Network Penetration Testing Report

MR ROBOT box

Anbarasan K

22MIS1173

Executive summary

This report briefly explains what kind of vulnerability is there in the provided network, what kind of steps and techniques has been followed to exploit the give network, what is severity of the vulnerability and what are the mitigation steps to can be taken to avoid the vulnerability.

Technical Details

Bug name: WordPress Theme Workreap 2.2.2 - Unauthenticated Upload Leading to Remote Code Execution

CVE: 2021-24499

EDB-ID: 51510

Description: The Workreap WordPress theme before 2.2.2 AJAX actions workreap_award_temp_file_uploader and workreap_temp_file_uploader did not perform nonce checks, or validate that the request is from a valid user in any other way. The endpoints allowed for uploading arbitrary files to the uploads/workreap-temp directory. Uploaded files were neither sanitized nor validated, allowing an unauthenticated visitor to upload executable code such as php scripts.

Severity:

CVSS version 3.0 – Base score: 9.8 Critical

CVSS version 2.0 – Base score: 7.5 High

Working: Uploaded files represent a significant risk to applications. The first step in many attacks is to get some code to the system to be attacked. Then the attacker only needs to find a way to get the code executed. Using a file upload helps the attacker accomplish the first step.

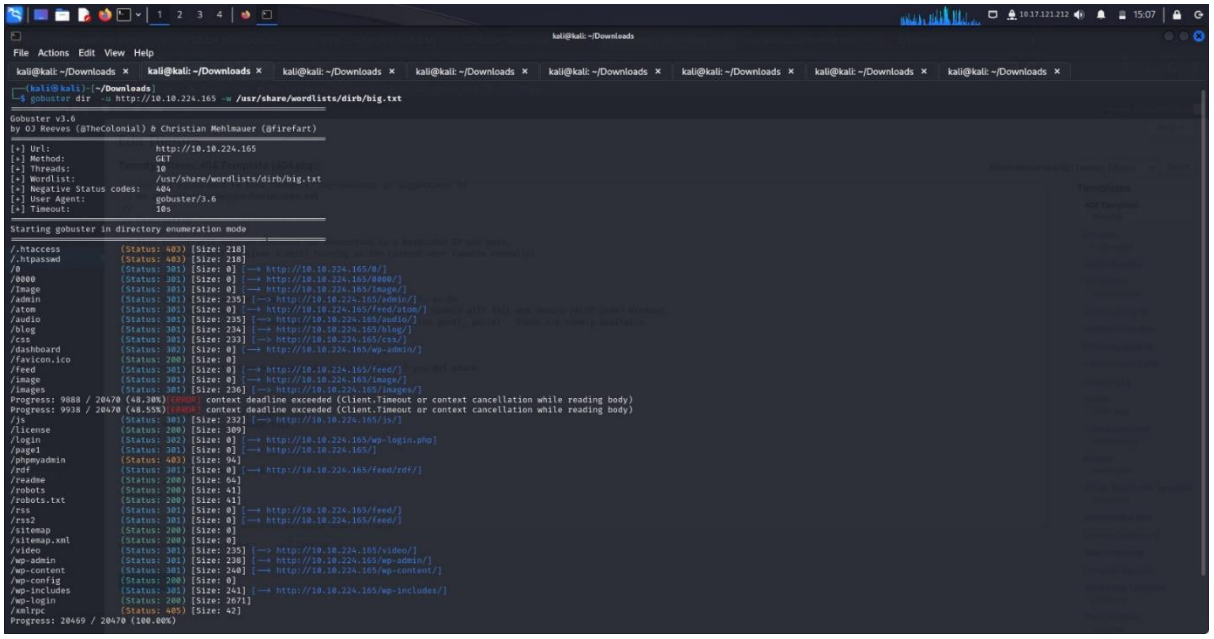
Consequence: The consequences of unrestricted file upload can vary, including complete system takeover, an overloaded file system or database, forwarding attacks to back-end systems, client-side attacks, or simple defacement. It depends on what the application does with the uploaded file and especially where it is stored.

Proof of concept

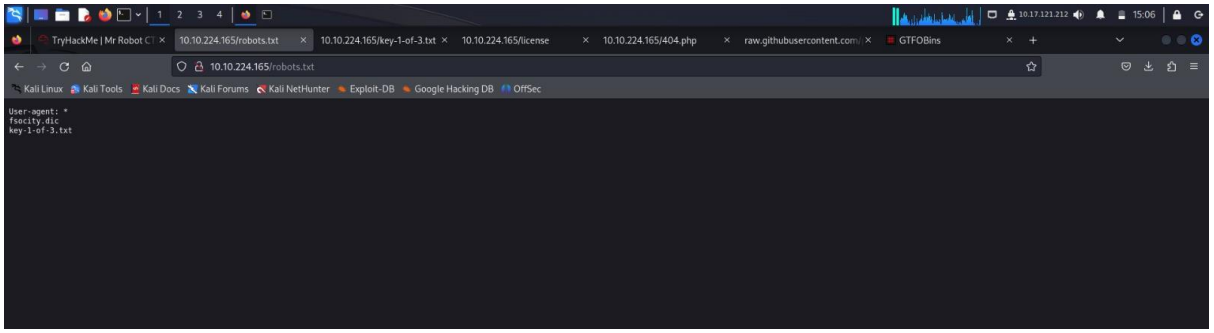
Steps to reproduce

- Do basic **nmap** scan

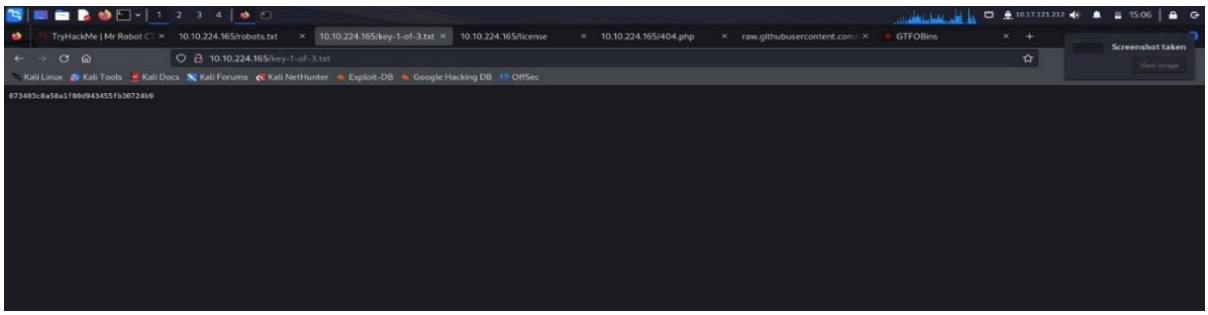
- I used **gobuster** to enumerate hidden directories on the target website using a common wordlist.
- This scan revealed several directories, some of which appeared interesting, like **/admin**, **/login**, **/license** and others worth investigating further.



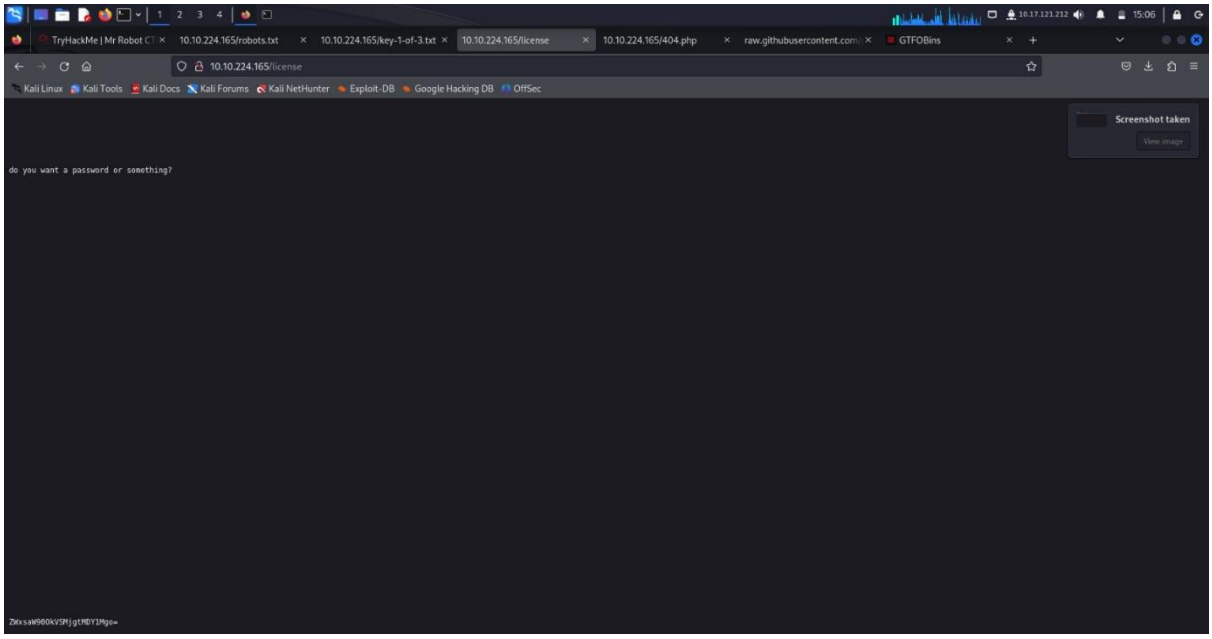
- When we search for the **robots.txt** directory we get a page containing some information



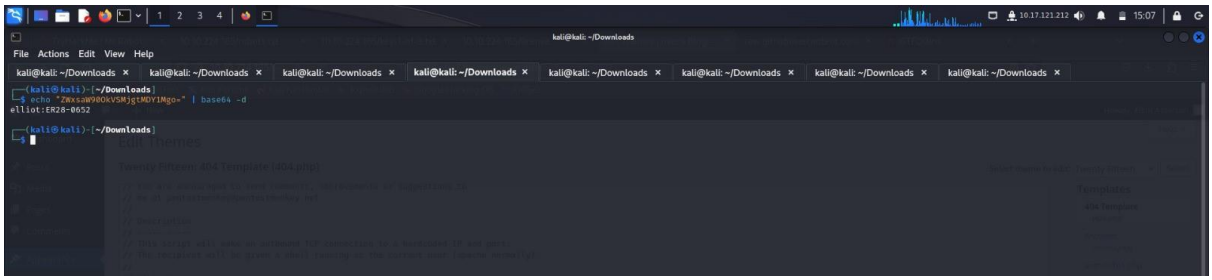
- In that above page we have **key-1-of-3.txt**, when we run this **key-1-of-3.txt** we get another page which have the required **flag 1**



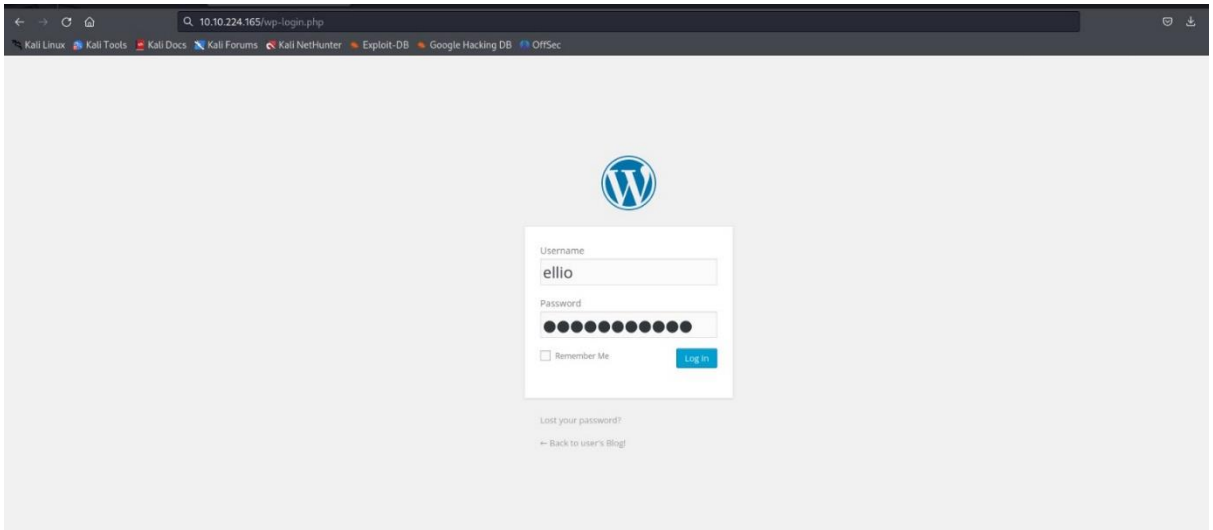
- Similarly, when we navigated to the **license** directory, where we get another web page which contains some encrypted data **ZWxsaW900kVSMjgtMDY1Mgo=**



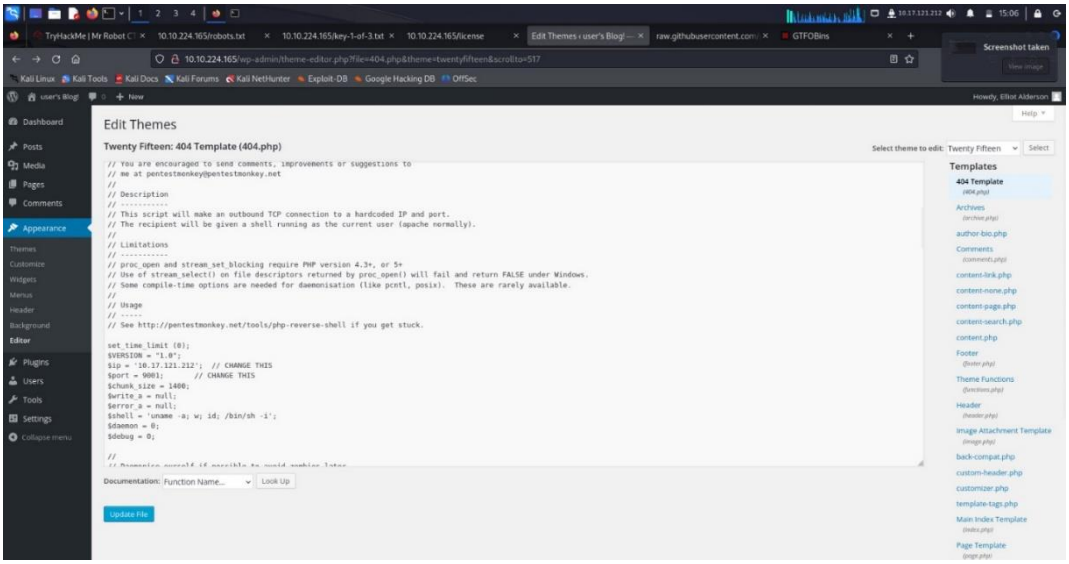
- I have decrypted the data using the **| base64 -d** command and get the result as **elliot: ER28-0652**



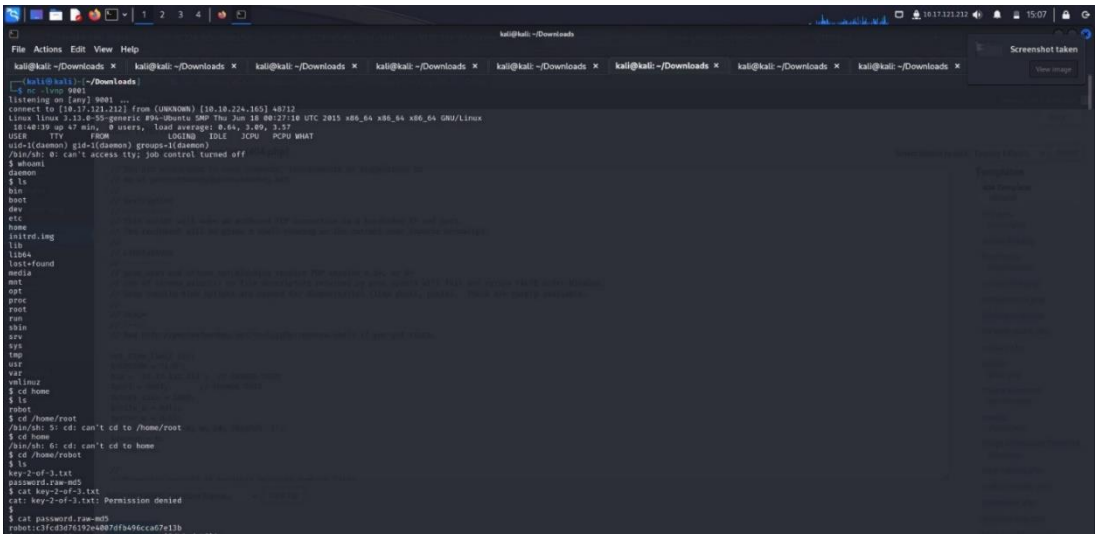
- I navigated to the **login** directory, which redirected me to a **WordPress login page**.
- Using the **username** and **password** obtained from the previously cracked hash, I successfully logged into the WordPress admin panel.



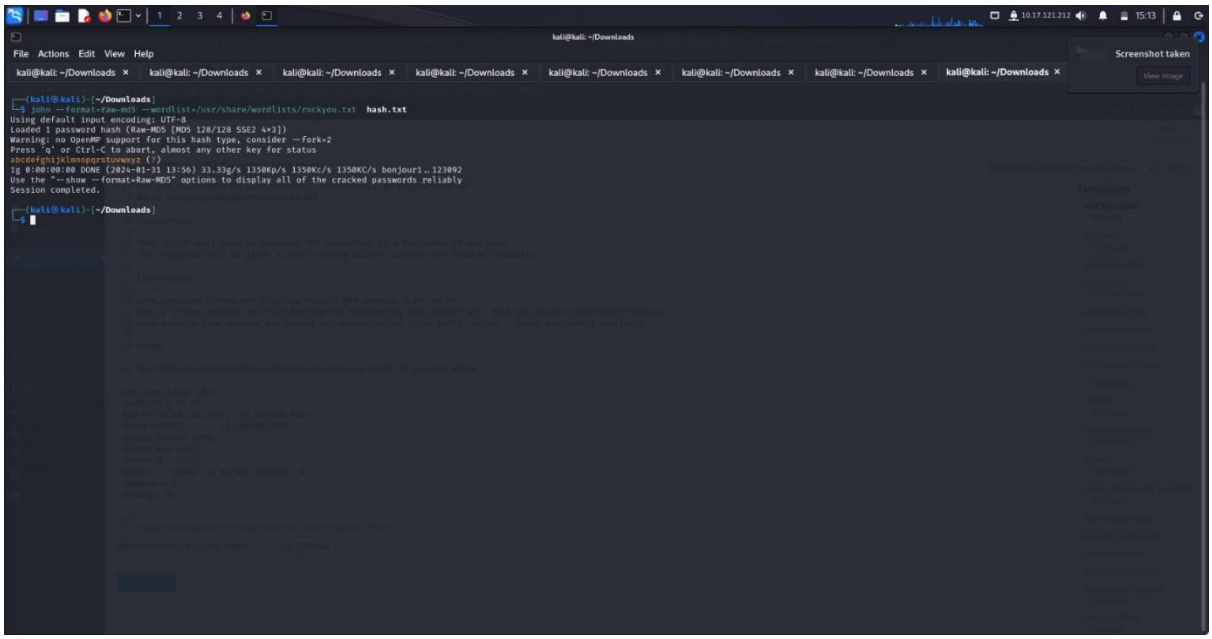
- After logging in, I accessed the **WordPress dashboard**.
- I navigated to **Appearance → Theme Editor**, where several PHP files were available for editing.
- I uploaded a **PHP reverse shell exploit** by modifying one of the existing theme files (e.g., 404.php).
- In the exploit file, I specified **my device's IP address and listening port** to establish a reverse shell connection back to my system.



- Using the **nc -lvnp** command, I established a reverse shell connection to the victim's system.
- I navigated to the /home/robot directory and listed its contents, where I found two files**: key-2-of-3.txt** and **password.raw-md5.**



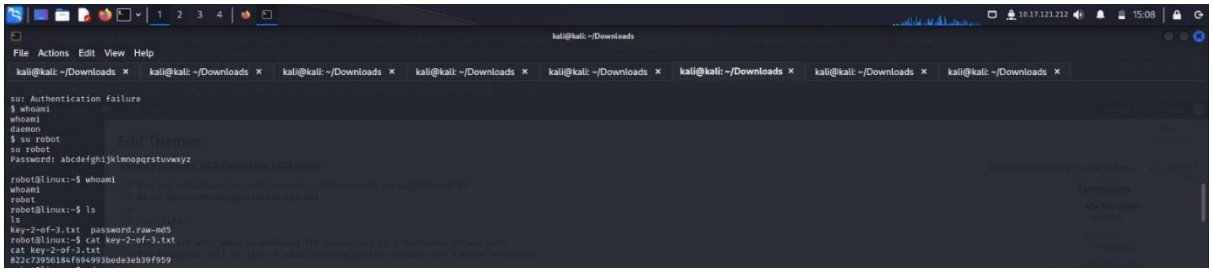
- Upon reading **password.raw-md5**, I discovered an encrypted hash. However, I was initially unable to access the contents of key-2-of-3.txt due to permission restrictions.
- So I have extracted the encrypted password from the hash file and used **John the Ripper** to successfully crack it, revealing the plaintext password



- To get a **fully interactive shell**, I used the command:
  python -c 'import pty; pty.spawn("/bin/sh")'
- After that, I switched to the **robot user** using the password that I had obtained earlier during the enumeration phase.

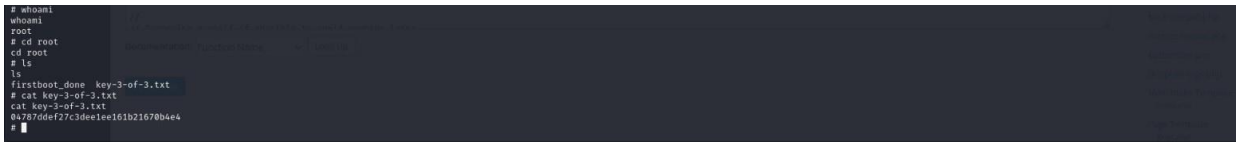
```
$ python -c 'import pty; pty.spawn("/bin/sh")'
$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz
```

- After switching to the **robot** user using the cracked password, I was granted the necessary permissions to read the file **key-2-of-3.txt**.
- This file contained the second flag: flag 2.



- To escalate privileges and gain root access, I ran the command **find / -perm -4000 2>/dev/null | grep '/bin/'** to search for binaries with the SUID permission.
- Among the results, I identified nmap as an exploitable binary. I then executed nmap **--interactive**, entered its interactive mode, and used the **!sh** command to spawn a shell with root privileges.

- Now we list the contents inside the root and we have **key-3-of-3.txt**
- By reading this file we have found the required **flag 3**



Mitigation

The file types allowed to be uploaded should be restricted to only those that are necessary for business functionality.

Never accept a filename and its extension directly without having an allow list filter.

The application should perform filtering and content checking on any files which are uploaded to the server. Files should be thoroughly scanned and validated before being made available to other users. If in doubt, the file should be discarded.

It is necessary to have a list of only permitted extensions on the web application. And, file extension can be selected from the list. For instance, it can be a “select case” syntax (in case of having VBScript) to choose the file extension in regards to the real file extension.

All the control characters and Unicode ones should be removed from the filenames and their extensions without any exception

Uploaded directory should not have any “execute” permission and all the script handlers should be removed from these directories.

Thank you