| Cybersecurity |
| --- |
| Penetration Test Report |

Rekall Corporation

Penetration Test Report

**Andrew Barry Cyber, 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](#_30j0zll)

[Contact Information 4](#_1fob9te)

[Document History 4](#_3znysh7)

[Introduction 5](#_2et92p0)

[Assessment Objective 5](#_3dy6vkm)

[Penetration Testing Methodology 6](#_2s8eyo1)

[Reconnaissance 6](#_17dp8vu)

[Identification of Vulnerabilities and Services 6](#_3rdcrjn)

[Vulnerability Exploitation 6](#_26in1rg)

[Reporting 6](#_lnxbz9)

[Scope 7](#_35nkun2)

[Executive Summary of Findings 8](#_44sinio)

[Grading Methodology 8](#_z337ya)

[Summary of Strengths 9](#_3j2qqm3)

[Summary of Weaknesses 9](#_1y810tw)

[Executive Summary Narrative](#_4i7ojhp) 10

[Summary Vulnerability Overview 1](#_2xcytpi)3

Vulnerability Findings [1](#_1ci93xb)4

# 

## Contact Information

| **Company Name** | Andrew Barry Cyber, LLC |
| --- | --- |
| **Contact Name** | Andrew Barry |
| **Contact Title** | Penetration Tester |

## 

## Document History

| **Version** | **Date** | **Author(s)** | **Comments** |
| --- | --- | --- | --- |
| 001 | 15/06/2024 | Andrew Barry |  |

# 

## 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:

Chart

Description automatically generated with medium confidence

### 

### 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.

* Website was easy to use.
* Infrastructure was divided equally into web app, Linux and Windows.
* Rekall invited us for pen testing and is taking a proactive approach.

### 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.

* Evidence of weak passwords with vulnerability to brute force and cracking.
* Open ports.
* Exposure of sensitive data on public platforms.
* Weak infrastructure.
* Use of outdated software/services with known vulnerabilities.
* Vulnerable to privilege escalation.
* Vulnerable to various injection attacks.
* Vulnerable to credential dumping.

## Executive Summary

We conducted our work over three separate days targeting your web application on day one, Linux systems on day two and finally turning our attention to your Windows systems on day three.

On day one we targeted your web application. We navigated through various tabs and pages. We deployed multiple scripts during our efforts. We were able to successfully exploit a variety of vulnerabilities. We also saw evidence of weak passwords and exposure of sensitive data.

On day two we focussed our efforts on your Linux systems. We ran multiple scans and identified a variety of open ports and services running on these. This allowed us to identify vulnerabilities, particularly in services where you are still running unpatched older versions. We gained access, escalated our privileges and exploited your system. We saw evidence of open-source exposed data and again saw weak password practices.

On our final day we turned our attention to your Windows systems. We found an employee’s user name and password hash through an open-source search. We successfully cracked this hash and were able to access confidential files. We ran multiple scans to identify open ports and the services running on these. This allowed us to establish a foothold on your system by exploiting known vulnerabilities that you have failed to patch. We stole credentials through two LSA dumps and a DCSync attack. We successfully cracked the password hashes of two different admin users and were able to laterally move and gain access to your Domain Controller. Once again we saw evidence of poor password practices.

Below we have outlined in detail the hosts, ports and services we targeted. We have highlighted the actions we took to exploit your systems and the steps you should take to remediate deficiencies and strengthen your network. We have ranked the various vulnerabilities we identified in order to assist you in prioritising your remediation efforts and allocation of resources.

We remain available should you wish to discuss this report in more detail or require further advice and assistance.

## 

## 

## Summary Vulnerability Overview

| **Vulnerability** | **Severity** |
| --- | --- |
| SQL injection - **Web App** | **Critical** |
| Command Injection - **Web App** | **Critical** |
| PHP injection - **Web App** | **Critical** |
| Cross-Site Scripting (Reflected & Stored) - **Web App** | **High** |
| Exposure of Sensitive Data - **Web App** | **High** |
| Local File Inclusion - **Web App** | **High** |
| Directory Traversal - **Web App** | **High** |
| Brute Force Attack - **Web App** | **Medium** |
| Session Management - **Web App** | **Medium** |
|  |  |
| Apache Struts - CVE-2017-5638 - **Linux OS** | **Critical** |
| Apache Tomcat Remote Code Execution Vulnerability - CVE-2017-12617 - **Linux OS** | **Critical** |
| Shellshock - **Linux OS** | **Critical** |
| Privilege Escalation - CVE-2019-14287 - **Linux OS** | **Critical** |
| Domain Discovery - **Linux OS** | **High** |
| Open Source Exposed Data - **Linux & Windows OS** | **High** |
| Open Ports - **Linux & Windows OS** | **High** |
| Drupal - CVE-2019-6340 - **Linux OS** | **High** |
|  |  |
| DCSync Attack - **Windows OS** | **Critical** |
| Buffer overflow - POP3/SLMail - **Windows OS** | **High** |
| LSA Credential Dumping - **Windows OS** | **High** |
| SMB/PsExec - **Windows OS** | **High** |
| FTP Anonymous - **Windows OS** | **Medium** |
|  |  |
| Weak Passwords - **General** | **High** |

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

| **Scan Type** | **Total** |
| --- | --- |
| Hosts | 192.168.14.35  192.168.13.1  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  172.22.117.100 |
| Ports | 21, 22, 53, 79, 80, 88, 106, 110, 135, 139, 389, 443, 445, 464, 593, 636, 5901, 6001, 8009, 8080, 10000, 10001 |

| **Exploitation Risk** | **Total** |
| --- | --- |
| **Critical** | 8 |
| **High** | 12 |
| **Medium** | 3 |
| **Low** | 0 |

## Vulnerability Findings

| **Vulnerability 1** | **Findings** |
| --- | --- |
| **Title** | Cross-Site Scripting (Reflected & Stored) |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **High** |
| **Description** | The Rekall welcome.php (Web app flag 1) and Memory-Planner.php (Web app flag 2) pages are vulnerable to reflected cross-site scripting.The “Put Your Name Here” and “Choose Your Character” input fields have insufficient input validation and output encoding. The application is not validating input properly and ensuring that only safe characters are accepted. The application then also fails to encode output properly. This failure allows the input to be interpreted as executable code. We were able to run cross-site scripts and make pop-ups appear. There was a degree of input validation observed on the Memory-Planner.php page. However, we bypassed this by obfuscating the word script <SCRIPscriptT>alert("hi")</SCRIPscripTt>.  The Rekall comments.php (Web app flag 3) page is vulnerable to stored cross-site scripting through the “Comments” input field. The same input validation and output encoding issues were apparent. As this is a stored cross-site script it will affect all users who visit the compromised page. |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Input validation - make sure all input conforms to the expected type, format and length. * Input sanitisation - ensure all input is properly sanitised by removing or escaping special characters. * Output encoding - prevent malicious scripts from being executed by ensuring that data being outputted to the browser is always treated as text rather than executable code. * Content Security Policy - use a CSP to stop the execution of scripts from unknown domains. By restricting the sources from which content will be accepted there is less chance of malicious scripts being executed. * Education & training - run training and awareness initiatives so that your developers and users are aware of the risks of cross-site scripting. * Security - improve the overall security of your web app by running regular automated scans and twice-yearly pen tests. |

| **Vulnerability 2** | **Findings** |
| --- | --- |
| **Title** | Exposure of sensitive data |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **High** |
| **Description** | By using a curl command (curl -v <http://192.168.14.35/About-Rekall.php>) we were able to view sensitive data (Web app flag 4) in the HTTP response headers of your web application.  The Rekall login.php page (Web app flag 8) has sensitive data exposed on it. Both the username and password for the Admin Login could be viewed by simply highlighting the page or by inspecting the HTML source code.  The robots.txt file is publicly accessible (Web app flag 9) and there are no controls to stop unauthorised access. We were able to view sensitive data about your application by accessing the robots.txt file directly through our browser. |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Remove sensitive information - ensure that no sensitive data (ex session tokens, API keys etc) is included in HTTP response headers. * Encryption - ensure that data in transit is encrypted by use of HTTPS. * Logging & monitoring - detect any unauthorised access to data by implementing logging and regular monitoring. * Restrict access - implement access controls to sensitive data by ensuring strong authorisation and authentication mechanisms are in place. * Remove sensitive info from HTML - conduct and audit of the HTML source code and remove all sensitive information from it. * Secure session management - rather than storing or transmitting sensitive data via the client ensure that it is handled securely by session management on the server side. * Two-Factor Authentication - implement 2FA for all admin logins. * Security - conduct regular security audits and twice yearly pen testing to search for sensitive data exposure. * Removal of information - review the robots.txt file and remove any sensitive information that might be helpful to a potential attacker (ex info about sensitive endpoints, directories or files). Conduct periodic reviews to ensure the document remains appropriately sanitised. * Education & training - run training and awareness initiatives so that your developers, admins and website owners are aware of the risks posed by failing to secure and manage sensitive documents such as robots.txt and others. |

| **Vulnerability 3** | **Findings** |
| --- | --- |
| **Title** | Local file inclusion |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **High** |
| **Description** | The Rekall Memory-Panner.php page is vulnerable to local file inclusion. Through the second (Web app flag 5) and third (Web app flag 6) fields where users can “Upload Your File” we were able to execute local file inclusion. This allowed us to upload a PHP file which would allow us to execute arbitrary PHP code and to access sensitive documents such as configuration files.  There was a degree of input validation observed in the third field as checks for the presence of “.jpg” in the title of the uploaded file were carried out. However, we were able to bypass this and upload a PHP file by naming our malicious file “script.jpg.php. |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Input validation - only permit a controlled set of acceptable file names or paths. * Input sanitisation - ensure all input is properly sanitised by removing or escaping special characters. * Implement least privilege - ensure that the web server has the least privilege required to operate and does not have write permission on sensitive directories. * Logging - implement logging and regularly monitor these logs for any suspicious activity. |

| **Vulnerability 4** | **Findings** |
| --- | --- |
| **Title** | SQL injection |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **Critical** |
| **Description** | The login.php page is vulnerable to SQL injection. User input in the first field of this page is not properly sanitised. By adding “ok' or 1=1–” to our input we were able to successfully exploit this vulnerability (Web app flag 7). |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Prepared staments & parametrised queries - ensure that an attacker cannot change the intent of a query by using prepared statements and queries. * Use stored procedures. * User input validation & sanitisation - ensure all input is validated and conforms to expected length, type and format. Reject any input that contain malicious content. * Escape special characters in user input. * WAF - use a firewall to detect and block attempts at SQL injection. * Patch - regularly update and patch your database management system * Conduct regular code reviews and twice yearly pen testing. |

| **Vulnerability 5** | **Findings** |
| --- | --- |
| **Title** | Command injection |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **Critical** |
| **Description** | The Rekall networking.php page is vulnerable to command injection. Through the first (Web app flag 10) and second (Web app flag 11) fields we were able to inject commands which resulted in unauthorised access of sensitive files. Through the first input field, we were able to execute an arbitrary command by entering “www.welcometorecall.com && cat vendors.txt”.  A degree of input validation was observed in the second field as "&" and ";" were stripped out of our initial commands. However, by using a pipe we were able to successfully bypass the input validation and run “www.welcometorecall.com | cat vendors.txt”. |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Input validation - make sure all input conforms to the expected type, format and length. Do not allow the use of characters that act as command delimiters. * Input sanitisation - ensure all input is properly sanitised by removing or escaping special characters that might enable execution of malicious commands. * Use prepared or parametrised statements when executing commands thereby preventing injection attacks. * Implement least privilege - ensure that the web server and any processes it initiates have the least privilege required to operate and access to system resources, sensitive operation/data is minimised to only those absolutely essential. * Whitelist - validate all user input against a list of allowed commands. |

| **Vulnerability 6** | **Findings** |
| --- | --- |
| **Title** | Brute force attack |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **Medium** |
| **Description** | By utilising command injection we view able to view /etc/passwd and gather usernames and then brute force passwords (Web app flag 12). |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Srong passwords - implement a strong password policies that ensures paswords are a mix of lowercase and uppercase, a minimum length and also contain special characters. * Hashing & salting - ensure passwords are hashed using a modern and strong cryptographic algorithm. Add a further layer of complexity by ensuring hashes are salted. * Training - run awareness training so that users and admins are educated regarding the importance of strong passwords and overall password hygiene/management. Discourage the use of simple passwords. * Two-factor authentication - enable 2FA for admin logins. * Lockout - implement a lockout mechanism that temporarily locks an account and notifies an admin after a certain number of failed attempts. * Rate limiting - implement rate limiting which limits the number of login attempts from an IP address or a user within a certain period of time. * Logging & monitoring - detect suspicious activity such as multiple failed login attempts in an effort to identify and stop brute force attacks. |

| **Vulnerability 7** | **Findings** |
| --- | --- |
| **Title** | PHP injection |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **Critical** |
| **Description** | A hidden webpage was identified when we accessed the robots.txt document. This webpage was vulnerable to PHP injection. By changing the URL to “http://192.168.13.35/souvenirs.php?message=""; system('cat /etc/passwd')” we were successfully able to view sensitive files (Web app flag 13). |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Input validation & sanitisation - ensure all user input conforms to expected lengths and formats and does not contain malicious PHP code. * Use prepared or parametrised statements when executing commands. * File paths - use predefined file paths and valid user requests against a whitelist of allowed files. * Security - conduct period security assessments, code reviews, vulnerability scans and twice-yearly pen tests. |

| **Vulnerability 8** | **Findings** |
| --- | --- |
| **Title** | Session management |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **Medium** |
| **Description** | The web app is susceptible to a session management vulnerability. By using Burp Intruder we were able to analyse intercepted session tokens and accurately predict the pattern. We identified a valid session token and then crafted a request with the valid token (Web app flag 14). http://192.168.13.35/admin\_legal\_data.php?admin=87 |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Session tokens - strengthen the token generation, tokens need to be unique, long, random and hard to predict/brute force. * Token transmission - tokens need to be transmitted using HTTPS to prevent interception. * Session management - ensure tokens are validated at each request and reduce the chances of client-side manipulation by implementing server-side session management, expire tokens after a certain period of inactivity, and issue new tokens when attempting sensitive operations. |

| **Vulnerability 9** | **Findings** |
| --- | --- |
| **Title** | Directory traversal |
| **Type (Web app / Linux OS / WIndows OS)** | Web app |
| **Risk Rating** | **High** |
| **Description** | The disclaimer.php page is vulnerable to directory traversal. By amending the URL to <http://192.168.13.35/disclaimer.php?page=old_disclaimers/disclaimer_1.txt> we were able to access a different resource (Web app flag 15). |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35 |
| **Remediation** | * Input validation & sanitisation - validate and sanitise all user input, have a predefined whitelist of filenames and check requests against this, reject user input that contains special characters or directory traversal sequences. * Server configuration - ensure your web server is configured so access to sensitive directories is restricted. * Least privilege -ensure that the web server is running with the least privilege required for operations and do not run the server as admin or root if possible. * Log/monitor - introducing logging and monitoring to allow detection and response to traversal attacks. * Security - educate admins and web developers concerning traversal attacks and implement twice-yearly pen testing. |

| **Vulnerability 10** | **Findings** |
| --- | --- |
| **Title** | Open-Source Exposed Data |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **High** |
| **Description** | By using OSINT tools we were able to view sensitive data regarding totalrekall.xyz (Linux flag 1).  By examining the SSL certificate for totalrekall.xyz we were able to gather sensitive data about your systems (Linux flag 3). |
| **Images** |  |
| **Affected Hosts** | totalrekall.xyz |
| **Remediation** | * Conduct a review of all publically exposed data and remove anything that might be deemed useful to an attacker. * Review certificate - conduct a review of the certificate and remove any sensitive data such as IP addresses, server hostnames and internal domain names. * Generic naming - use generic naming within the certificate to avoid revealing the internal infrastructure layout to potential attackers. * Secure key - ensure your private key is stored securely and encrypted by a strong modern algorithm. * Monitor cert issuance - implement monitoring and reviewing of all certs issued for your domain to detect any unauthorised certs. * Rotation of certs - routinely rotate certs and rotate immediately if you discover sensitive info has been included on a cert. * Certificate management - introduce automated cert management, this will remove human error, increase standardisation, improve configuration and streamline the issuance, renewal, removal and monitoring of all certs. |

| **Vulnerability 11** | **Findings** |
| --- | --- |
| **Title** | Domain Discovery |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **High** |
| **Description** | We performed a named server lookup and were able to view domain related information (Linux flag 2). This allowed us to map out a digital footprint of your organisation and helped in deciding what entry points we would target |
| **Images** |  |
| **Affected Hosts** | totalrekall.xyz |
| **Remediation** | * Audit - regularly audit all your domains and subdomains to esure they are properly secured. * Monitor - use a domain monitoring service so you are alerted when a new domain/subdoamin is registered. * Domain Name System Security Extension - use DNSSEC to safeguard your DNS integrity and prevent DNS spoofing. * WHOIS - hide sensitive information in WHOIS records by using a privacy protection service, always use organisational instead of personal information. * Security - conduct regular audits, scans and twice yearly pen testing to detect and address any vulnerabilities in your domains/subdomains. |

| **Vulnerability 12** | **Findings** |
| --- | --- |
| **Title** | Open Ports |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **High** |
| **Description** | We were able to identify multiple open ports through Nmap and Zenmap scans (Linux flags 4 & 5). This also allowed us to identify the services running on these ports and target vulnerabilities in these. |
| **Images** |  |
| **Affected Hosts** | 192.168.14.35, 192.168.13.1, 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, 172.22.117.100 |
| **Remediation** | * Close any ports that are not absolutely essential for operations. * Conduct an audit of all applications and services currently running on your system and uninstall any that are deemed unnecessary or outdated. * Ensure services that are running are up-to-date versions and any patches issued are immediately implemented. * Conduct frequent vulnerability scans and remediate any weaknesses or misconfigurations. |

| **Vulnerability 13** | **Findings** |
| --- | --- |
| **Title** | Apache Struts - CVE-2017-5638 |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **Critical** |
| **Description** | A Nessus scan revealed (Linux flag 6) that a server was running a vulnerable version of Apache Struts. As a result attackers might exploit this by sending specially crafted HTTP requests to a web application. The server may process the request and execute the malicious code. This could potentially lead to the compromise of that server.  We successfully used the multi/http/struts2\_content\_type\_ognl module to exploit this vulnerability start a meterpreter shell session and access files (Linux flag 10) on your system. |
| **Images** |  |
| **Affected Hosts** | 192.168.13.12 |
| **Remediation** | * Patch - immediately implement the patch that the manufacturer has issued for this vulnerability, ensure any future patches are also immediately updated. * WAF - ensure that you have an effective Web Application Firewall in place to detect and stop any malicious HTTP requests attempting to exploit this vulnerability. * Monitor & log - monitor and log traffic and server activity for any attempts at exploitation. * Network segmentation - segmenting your network can limit the damage caused if this vulnerability is successfully exploited by an attacker. * Security - conduct regular audits, vulnerability scans and twice-yearly pen testing. |

| **Vulnerability 14** | **Findings** |
| --- | --- |
| **Title** | Apache Tomcat Remote Code Execution Vulnerability - CVE-2017-12617 |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **Critical** |
| **Description** | This vulnerability means that an attacker can potentially remotely execute arbitrary code on this server. By using the Metasploit multi/http/tomcat\_jsp\_upload\_bypass module we exploited this vulnerability and successfully established a shell session with this host. We were then able to access files on the system (Linux flag 7). |
| **Images** |  |
| **Affected Hosts** | 192.168.13.10 |
| **Remediation** | * Patch - immediately implement the patch that the manufacturer has issued for this vulnerability, ensure any future patches are also immediately updated. * Disable CGI servlet - if the CGI servlet is not essential for operations disable it by ammending the web.xml config file. * WAF - ensure that you have an effective Web Application Firewall in place to detect and stop any malicious HTTP requests attempting to exploit this vulnerability. * Monitor & log - monitor and log traffic and server activity for any attempts at exploitation. * Network segmentation - segmenting your network can limit the damage caused if this vulnerability is successfully exploited by an attacker. * Security - conduct regular audits, vulnerability scans and twice-yearly pen testing. |

| **Vulnerability 15** | **Findings** |
| --- | --- |
| **Title** | Shellshock |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **Critical** |
| **Description** | We used the Metasploit module ***exploit/multi/http/apache\_mod\_cgi\_bash\_env\_exec*** to exploit this host's Shellshock vulnerability. The module sent a specially crafted HTTP request to the server which then allowed us to execute arbitrary commands on the target. We successfully gained access to the server and were able to examine sensitive documents such as the /etc/sudoers (Linux flag 8) and /etc/passwd (Linux flag 9) files. |
| **Images** |  |
| **Affected Hosts** | 192.168.13.11 |
| **Remediation** | * Patch - immediately implement the patch to Bash that the manufacturer has issued for this vulnerability, ensure any future patches are also immediately updated. * Disable CGI scripts - until the Bash vulnerability has been patched. * WAF - ensure that you have an effective Web Application Firewall in place to detect and stop any malicious HTTP requests attempting to exploit this vulnerability. * Monitor & log - monitor and log traffic and server activity for any attempts at exploitation. * Network segmentation - segmenting your network can limit the damage caused if this vulnerability is successfully exploited by an attacker. * Security - conduct regular audits, vulnerability scans and twice-yearly pen testing. * Training - educate admins about the Shellshock vulnerability and stress the importance of promptly applying security patches. |

| **Vulnerability 16** | **Findings** |
| --- | --- |
| **Title** | Drupal - CVE-2019-6340 |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **High** |
| **Description** | This vulnerability arises due to insufficient validation of user-supplied input by the RESTWS module resulting in remote code execution on the server which is targeted. We utilsed the ***unix/webapp/drupal\_restws\_unserialize*** Metasploit module to exploit this vulnerability. By sending a specially crafted malicious HTTP request we were able to execute code remotely and successfully establish a shell session (Linux flag 11). |
| **Images** |  |
| **Affected Hosts** | 192.168.13.13 |
| **Remediation** | * Patch - immediately implement the patch to Drupal that the manufacturer has issued for this vulnerability, ensure any future patches are also immediately updated. * Disable RESTful web services - if this module is not needed disable it. * WAF - ensure that you have an effective Web Application Firewall in place to detect and stop any malicious HTTP requests attempting to exploit this vulnerability. * Monitor & log - monitor and log traffic and server activity for any attempts at exploitation. * Network segmentation - segmenting your network can limit the damage caused if this vulnerability is successfully exploited by an attacker. * Security - conduct regular audits, vulnerability scans and twice-yearly pen testing. * Training - educate admins about the Shellshock vulnerability and stress the importance of promptly applying security patches. |

| **Vulnerability 17** | **Findings** |
| --- | --- |
| **Title** | Sensitive data exposure, weak passwords & privilege escalation - CVE-2019-14287 |
| **Type (Web app / Linux OS / WIndows OS)** | Linux OS |
| **Risk Rating** | **Critical** |
| **Description** | Sensitive data exposure allowed us to establish that there was an SSH user named “Alice”. Her password was also Alice. The exposure of her username and her weak password allowed us to establish an SSH session with 192.168.13.14. We then ran the command ***sudo -u#-1 cat /root/flag12.txt***. This exploited a vulnerability in the way sudo handles the user ID -1 and granted us root privilege and the ability to access the document flag12.txt (Linux flag 12). |
| **Images** |  |
| **Affected Hosts** | 192.168.13.14 |
| **Remediation** | * Conduct an audit - examine all information publicly available and remove any sensitive data. * Educate - all staff on the importance of strong, complex passwords containing uppercase and lowercase letters as well as numbers and special characters. * Two-factor authentication - implement 2FA for SSH sessions. * Upgrade sudo - any version from 1.8.28 or later will ensure that this vulnerability is mitigated. * Configuration file - regularly review and amend as required the sudoers config file. * Least privilege - ensure that all users have the least privileges required to perform their function. Ensure staff who move roles do not retain unnecessary privileges. |

| **Vulnerability 18** | **Findings** |
| --- | --- |
| **Title** | Open-source exposed data & weak passwords |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **High** |
| **Description** | A Google search revealed a publicly accessible totalrekall GitHub page. Within the site repository, we found an xampp.users page which contained the user name and hashed password of a user. This hash was cracked using John the Ripper and the users password was exposed (Windows flag 1). With these credentials, we were able to login and view restricted content hosted at 172.22.117.20 (Windows flag 2). |
| **Images** |  |
| **Affected Hosts** | 172.22.117.20 |
| **Remediation** | * Conduct an audit - examine all information publicly available and remove any sensitive data. * Educate - all staff on the importance of strong, complex passwords containing uppercase and lowercase letters as well as numbers and special characters. |

| **Vulnerability 19** | **Findings** |
| --- | --- |
| **Title** | FTP anonymous |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **Medium** |
| **Description** | An Nmap scan revealed that FTP was running and had anonymous login enabled. This allowed us to login without authentication and view files (Windows flag 3). |
| **Images** |  |
| **Affected Hosts** | 172.22.117.20 |
| **Remediation** | * Disable anonymous FTP access. * Access - restrict access to the FTP server to a whitelist of IP addresses. * Use secure FTP protocols - implement either SFTP or FTPS both of which are more secure than your current protocol. * Patch - regularly apply any updates/patches issued to the FTP server in order to mitigate known vulnerabilities. * Passwords - require strong, complex passwords containing uppercase and lowercase letters as well as numbers and special characters. * Two-factor authentication - implement 2FA for FTP sessions. * Log/monitor - log and review FTP server traffic to identify suspicious activity. |

| **Vulnerability 20** | **Findings** |
| --- | --- |
| **Title** | Buffer overflow - POP3/SLMail |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **High** |
| **Description** | Our Nmap scan had revealed that SLMail and POP3 were running. Using the windows/pop3/seattlelab\_pass exploit we were able to take advantage of a vulnerability in the POP3 service of SLMail and gain access to your Windows10 machine. This exploit causes buffer overflow which allowed us to execute arbitrary code and establish a meterpreter shell session. We were then able to search through the system and access files (Windows flags 4 & 7) and examine your scheduled tasks (Windows flag 5). |
| **Images** |  |
| **Affected Hosts** | 172.22.117.20 |
| **Remediation** | * Replace SLMail - consider replacing SLMail with a more up-to-date option such as Postfix or Microsoft Exchange. * Update SLMail - if you must keep SLMail ensure that any available updates/patches are immediately implemented. * Firewall - configure your firewall rules so that access to POP3 is restricted to whitelisted IPs. * IPS/IDS - detect and block attempts to exploit this vulnerability by using IPS/IDS. * Least privilege - ensure that the mail server is run with the least privileges required. * Log/monitor - log and review server traffic to identify suspicious activity. |

| **Vulnerability 21** | **Findings** |
| --- | --- |
| **Title** | LSA Credential Dumping |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **High** |
| **Description** | Having gained access to your Windows10 machine we then used kiwi to steal credentials through two separate LSA dumps. First, we stole credentials from the Security Account Manager (Windows flag 6). Next, we stole the cached credentials of an administrator with the username ‘ADMBob’. All the stolen credentials were then cracked using John the Ripper. |
| **Images** |  |
| **Affected Hosts** | 172.22.117.20 |
| **Remediation** | * Credential guard - protect credentials by enabling Credential Guard and storing them in an isolated virtual environment. * LSA protection - enable LSA protection in the registry to make it harder for threat actors to access LSA. * Reduce cached creds - configure the cached credentials settings so fewer are stored (consider setting this to zero if operationally feasible). * Log/monitor - use your SIEM to detect any suspicious activities that may indicate credential dumping. * Passwords - implement a strong password policy so that if credentials are stolen they are harder to crack. * Two-factor authentication - implement 2FA so that it is harder for attackers to successfully use any stolen credentials. |

| **Vulnerability 22** | **Findings** |
| --- | --- |
| **Title** | SMB/PsExec |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **High** |
| **Description** | By using the stolen/cracked credentials of administrator ‘ADMBob’ and the windows/smb/psexec module in metasploit we were able to move laterally from the Windows10 machine to your WinDC domain controller. This exploit allowed us to connect to the target over SMB, obtain a SYSTEM shell, execute commands and access files with the privileges of ‘ADMBob’ (Windows flags 8 & 9). |
| **Images** |  |
| **Affected Hosts** | 172.22.117.10 |
| **Remediation** | * Patch - ensure that SMB has all relevant updates and patches implemented. * Firewall - use rules to restrict traffic to trusted networks and hosts, and block all inbound SMB traffic from untrusted networks. * Log/monitor - use your SIEM to detect any suspicious activities that may indicate SMB/PsExec exploitation. * Two-Factor Authentication - implement 2FA for admin account access. * Passwords - an attacker must have valid credentials to exploit this vulnerability, so implement a stong password policy and educate all staff regarding this. |

| **Vulnerability 23** | **Findings** |
| --- | --- |
| **Title** | DCSync attack |
| **Type (Web app / Linux OS / WIndows OS)** | Windows OS |
| **Risk Rating** | **Critical** |
| **Description** | Domain controllers use the Directory Replication Service (DRS) to replicate and share data. We used kiwi to generate a DRS replication request for a user named ‘administrator’. The Domain Controller responded to this request and provided a variety of sensitive information including the NTLM hash (Windows flag 10) of the user’s password. |
| **Images** |  |
| **Affected Hosts** | 172.22.117.10 |
| **Remediation** | * Limit number of high privilege accounts - to execute a DCSync attack an attacker typically needs high pirvilefge access, by limiting the number of accounts you can reduce your attack surface. * Group policy - unless specifically required deny users access to ‘replicating directory changes’. * Two-factor authentication - implement 2FA for all accounts with high privileges. * Passwords - implement a strong password policy so that if credentials are stolen they are harder to crack. Educate staff regarding this. * Log/monitor - use your SIEM to detect any suspicious activities that may indicate a DCSync attack. * Firewalls - restrict access to DCs. * Endpoint protection - implement EDR tools that will detect and block tools that are used to conduct a DCSync attack. |