Ayush Shrivastava

Jangow Vulnerable Machine Exploitation Report

20/11/24

Contents

[Lab Setup 2](#_Toc183031674)

[Network Configuration 2](#_Toc183031675)

[Reconnaissance 2](#_Toc183031676)

[Scan Results 3](#_Toc183031677)

[Gaining Access 4](#_Toc183031678)

[Enumeration of Web Directory: 4](#_Toc183031679)

[Access Attempt 1: Credentials from config.php 5](#_Toc183031680)

[Access Attempt 2: Credentials from the Backup File 7](#_Toc183031681)

[Privilege Escalation 9](#_Toc183031682)

[Attempt 1: PHP Reverse Shell via Local File Inclusion 9](#_Toc183031683)

[Attempt 2 Reverse Shell via Bash Shell Script 10](#_Toc183031684)

[Post-Connection Steps: Making the Reverse Shell Interactive 11](#_Toc183031685)

[Upgrading to an interactive Shell 11](#_Toc183031686)

[Setting the terminal type 11](#_Toc183031687)

[Switching User with su Command 11](#_Toc183031688)

[Discovery of the User Flag 12](#_Toc183031689)

[Privilege Escalation to Root Access 14](#_Toc183031690)

[Downloading and uploading LinPEAS 14](#_Toc183031691)

[Changing File Permissions 15](#_Toc183031692)

[Executing LinPEAS 16](#_Toc183031693)

[Identifying the Vulnerability 16](#_Toc183031694)

[Compiling and Executing the Exploit 19](#_Toc183031695)

[Claiming the Root Flag 20](#_Toc183031696)

[Conclusion 21](#_Toc183031697)

[Key Findings 21](#_Toc183031698)

[Recommendations 22](#_Toc183031699)

[References 23](#_Toc183031700)

# Lab Setup

The lab environment consists of two Virtual Machines configured to simulate a penetration testing scenario:  
1. Attacker VM: Running Kali Linux 2024.3 to execute reconnaissance, exploitation, and enumeration tasks.  
2. TargetVM: Hosting the vulnerable Jangow 1.0.1 virtual machine on Oracle VirtualBox 7.0.  
  
Both Virtual Machines are configured on the same network, with the virtual machine set up in bridged adapter mode and promiscuous mode enabled to allow seamless interaction between the attacker and target systems.

## Network Configuration

Target System: Configured with an IP address assigned dynamically upon booting, ensuring connectivity within the network.  
Attacker System: Fully configured with penetration testing tools to interact with the target environment.

A screenshot of a computer

Description automatically generated

Figure Jangow VM and the Network Settings

# Reconnaissance

IP Identification:  
Upon booting the vulnerable virtual machine, the IP address was identified as 192.168.0.13, as displayed during initialization. Preliminary observations indicate that the machine was developed by a Spanish creator, suggesting the presence of Spanish language elements in its content.  
  
A screenshot of a computer

Description automatically generated

Figure - IP address of the Jangow VM

Initial Nmap Scan:  
An Nmap scan was conducted to identify open ports and services running on the target machine. The command was:

**nmap [ip] -sV -sC**

* sV: Enumerates software versions of services
* sC: Performs a script scan utilizing Nmap’s default scripts for vulnerability scanning, basic reconnaissance, and information gathering.

(Explain Shell, 2024)

## Scan Results

Open Ports:

•Port21: FTP Service

•Port 80: HTTP Service

-The HTTP service indicates potential vulnerabilities that could be exploited further

A screen shot of a computer

Description automatically generated

Figure - Nmap scan results with the switches -sV -sC

# Gaining Access

## Enumeration of Web Directory:

From the results of the Nmap scan, a directory named site/ was identified. Accessing this directory revealed a webpage with interactive elements:

* About and Projects sections: Navigated within the webpage.
* Buscar: Redirected to a blank webpage, presenting an opportunity for further enumeration

A screenshot of a computer

Description automatically generated

Figure - Web server of the IP address of Jangow

**A screenshot of a computer

Description automatically generated**

Figure - Webpage of the site directory

**A computer screen with a black rectangular frame

Description automatically generated with medium confidence**

Figure -Web Page after Clicking on Buscar

A screenshot of a computer

Description automatically generated

Figure - Performing ls -al on the site/ directory

## Access Attempt 1: Credentials from config.php

A WordPress directory was discovered via the browser source code after typing the command **ls -al** into the URL bar (Figure7). After navigating to the directory of /site/wordpress, the webpage was discovered broken and did not yield actionable information (Figure 8). Returning to the Buscar page, the **ls -al** command uncovered additional files, including config.php (Figure 9).

Using the cat command, the contents of config.php were revealed to include the following:

* Server Name
* Database Name
* Username
* Password

Fig 12.

These credentials were tested to establish an FTP connection to the target machine, this was achieved using the command:

**ftp [ip address]**

* ftp- Internet File Transfer Program, The program allows a user to transfer files to and from a remote network site.

(Ubuntu Manuals, 2024)

However, the attempt resulted in an FTP error code 530, indicating an authentication failure. Possible reasons could be (Figure 11):

* Incorrect Credentials
* Database connection issues
* Firewall restrictions
* Network connectivity problems

A screenshot of a computer

Description automatically generated

Figure - The broken wordpress directory

A screenshot of a computer

Description automatically generated

Figure - Results of ls -al on Wordpress

A screenshot of a computer

Description automatically generated

Figure - The credentials inside the Config.php file

A screenshot of a computer

Description automatically generated

Figure -FTP connection attempt with the credentials provided in the config.php file

## Access Attempt 2: Credentials from the Backup File

Further exploration of the parent directory /var/www/html revealed a .backup file (Figure 12). Examining its contents using the **cat** command uncovered another set of credentials (Figure 13). These credentials were used to iniate a second FTP login attempt.

This attempt was successful, granting access to the server as the user jangow01 (Figure 14).

A screenshot of a computer

Description automatically generated

Figure - Exploration of the parent directory using **ls -al**

A screenshot of a computer

Description automatically generated

Figure - The credentials located in the .backup file

A screenshot of a computer

Description automatically generated

Figure - Successful FTP Login after using the credentials found in the .backup file

# Privilege Escalation

After obtaining user-level access as jangow01, the next objective was to escalate privileges to the root account. To maintain access to the target system and execute further actions, establishing a reverse shell was necessary. Below are the two attempts made to achieve this:

## Attempt 1: PHP Reverse Shell via Local File Inclusion

The first attempt leveraged the Local File Inclusion (LFI) vulnerability identified earlier to push a PHP reverse shell payload onto the target machine. The steps included:

1. Searching online for publicly available PHP reverse shell scripts, such as those found on GitHub. This particular script was found at this link: <https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>
2. Editing the script using nano to modify parameters, such as the attacker’s IP address and the target port.
3. Saving the customized script and re-establishing an FTP connection as the **jangow01** user (using the credentials found in the .backup file).
4. Navigating to the /var/www/html directory and attempting to upload the PHP payload.

A screenshot of a computer

Description automatically generated

Figure - Snapshot of attempting to push the payload onto the webserver

As seen on the above screenshot, the attempt was unsuccessful due to an 553 error, and after research into the combination of the errors displayed suggested that the user account of jangow01 does not possess the permissions to make changes to a web server. This attempt was a failure.

## Attempt 2 Reverse Shell via Bash Shell Script

This attempt utilised a Bash based reverse shell, delivered via a URL encoded payload. The process involved:

1. Finding a suitable Bash reverse shell script on <https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet>
2. Copying the script into a text editor and modifying the code to include the attacker’s IP address and port number
3. Encoding the script using a URL encoder to enable it to be executed directly from the URL bar of the target webserver
4. Setting up a Netcat listener on the attacker’s machine using the command:

**nc -lvnp [port number].**

* nc- netcat is a simple Unix utility which reads and writes data across network connections, using TCP or UDP protocol
* -lvnp:
  + -l: Listen mode, for inbound connecs
  + -v: verbose
  + -n: numeric-only IP addresses, no DNS
  + -p: local port number

(Explain Shell, 2024)

* 1. The port number here matched the port number used in the Bash Script

This attempt was successful, as the reverse shell connection was established via the Netcat listener, providing access to the target.

A screenshot of a computer

Description automatically generated

Figure - The snapshot of the reverse shell captured on port 443 using the netcat listener

## Post-Connection Steps: Making the Reverse Shell Interactive

After successfully capturing the reverse shell connection, the next objective was to make the shell fully interactive. This was necessary to improve usability and enable more effective interaction with the target system. Below are the steps taken to achieve this:

### Upgrading to an interactive Shell

The initial reverse shell was a limited, non-interactive connection. To enhance its functionality, the following Python command was executed on the shell:

**python3 -c ‘import pty;pty.spawn(“/bin/bash”)’**

* python3: Launches the Python3 interpreter, which is often available on Linux systems.
* -c: Executes the following string as a Python code
* Import pty: imports Python’s pseudo-terminal (pty) module, which enables terminal emulation

(elastic, 2024)

This command allowed the reverse shell to act like a regular shell session, improving command-line usability (Figure 17).

### Setting the terminal type

To optimise the shell for linux-based operations, the terminal type was set to term. This was achieved by exporting the terminal environment variable:

**export TERM=xterm**

* export: Sets an environment variable for the current session
* TERM=xterm: Specifies the terminal typer as xterm, which supports a wide range of functionalities, including color coding, command history, and better text rendering

(Dickey, 2024)

This step made sure that the shell behaved like a fully interactive Linux Terminal .

### Switching User with su Command

With the interactive shell active, the next step was to switch to the jangow01 user account to gain additional privileges. Using the credentials retrieved earlier, the following command was executed (Figure 18):

**su jangow01**

* su: Switches to another user account, prompting for the password
* jangow01: The username of the account to switch to

(explainshell, 2024)

### Discovery of the User Flag

After successfully capturing the reverse shell connection and upgrading it to an interactive shell, the next step was to explore the directories accessible under the jangow01 user account. Navigating to the /home/jangow01 directory revealed a key file of interest.

Steps taken to discover the Flag:

1. Executing the ls command in the /home/jangow01 directory displayed a file named user.txt (Figure19)
2. The user.txt file was opened using the cat command to reveal its contents
3. Inside the user.txt file was a string of characters representing the user flag, marking a significant milestone in the exploitation process (Figure 20).
   1. The flag was: **d41d8cd98f00b204e9800998ecf8427e**

This flag is proof of access to the jangow01 user account.

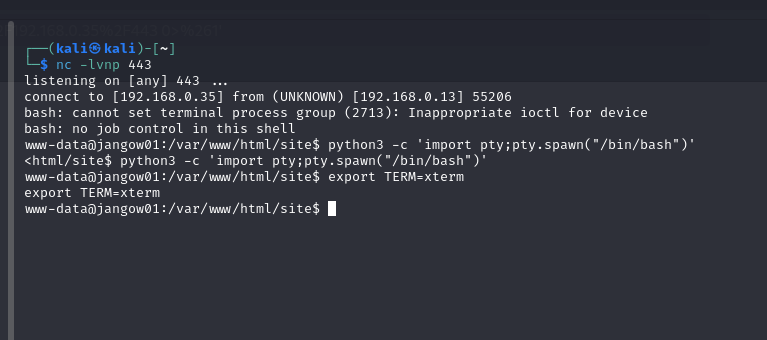
****

Figure - Making the changes to the shell to make it interactive

A screenshot of a computer program

Description automatically generated

Figure - Successfully gaining access to the jangow01 user account via the reverse shell

A screenshot of a computer

Description automatically generated

Figure - Changing directories and finding the user.txt flag

A screenshot of a computer program

Description automatically generated

Figure -FInding the flag in the user.txt file

## Privilege Escalation to Root Access

After establishing a stable and interactive connection with the target system, the next goal was to escalate privileges to the root account. To achieve this, an automated enumeration tool called LinPEAS was utilized. Below are the steps and commands executed during this process:

### Downloading and uploading LinPEAS

LinPEAS is a privilege escalation script designed to identify misconfigurations and vulnerabilities in Linux systems. The script was downloaded from GitHub, found at this link <https://github.com/peass-ng/PEASS-ng/releases/tag/20241101-6f46e855>, to the attacker’s machine. The following steps were taken to transfer it to the target system:

1. Using the FTP connection as the jangow01 user, the script was uploaded to the target.
2. Before uploading, it was ensured that the current working directory on the attacker’s machine was set to the home directory to allow seamless file transfer.

A screenshot of a computer

Description automatically generated

Figure - Downloading linepeas.sh

**A screenshot of a computer

Description automatically generated**

Figure - Linpeas.sh is the downloads directory

A screenshot of a computer

Description automatically generated

Figure - Pushed the payload through to the jangow01 user account and it confirmed on the right-hand side terminal

### Changing File Permissions

After transferring the linpeas.sh script to the target, its permissions were modified to make it executable. The following command was used:

**chmod +x linpeas.sh**

* + Chmod: Command to change file permissions
  + +x: Adds executable permissions to the file
  + Linpeas.sh: The name of the file whose permissions are being modified

(Explain Shell, 2024)

This step allowed the script to be executed on the target system.

A screenshot of a computer

Description automatically generated

Figure - Adding execute permission to the linpeas.sh file

### Executing LinPEAS

With the file now executable, the script was run to enumerate the target system for potential privilege escalation paths:

**./linpeas.sh**

The output of LinPEAS provided detailed information about vulnerabilities and misconfigurations, highlighting potential exploitation paths.

Figxx snapshot of the list of Linpeas running and the vulnerabilities available

### Identifying the Vulnerability

After reviewing the output, the first vulnerability identified was selected. The LinPEAS report included a link to an exploit associated with this vulnerability. The following steps were taken:

1. The exploit code was downloaded to the attacker’s machine.
   1. The exploit I downloaded was from the link: <https://www.exploit-db.com/download/45010>
2. The exploit was transferred to the target machine using the same FTP connection as before

A screenshot of a computer

Description automatically generated

Figure - Snapshot of Linpeas executing

A screenshot of a computer program

Description automatically generated

Figure -List of Exploits and download link

A screenshot of a computer

Description automatically generated

Figure - The exploit downloaded and pushed onto the user account of jangow01

### Compiling and Executing the Exploit

To utilize the exploit, it was compiled on the target machine using the following command:

**gcc [filename] -o [different\_filename]-**

* gcc: The GNU Compiler Collection, used for compiling C programs
* filename: The name of the source code file downloaded from the exploit database
* -o different\_filename: Specifies the name of the compiled executable file

A screenshot of a computer program

Description automatically generated

Figure - The exploit has been compiled and ready to execute

Executing the compiled exploit successfully escalated privileges to the root account.

A screenshot of a computer screen

Description automatically generated

Figure - Exploit is run and have gained access to root user account

The command **whoami** was run to show what user account was being used and as shown in Figure 29 it was root.

## Claiming the Root Flag

With root access granted, the final step was to navigate through the system directories, locate the root flag, and claim it to complete the machine.

A computer screen shot of a logo

Description automatically generated

Figure - Proof of the flag recieved

The flag was: da39a3ee5e6b4b0d3255bfef95601890afd80709

# Conclusion

The successful exploitation of the Jangow 1.0.1 vulnerable machine demonstrated the potential risks posed by common misconfigurations and vulnerabilities in web applications and Linux systems. Through a combination of manual techniques and automated tools, the following objectives were achieved:

1. Reconnaissance and Initial Access: Identified open ports and services using Nmap, followed by enumeration of directories and files to uncover sensitive credentials.
2. Privilege Escalation: Utilized tools like LinPEAS to enumerate the system and identify an exploitable vulnerability, leading to root access.
3. Flag Discovery: Retrieved the user.txt and root.txt flags as proof of successful exploitation.

## Key Findings

Weak Credential Management: Credentials stored in accessible files such as config.php and .backup present significant security risks.

Improper Directory Permissions: Misconfigured permissions allowed for unnecessary access and the ability to enumerate sensitive information.

Unpatched Vulnerabilities: The system contained vulnerabilities that were exploited for privilege escalation.

## Recommendations

To mitigate similar risks in production systems, the following actions are recommended:

Secure Configuration: Ensure proper permissions and access controls for critical files and directories.

Credential Management: Avoid storing plain-text credentials in web-accessible locations. Use environment variables or secret management tools.

Patch Management: Regularly update and patch systems to address known vulnerabilities.

Regular Security Audits: Conduct regular penetration tests and audits to identify and address security gaps proactively.

Monitoring and Logging: Implement robust logging and monitoring to detect unauthorized access or malicious activity.

This assessment highlights the importance of a layered defense strategy to protect against unauthorized access and privilege escalation. The findings and recommendations provided here serve as a roadmap for improving the security posture of similar systems.

# References

Dickey, T. E. (2024, November 20). *invisible-island.net*. Retrieved from xterm: https://invisible-island.net/xterm/

elastic. (2024, November 18). *Interactive Terminal Spawned via Python*. Retrieved from Elastic.co: https://www.elastic.co/guide/en/security/8.15/prebuilt-rule-8-3-1-interactive-terminal-spawned-via-python.html

Explain Shell. (2024, November 17). *Explain Shell*. Retrieved from https://explainshell.com/explain?cmd=nc+-lvnp

Explain Shell. (2024, Novmeber 20). *Explain Shell*. Retrieved from https://explainshell.com/explain?cmd=chmod+%2Bx+

Explain Shell. (2024, November 10). *Explain Shell*. Retrieved from https://explainshell.com/explain?cmd=nmap+ip+-sV+-sC

explainshell. (2024, November 20). *explain shell*. Retrieved from https://explainshell.com/explain?cmd=su+jangow01

Ubuntu Manuals. (2024, November 20). *Ubuntu Manpage Repository*. Retrieved from Ubuntu Manuals: https://manpages.ubuntu.com/manpages/oracular/en/man1/ftp-ssl.1.html#name