict lateral movement by attackers. Monitor and Audit Logs: Enable logging and auditing features on your SLMail server to monitor and review user activities, login attempts, and system events. Regularly review logs for any suspicious or unauthorized access attempts and take appropriate actions if anomalies are detected. Security Awareness Training: Provide security awareness training to system administrators and users responsible for managing and using the SLMail server. Educate them about security best practices, including the importance of using strong

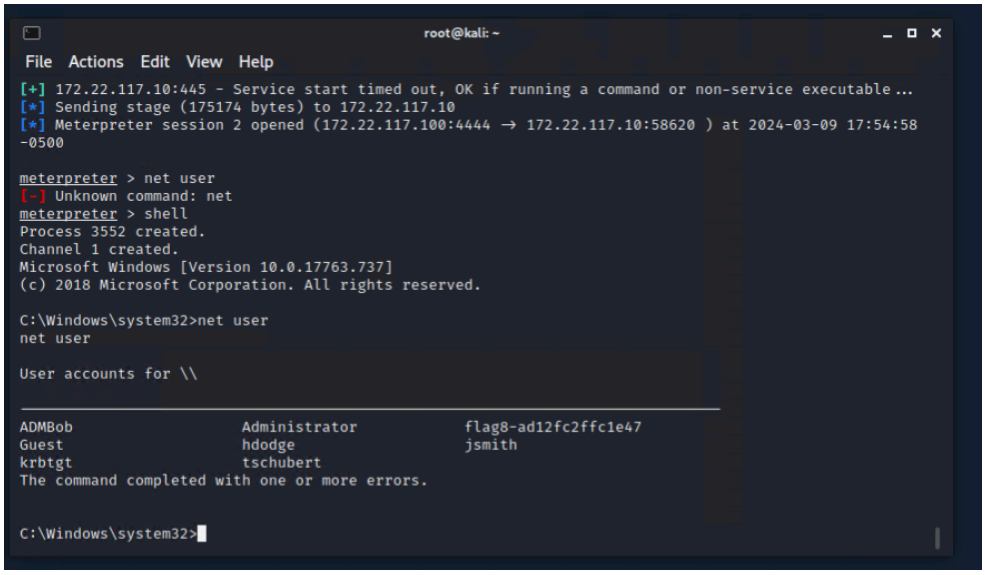
	passwords, avoiding phishing attacks, and reporting suspicious activities. Security Updates for Operating System: Ensure that the operating system running on the host with the IP address 172.22.117.20 is also updated with the latest security patches and updates. Vulnerabilities in the underlying operating system can also pose security risks to the SLMail server. Regular Security Audits and Vulnerability Scanning: Conduct regular security audits and vulnerability scanning of your SLMail server to identify and address potential security weaknesses and misconfigurations. Perform penetration testing to assess the effectiveness of your security controls. Implement Intrusion Detection Systems (IDS): Deploy intrusion detection systems (IDS) to monitor network traffic and detect any suspicious or malicious activities targeting your SLMail server. Configure the IDS to alert administrators of potential security incidents in real-time.

Vulnerability 13	Findings
Title	**Windows/pop3/seattlelab_pass(Multiple occurrences)**
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**Critical**
Description	ZDCA was able to successfully complete the seattlelab_pass exploit gaining a meterpreter session, using this we were able to obtain user information in the form of a password hash, using the tool john the ripper we were able to crack this hash giving us valid credentials to the system.

Images	 
Affected Hosts	172.22.117.20
Remediation	Update and Patch: Ensure that the system (host IP address 172.22.117.20) is fully patched and up-to-date with the latest security updates provided by the operating system vendor. Patching helps address known vulnerabilities and improves the overall security posture of the system. Secure Configuration: Review and secure the configuration of the POP3 (Post Office Protocol version 3) server running on the system. Disable unnecessary services, ports, or features that are not required for the system's operation to minimize the attack surface. Strong Password Policies:

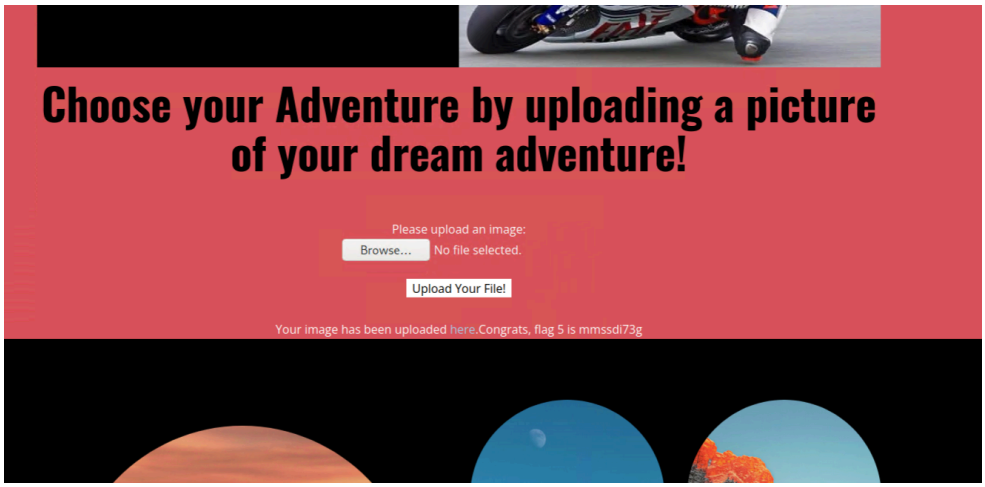
	Enforce strong password policies on the system to prevent easy password cracking. Require users to create complex passwords that include a mix of uppercase and lowercase letters, numbers, and special characters. Implement password expiration and account lockout policies to enhance security. Network Segmentation: Implement network segmentation to isolate critical systems from less secure networks or segments. Segment the network to restrict access to sensitive systems and resources, including the system hosting the POP3 server. Intrusion Detection and Prevention: Deploy intrusion detection and prevention systems (IDPS) to monitor network traffic and detect suspicious or malicious activities targeting the system. Configure the IDPS to alert administrators of potential security incidents in real-time. Two-Factor Authentication (2FA): Implement two-factor authentication (2FA) for accessing sensitive systems and services, including the POP3 server. Require users to provide an additional authentication factor, such as a one-time code sent to their mobile device, in addition to their password. Regular Security Audits: Conduct regular security audits and vulnerability assessments of the system to identify and address any security weaknesses or misconfigurations. Perform penetration testing to assess the effectiveness of your security controls. Security Awareness Training: Provide security awareness training to system administrators and users to educate them about the risks of unauthorized access and the importance of following security best practices, such as using strong passwords and safeguarding credentials.

Vulnerability 14	Findings
Title	**Windows/smb/psexec**
Type (Web app / Linux OS / WIndows OS)	Windows OS

Risk Rating	Critical
Description	Targeting the windows system leveraging the SMB protocol we were able to gain unauthorized access to a second machine on the network after obtaining and using cracked credentials.
Images	
Affected Hosts	172.22.117.10
Remediation	Patch and Update Systems: Ensure that all systems, including the host with IP address 172.22.117.10 and the second machine on the network, are fully patched and up-to-date with the latest security updates. Regularly apply patches provided by the operating system vendor to address known vulnerabilities. Password Management: Enforce strong password policies on all systems to prevent easy password cracking. Require users to create complex passwords that include a mix of upper and lower case letters, numbers, and special characters. Additionally, implement password expiration policies and encourage regular password changes. Implement Multi-Factor Authentication (MFA): Enable multi-factor authentication (MFA) on systems and services to add an extra layer of security. MFA requires users to provide two or more forms of authentication (e.g., password and a one-time code sent to their mobile device) before accessing resources, making it harder for attackers to gain unauthorized access. Network Segmentation: Implement network segmentation to limit the impact of a successful exploit. Segment the network into separate zones with access controls and firewall rules to restrict lateral movement by attackers. Ensure that

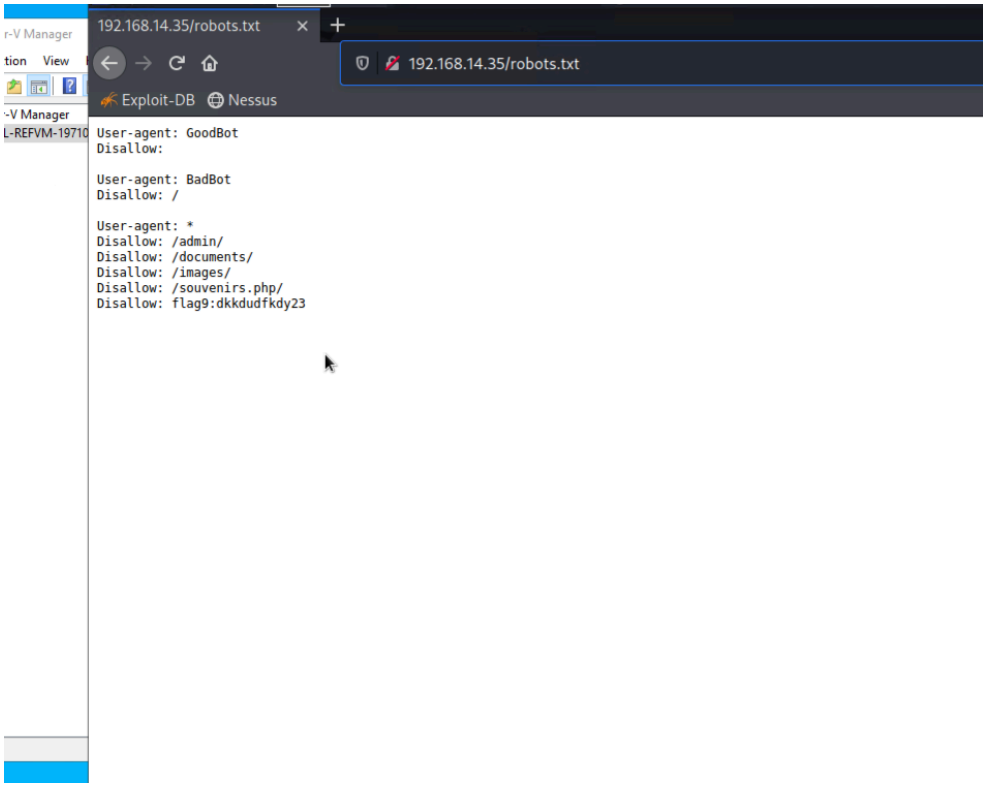
	systems are only accessible from authorized networks or subnets. Monitor and Detect Suspicious Activity: Deploy intrusion detection systems (IDS) or intrusion prevention systems (IPS) to monitor network traffic for suspicious activity and detect potential exploit attempts. Configure these systems to alert administrators of any unusual behavior or unauthorized access attempts. Limit Administrative Privileges: Follow the principle of least privilege and limit administrative privileges to only those users who require them for their roles. Avoid using privileged accounts for everyday tasks and implement separate accounts for administrative activities. Educate Users about Security Best Practices: Provide security awareness training to users and administrators to educate them about common attack techniques, such as password cracking and exploiting vulnerabilities like windows/smb/psexec. Train them on best practices for password management, recognizing phishing attempts, and securely managing credentials. Regular Security Audits and Penetration Testing: Conduct regular security audits and penetration testing of systems to identify and address potential security weaknesses and misconfigurations. Perform vulnerability assessments to proactively identify and remediate vulnerabilities before they can be exploited.

Vulnerability 15	Findings
Title	**Local File Inclusion**
Type (Web app / Linux OS / WIndows OS)	Web application
Risk Rating	**High**
Description	ZDCA was able to successfully upload “script.php” to the section on the web application that is intended to upload images for there “adventure”

Images	
Affected Hosts	192.168.14.35/Memory-Planner.php
Remediation	Input Validation and Sanitization: Implement strict input validation and sanitization mechanisms within the application. Ensure that user-supplied input, such as filenames, is properly validated and sanitized to prevent malicious file uploads. File Upload Restrictions: Implement file upload restrictions to allow only specific file types and limit the size of uploaded files. Validate file extensions and content types to prevent users from uploading executable scripts or other dangerous file types. Secure File Storage: Store uploaded files in a secure location outside of the web root directory. This prevents direct access to uploaded files by users and reduces the risk of unauthorized file inclusion. Access Controls: Implement access controls to restrict access to sensitive directories and files. Ensure that only authorized users have permission to upload files and access specific sections of the application. Use Whitelists for File Inclusion: Instead of directly including files based on user-supplied input, use whitelists or predefined lists of allowed files to include. This prevents attackers from including arbitrary files on the server. Disable PHP File Inclusion: Disable the use of dynamic file inclusion functions in PHP, such as `include()` and `require()`, for including user-supplied files. Use safer alternatives, such as `readfile()` or `file_get_contents()`, with

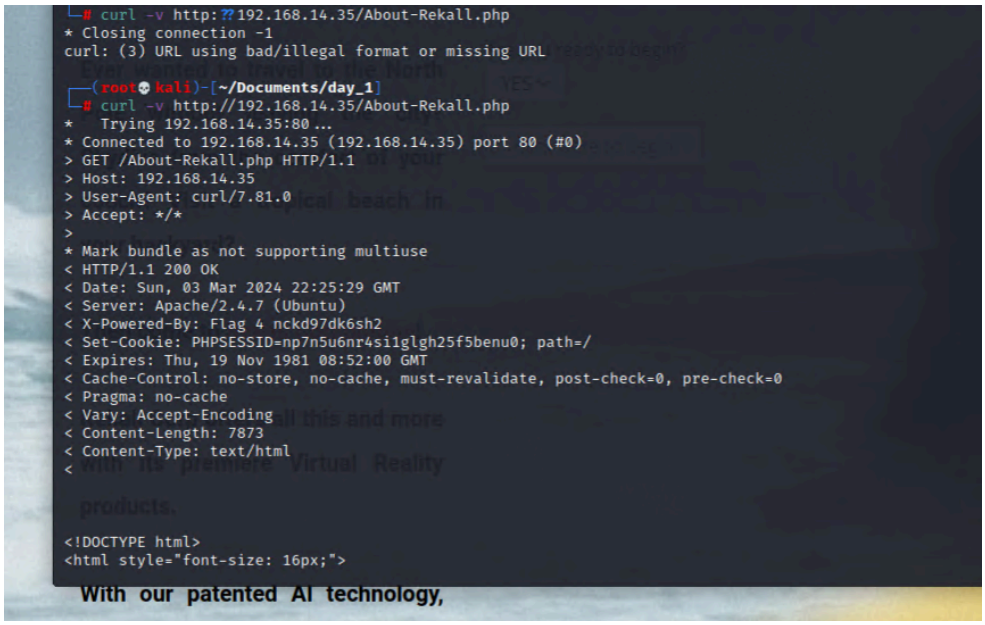
	carefully controlled file paths. Security Headers: Implement security headers, such as Content Security Policy (CSP), to mitigate the risk of Cross-Site Scripting (XSS) attacks and prevent malicious file uploads through client-side vulnerabilities. Regular Security Audits: Conduct regular security audits and vulnerability assessments of the application to identify and address any security weaknesses, including Local File Inclusion vulnerabilities. Perform code reviews and automated security testing to detect and fix vulnerabilities early in the development lifecycle. Security Awareness Training: Provide security awareness training to developers and users to educate them about secure coding practices, file upload vulnerabilities, and the importance of maintaining security in web applications.

Vulnerability 16	Findings
Title	**Sensitive Data Exposure**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**High**
Description	ZDCA was able to input robots.txt into the address field which was obtained through a google search revealing the contents of robots.txt

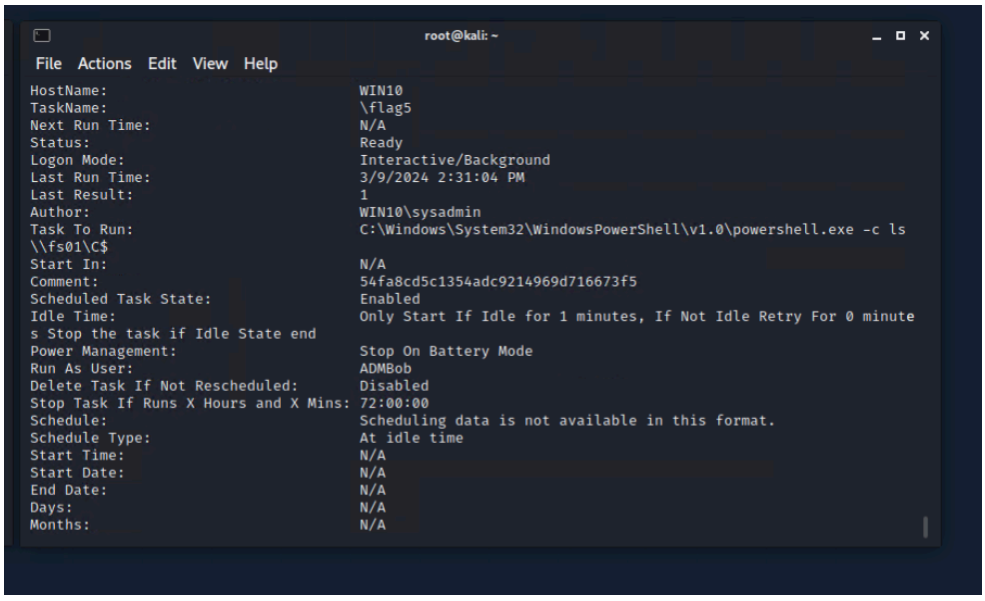
Images	
Affected Hosts	192.168.4.35/robots.txt
Remediation	Review and Update robots.txt: Review the contents of the robots.txt file to ensure that it does not contain sensitive information that should not be publicly accessible. Remove any sensitive or confidential directives from the robots.txt file. Restrict Access to robots.txt: Configure the web server to restrict access to the robots.txt file. You can use server-side configuration (e.g., .htaccess file for Apache servers) to deny access to the robots.txt file from unauthorized users or bots. Implement Authentication: Implement authentication mechanisms to restrict access to sensitive files, including the robots.txt file. Configure the web server to require authentication (e.g., username and password) for accessing the robots.txt file. Use Disallow Directive Carefully: If the robots.txt file contains directives to disallow access to certain directories or files, ensure that these directives are used carefully and do not inadvertently expose sensitive information. Review and update the disallow directives as necessary. Regularly Monitor and Update: Regularly monitor the contents of the robots.txt file for any changes or

	updates. Update the robots.txt file as needed to reflect changes in your website's content and structure, and to ensure that sensitive information is not inadvertently exposed. Security Awareness Training: Provide security awareness training to website administrators and developers to educate them about the importance of properly managing and securing sensitive information, including the contents of the robots.txt file. Review Web Application Security: Perform a comprehensive security review of your web application to identify and address any other potential sensitive data exposure issues. This may include reviewing access controls, data encryption practices, and other security measures. Regular Security Audits: Conduct regular security audits and vulnerability assessments of your web application to identify and address any security weaknesses or vulnerabilities, including sensitive data exposure issues.

Vulnerability 17	Findings
Title	**Sensitive Data Exposure (HTTP HEADERS via Curl/BURP)**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**High**
Description	When using curl command against the host it provides us with sensitive information.

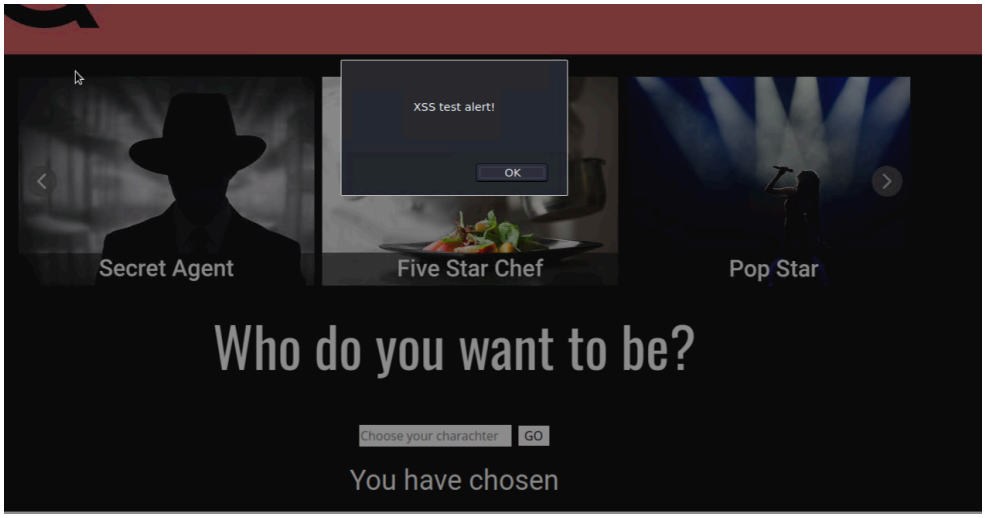
Images	
Affected Hosts	192.168.14.35/About-Rekall.php
Remediation	Review and Sanitize HTTP Headers: Review the HTTP headers being sent by the server and ensure that sensitive information is not exposed. Remove or sanitize any headers that contain sensitive data, such as cookies, tokens, or server information. Server Configuration: Review the server configuration to ensure that sensitive headers are not being inadvertently included in the server response. Configure the server to prevent the inclusion of sensitive headers in responses to client requests. Secure Development Practices: Implement secure development practices to prevent sensitive information from being included in HTTP headers. Avoid hardcoding sensitive data in headers and use secure methods for handling sensitive information, such as encryption and hashing. HTTP Header Management: Implement proper management of HTTP headers in your application code. Ensure that headers are properly handled and validated to prevent unauthorized access to sensitive information. Regular Security Audits: Conduct regular security audits of your application to identify and address any vulnerabilities related to sensitive data exposure via HTTP headers. Perform thorough testing, including security testing, to identify and mitigate potential risks. Security Headers:

	Implement security headers, such as Content-Security-Policy (CSP), Strict-Transport-Security (HSTS), and X-Content-Type-Options, to mitigate the risk of sensitive data exposure via HTTP headers. Configure these headers to enforce security policies and prevent unauthorized access to sensitive information. HTTP Header Injection Prevention: Implement measures to prevent HTTP header injection attacks, which could potentially lead to sensitive data exposure. Validate and sanitize user input to prevent malicious manipulation of HTTP headers. Security Awareness Training: Provide security awareness training to developers and administrators to educate them about the risks of sensitive data exposure via HTTP headers and the importance of implementing secure coding practices.

Vulnerability 18	Findings
Title	**Windows/pop3/seattlelab_pass(Continued from Critical Vulnerabilities)**
Type (Web app / Linux OS / WIndows OS)	Windows OS
Risk Rating	**Medium**
Description	After establishing our meterpreter session from the critical vulnerability we outlined above in our report we were able to view all the scheduled tasks in the Windows Machine giving us the ability to view and alter these tasks.
Images	

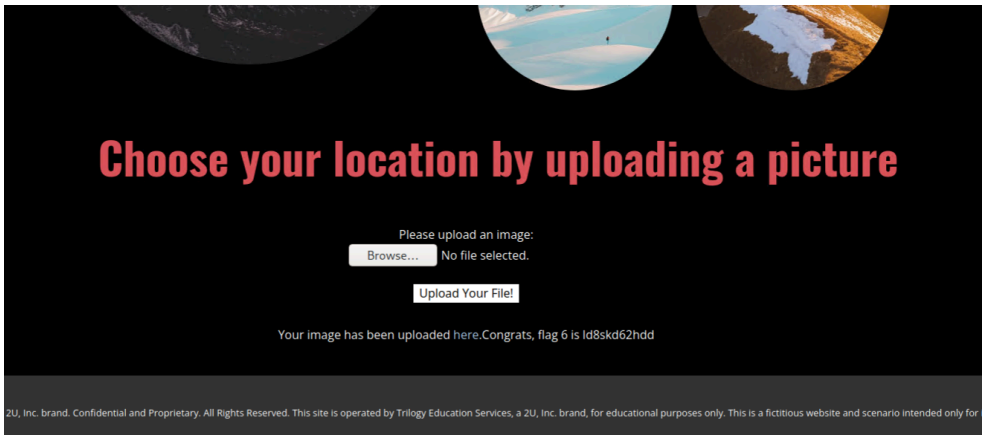
Affected Hosts	172.22.117.20
Remediation	Patch and Update: Ensure that the Windows 10 system (host IP address 172.22.117.20) is fully patched and up-to-date with the latest security updates provided by Microsoft. Regularly install security patches and updates to address known vulnerabilities and improve system security. Disable Unnecessary Services: Disable or restrict unnecessary services and protocols, including the POP3 service if it is not required for business operations. This reduces the attack surface and minimizes the risk of unauthorized access. Strong Password Policies: Enforce strong password policies for user accounts on the Windows 10 system. Require users to create complex passwords that include a mix of uppercase and lowercase letters, numbers, and special characters. Implement password expiration and account lockout policies to enhance security. Scheduled Task Permissions: Review and configure permissions for scheduled tasks on the Windows 10 system. Ensure that only authorized users have permission to view or modify scheduled tasks. Restrict access to sensitive tasks to prevent unauthorized access. Network Segmentation: Implement network segmentation to isolate critical systems, including the Windows 10 system hosting the scheduled tasks, from less secure networks or segments. Segment the network to restrict access to sensitive systems and resources. Intrusion Detection and Prevention: Deploy intrusion detection and prevention systems (IDPS) to monitor network traffic and detect suspicious or malicious activities targeting the Windows 10 system. Configure the IDPS to alert administrators of potential security incidents in real-time. Security Awareness Training: Provide security awareness training to system administrators and users to educate them about the risks of unauthorized access and the importance of following security best practices. Train users to recognize and report suspicious activities related to scheduled tasks. Regular Security Audits:

	Conduct regular security audits and vulnerability assessments of the Windows 10 system to identify and address any security weaknesses or misconfigurations. Perform penetration testing to assess the effectiveness of your security controls.

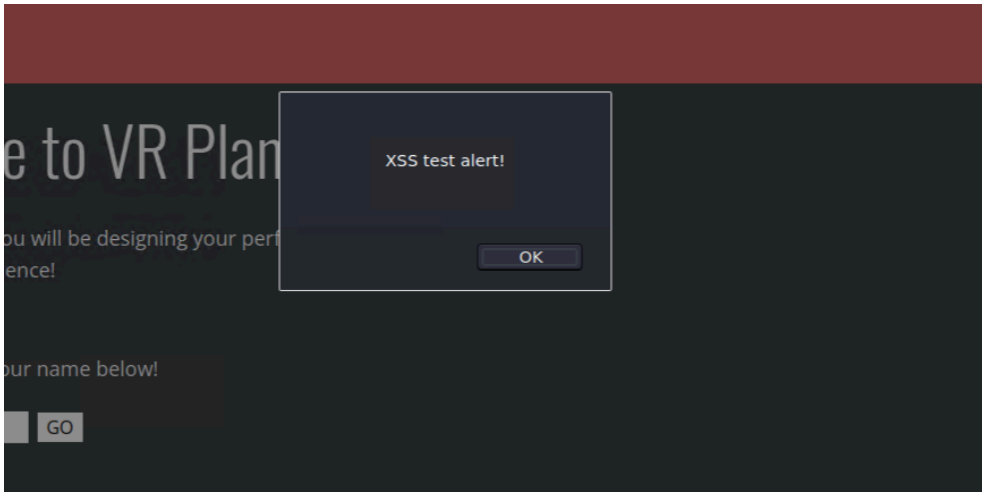
Vulnerability 19	Findings
Title	**Advanced XSS (scripting avoiding input validation)**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Medium**
Description	ZDCA was able to create an alert on the web application page displaying to us that the input validation that is in place here can be worked around. example : <SCRIPscriptT>alert('XSS test alert!')</SCRIPscriptT>
Images	
Affected Hosts	192.168.14.35/Memory-Planner.php
Remediation	Input Validation and Sanitization: Improve input validation and sanitization mechanisms in the application to prevent the execution of malicious scripts. Ensure that user inputs are properly validated and sanitized to remove or escape potentially dangerous characters, including HTML tags and JavaScript code. Content Security Policy (CSP): Implement a Content Security Policy (CSP) to mitigate the risk of XSS attacks. CSP allows you to define a whitelist of trusted sources for content, scripts, and other resources. Configure CSP directives to restrict

	the execution of inline scripts and other potentially dangerous content. Cross-Site Scripting (XSS) Protection: Enable XSS protection mechanisms provided by web application frameworks or security libraries. These protections include automatic escaping of user inputs, output encoding, and XSS filtering mechanisms to detect and block malicious scripts. Secure Coding Practices: Train developers on secure coding practices to prevent XSS vulnerabilities. Educate them about the risks of XSS attacks and the importance of properly validating and sanitizing user inputs. Encourage the use of framework-specific security features and libraries for input validation and output encoding. Regular Security Reviews: Conduct regular security reviews and code audits of the application to identify and address potential XSS vulnerabilities. Use automated scanning tools and manual code reviews to detect and remediate security weaknesses in the application code. HTTPOnly and Secure Flags: Set the HTTPOnly and Secure flags for session cookies to prevent XSS attacks from stealing session tokens. The HTTPOnly flag restricts cookie access to HTTP requests only, while the Secure flag ensures that cookies are transmitted over HTTPS connections only. Security Headers: Implement security headers, such as X-XSS-Protection, to enable browser-based XSS protection mechanisms. Configure the X-XSS-Protection header to enable XSS filtering in modern web browsers and prevent the execution of malicious scripts. Web Application Firewall (WAF): Deploy a Web Application Firewall (WAF) to provide an additional layer of protection against XSS attacks. Configure the WAF to detect and block malicious requests that attempt to exploit XSS vulnerabilities in the application.

Vulnerability 20	Findings
Title	**Local File Inclusion (Input Validation)**

Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Medium**
Description	ZDCA was able to still upload a malicious file avoiding the input validation that is currently in place looking for ".jpg", using "script.jpg.php" we were able to get a successful upload on to the web application.
Images	
Affected Hosts	192.168.14.35/Memory-Planner.php
Remediation	File Type Verification: Instead of relying solely on file extensions for input validation, implement file type verification based on the content of the uploaded file. Use server-side techniques to verify the file's MIME type or magic bytes to ensure that it matches the expected file type. Whitelist Validation: Implement a whitelist validation approach to restrict the allowed file types for upload. Only allow specific file types that are necessary for the application's functionality, such as image files like JPEG or PNG. Reject any files that do not match the allowed types. Dual Extension Detection: Implement detection mechanisms to identify and prevent files with dual extensions, such as "script.jpg.php." Reject any file names that contain multiple extensions or unusual patterns that may indicate malicious intent. Rename Uploaded Files: Automatically rename uploaded files to remove any potentially malicious file extensions or patterns. Use a unique and secure naming convention to ensure that uploaded files are safely stored and do not pose a security risk when accessed by the application. Secure File Storage:

	Store uploaded files in a secure directory outside of the web root to prevent direct access by users. Ensure that proper access controls are in place to restrict access to uploaded files and prevent unauthorized retrieval or execution. Regular Security Audits: Conduct regular security audits of the application code to identify and address any vulnerabilities related to file uploads and input validation. Perform code reviews and security testing to ensure that input validation mechanisms are effective and properly implemented. Security Awareness Training: Provide security awareness training to developers and administrators to educate them about the risks of LFI vulnerabilities and the importance of implementing secure coding practices, including proper input validation for file uploads.

Vulnerability 21	Findings
Title	**XSS Cross-Site Scripting (XSS)**
Type (Web app / Linux OS / WIndows OS)	Web Application
Risk Rating	**Low**
Description	ZDCA was able to trigger an alert pop up on the screen indicating that input was received as html code on the web application. example: ”<script>alert(‘XSS tets alert!’)</script>”
Images	
Affected Hosts	192.168.14.35

Remediation	Input Sanitization: Ensure all user inputs are sanitized before they are used within web pages. This includes escaping special characters such as `<`, `>`, `&`, `'`, and `"` to their respective HTML entities. Sanitization should occur both on the client-side (for immediate feedback) and on the server-side (as the primary defense mechanism). Output Encoding: When displaying user input or data retrieved from databases, apply proper output encoding to ensure that any HTML or JavaScript code is treated as plain text and not executed by the browser. Use context-appropriate encoding techniques for HTML element content, HTML attribute values, JavaScript, CSS, and URLs. Content Security Policy (CSP): Implement a Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities by declaring what dynamic resources are allowed to load. For instance, a CSP can be configured to disallow the execution of inline JavaScript and restrict JavaScript to files loaded from specific, trusted domains. Use Frameworks That Automatically Escape XSS: Many modern web development frameworks and templating engines automatically escape XSS by design. Whenever possible, use frameworks (such as React, Angular, Vue.js, or others that auto-escape XSS) to handle dynamic content. Validate Input Based on a Whitelist: Where applicable, validate all user input against a whitelist of allowed characters. This is particularly important for fields that require a specific format (e.g., date, email address, phone number). Secure Cookies: Mark sensitive cookies as `HttpOnly` and `Secure` to prevent XSS attacks from accessing these cookies. The `HttpOnly` attribute prevents JavaScript access to the cookie, while the `Secure` attribute ensures the cookie is sent over HTTPS connections only. Regularly Update Libraries and Frameworks: Keep all third-party libraries and frameworks updated to their latest stable versions. Many XSS vulnerabilities are discovered in older versions of software, and updates often include patches for these security holes. Educate and Train Developers: Provide security awareness training for developers, focusing on the risks associated with XSS attacks and best practices for preventing them. This includes secure coding practices, input sanitization, and the importance of keeping software updated. Implement a WAF (Web Application Firewall): As an additional security measure, deploy a Web Application Firewall (WAF) configured to detect and block XSS attacks. While not a substitute for secure coding practices, a WAF can provide an extra layer of defense against attackers exploiting known and unknown vulnerabilities.

Vulnerability 22	Findings

Title	Open Source Exposed Data(Multiple occurrences)
Type (Web app / Linux OS / WIndows OS)	We Application
Risk Rating	Low
Description	ZDCA was able to run a whois on the domain totalrekall.xyz and was exposed to sensitive data.
Images	Domain Status: autoRenewPeriod https://icann.org/epp#autoRenewPeriod Registrant Organization: Registrant State/Province: Georgia Registrant Country: US Registrant Email: Please query the RDDS service of the Registrar of Record identified in this Admin Email: Please query the RDDS service of the Registrar of Record identified in this outpu Tech Email: Please query the RDDS service of the Registrar of Record identified in this output Name Server: NS51.DOMAINCONTROL.COM Name Server: NS52.DOMAINCONTROL.COM DNSSEC: unsigned Billing Email: Please query the RDDS service of the Registrar of Record identified in this out Registrar Abuse Contact Email: abuse@godaddy.com Registrar Abuse Contact Phone: +1.4805058800 URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/ >>> Last update of WHOIS database: 2024-03-04T18:55:02.0Z <<< Queried **whois.godaddy.com** with "**totalrekall.xyz**"... Domain Name: totalrekall.xyz Registry Domain ID: D273189417-CNIC Registrar WHOIS Server: whois.godaddy.com Registrar URL: https://www.godaddy.com Updated Date: 2024-02-03T15:15:56Z Creation Date: 2022-02-02T19:16:16Z Registrar Registration Expiration Date: 2025-02-02T23:59:59Z Registrar: GoDaddy.com, LLC Registrar IANA ID: 146 Registrar Abuse Contact Email: abuse@godaddy.com Registrar Abuse Contact Phone: +1.4806242505 Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited Domain Status: clientUpdateProhibited https://icann.org/epp#clientUpdateProhibited Domain Status: clientRenewProhibited https://icann.org/epp#clientRenewProhibited Domain Status: clientDeleteProhibited https://icann.org/epp#clientDeleteProhibited Registry Registrant ID: CR534509109 Registrant Name: sshUser alice Registrant Organization: Registrant Street: h8s692hskasd Flag1 Registrant City: Atlanta Registrant State/Province: Georgia Registrant Postal Code: 30309 Registrant Country: US Registrant Phone: +1.7702229999 Registrant Phone Ext: Registrant Fax: Registrant Fax Ext: Registrant Email: jlow@2u.com Registry Admin ID: CR534509111 Admin Name: sshUser alice Admin Organization: Admin Street: h8s692hskasd Flag1 Admin City: Atlanta Admin State/Province: Georgia Admin Postal Code: 30309 Admin Country: US Admin Phone: +1.7702229999 Admin Phone Ext: Admin Fax: Admin Fax Ext: Admin Email: jlow@2u.com Registry Tech ID: CR534509110 Tech Name: sshUser alice Tech Organization: Tech Street: h8s692hskasd Flag1
Affected Hosts	totalrekall.xyz
Remediation	Domain Privacy Protection: Most domain registrars offer a privacy protection service (sometimes called WHOIS protection) that hides your personal information from the public WHOIS database. This service typically replaces your contact information with the information of a proxy service, thus protecting your personal details from being exposed.

	Review and Update Registrar Data: Ensure that the contact information you have on file with your domain registrar is accurate but minimal. For business-related domains, use official business contact information rather than personal information. Legal and Regulatory Compliance: Ensure that any data exposure complies with applicable laws and regulations, such as GDPR in Europe, which might require specific handling of personal data and potentially offer additional protection mechanisms for individuals' data. Use a Corporate Domain Registrar: If the domain hosts significant business operations, consider using a corporate domain registrar that specializes in providing services to businesses, including enhanced security options and dedicated support. Secure Domain Configuration: Beyond WHOIS privacy, ensure that your domain and DNS settings are secure. This includes using DNSSEC (Domain Name System Security Extensions) to protect against DNS spoofing, enabling registry locks to prevent unauthorized domain transfers, and using HTTPS for all your sites to protect user data. Regular Monitoring and Auditing: Regularly monitor your domain's WHOIS information and perform security audits on your web presence to identify any potential information leaks or security vulnerabilities. Tools and services are available that can help automate this process. Educate and Train Staff: Ensure that anyone involved in managing your web presence understands the importance of data privacy and security. This includes choosing secure passwords, recognizing phishing attempts, and knowing the correct procedures for updating domain and website information.