



How to OSCP

Nathan Subra

Intro

Sysadmin ~10 Years

Information Security ~5 Years

Industries:

Financial

Government


Defense

Current:

Overly Opinionated Pen Tester

Opinions are my own and not the views of my employer, government, spouse etc.

natesubra.com

 @natesubra

SecDSM



We meet monthly on the 3rd Thursday of every month starting at 6pm at The Forge By Pillar Technology in Downtown Des Moines.

Free food/drinks/beer
No registration required

secdsm.org

 [@SecDSM](https://twitter.com/SecDSM)

Offsec Disclaimer

Offensive Security has a strict academic policy:

No Spoilers about the lab (we'll talk about this more later)

Cheating in the Exam/Sharing Exam details == life ban

<https://support.offensive-security.com/#!academic-policy.md>

What is the OSCP?

Offensive Security Certified Professional <- Certification

Attached to the **Pentesting with Kali Course** (shorthand: 'PWK')

Offered by **Offensive Security company**

Course consists of **PDF+Videos** w/ attached Lab time and 1 Exam voucher.

~~Materials are ALL INCLUSIVE and will teach you EVERYTHING YOU NEED TO KNOW~~

Lab

~44 Hands-On Exercises in PDF

~4 Networks

>50 systems in the lab

They'll provide a VM (or build your own)

PWK VM is a tweaked version of Kali --

I highly recommend using theirs

Access to the lab via VPN

30/60/90 day (get 90)

Cost / CPE

30/60/90 day (get 90)

Penetration Testing with Kali + 30 days Lab access
+ Certification USD 800.00

Penetration Testing with Kali + 60 days Lab access
+ Certification USD 1000.00

Penetration Testing with Kali + 90 days Lab access
+ Certification USD 1,150.00

Exam retakes: \$60

40 (ISC)² CPE Credits

This course may qualify you for 40 (ISC)² CPE Credits after you submit your documentation at the end of the course or pass the certification challenge.

Time Requirements

How long it took me: Working full time + new job 90d+30d+15d (actual used ~50)

Plan 2-4 hour blocks at a minimum. There is some ramp up time

Significant Time Commitment / Self Paced

Exam

Point Based (0-100)

Live network with **5 machines** worth varying points

24 Hour Window to achieve 70 points (hack == full shell w/ root or system)

Some points given for low privilege

24 Hours after exam window to submit a report detailing your work

Strict documentation requirements

Technology restrictions: (Metasploit restrictions, automation restrictions)

10 “extra” points: Submit your lab exercises and 10 documented LAB machines - OPTIONAL (but do it)

Recommended Experience

Nothing required!

The more you know in each area, less time you'll spend researching in the lab

BUT, helpful to have knowledge of:

Networking (OSI)

Scripting (bash, python, perl, powershell)

CLI (Windows and Linux)

Assembly (x86)/Debuggers

C

Javascript

SQL (syntax, mssql/mysql)

Metasploit (mostly meterpreter/msfvenom)

Server side languages (php,asp)

Who should take it?

Everyone!

Defenders: Better understand how attackers work and think

Attackers: Better your skills!

Everyone: Building critical thinking skills, methodology, good note taking

Things you can learn from the OSCP

Common tools and usage

Common attack vectors

How to write documentation/workflow/checklists

How to develop your own Methodology

Critical thinking (cover all the angles/think sideways)

All the different ways you can mess up the above things

Things you probably WON'T LEARN from the OSCP

- Offensive “Tradecraft” is only covered briefly
 - Minimal Stealth
 - Minimal Anti-Forensics
 - Minimal Firewall/IDS evasion
- Post-Ex coverage is weak (we’ll talk about this later)
 - root/system ---> ????
- Many lab technologies are “older” (still valuable, but different)
- Persistence/C2

Tools - Using them

Dig into every tool, understand how they work.

`<command> -h`

`<command> --help`

`man <command>`

Tools - Know your tools (and try different ones)

What other tools offer that same capability? Are you fully utilizing your tools?

Did you know?

Nmap can:

- Scan Networks

- Scan for vulnerabilities

- Exploit vulnerabilities

- Brute force passwords

- Craft packets like scapy

Tools - Write your own

Scripting - if you have to do something more than once -- it's probably worth the time to script it

Tools - Mistakes I made

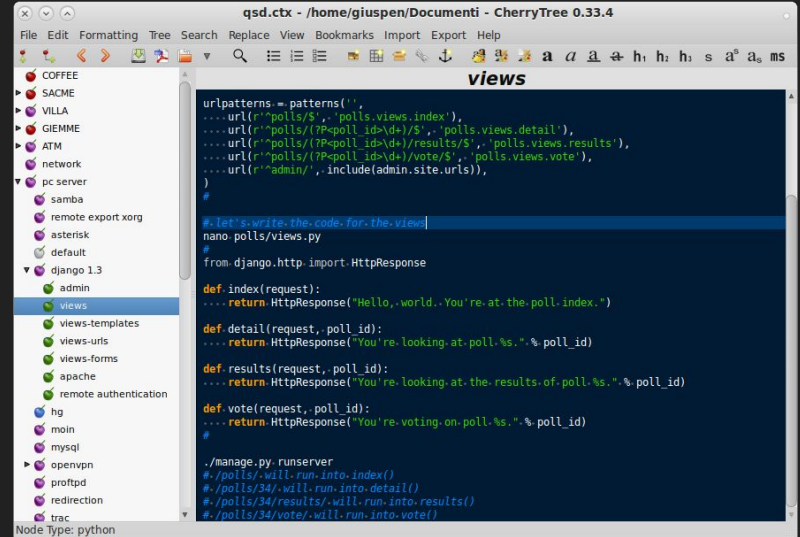
The internet is not a substitute for reading man pages/documentation

Versions aren't important right?

Not always optimal to write your own, consider modifying others

Documentation - Pick a tool

- OneNote (cloud)
- Evernote (cloud)
- Keepnote (old, but reliable)
- Cherrytree (What I used)
- Text files
- Latex (Awesome for reporting later)
- Markdown
- ??????



The screenshot shows a window titled "gisd.ctx - /home/giuspen/Documenti - CherryTree 0.33.4". The left sidebar displays a file tree with folders like COFFEE, SACME, VILLA, GIEMME, ATM, network, pc server, samba, remote export xorg, asterisk, default, and django 1.3. Under "django 1.3", there are sub-folders for "admin", "views", "views-templates", "views-urls", "views-forms", "apache", and "remote authentication". The "views" folder is selected. The main pane shows the contents of the "views" file, which is a Python file named "views.py". The code includes URL patterns for "polls" and "admin", and view functions for "index", "detail", "results", and "vote".

```
urlpatterns = patterns('',
    ...url(r'^polls/$', 'polls.views.index'),
    ...url(r'^polls/(?P=poll_id\d+)/$', 'polls.views.detail'),
    ...url(r'^polls/(?P=poll_id\d+)/results/$', 'polls.views.results'),
    ...url(r'^polls/(?P=poll_id\d+)/vote/$', 'polls.views.vote'),
    ...url(r'^admin/', include(admin.site.urls)),
)

# Let's write the code for the view
nano polls/views.py
#
from django.http import HttpResponse

def index(request):
    ...return HttpResponse("Hello, world. You're at the poll index.")

def detail(request, poll_id):
    ...return HttpResponse("You're looking at poll %s." % poll_id)

def results(request, poll_id):
    ...return HttpResponse("You're looking at the results of poll %s." % poll_id)

def vote(request, poll_id):
    ...return HttpResponse("You're voting on poll %s." % poll_id)
#

./manage.py runserver
# /polls will run into index()
# /polls/34 will run into detail()
# /polls/34/results will run into results()
# /polls/34/vote will run into vote()
```

Documentation - Types (capture all the things)

- Notes
- Screenshots
- Files you capture/use
- Reports



Documentation - Formatting

Formatting Matters

0-Notes (narrative of work, remember wtf you were doing)

1-Enumeration

- * Port

- * <Tool name> Output

2-Exploit

- * port/url

- * source code/script etc

3-Post_Exploit_Low

- * /etc/password

- * systeminfo

4-Privilege_Escalation

- * source code/script

5-Post_Exploit_High

- * hashes / shadow

Bottom Line, pick what works for you.

As a whole, the industry sucks right now at good documentation.

MAKE SURE THEY ARE BACKED UP

My Take:

Minimum goal should be to be able to recreate the successful exploit entirely from your notes.

Record your failures

Taking generic notes/context is extremely helpful.

Good notes will help you with “Rubber Ducky” hacking (aka, critical thinking outloud)

Documentation - Mistakes

Don't write the report as you go. Make your notes robust enough to write the report.

TAKE MORE SCREENSHOTS (wasted a lot of time going back and doing this)

If you make a checklist, put a version on it, you'll change it and will need to know which machine has run which version (example later)

Save your tool output via copy/paste and Screenshot so you can search later

Narratives are good - you won't remember 2 weeks later. Also when you are doing something dumb it tends to pop out at you when you write it down.

Enumeration

e·nu·mer·a·tion

əˌn(y)uːmərəˈrāSH(ə)n/

noun

noun: **enumeration**; plural noun: **enumerations**

1. the action of mentioning a number of things **one by one**.
2. "the complete enumeration of all possible genetic states"
 - *formal*
 - the action of establishing the number of something.
 - "detailed enumeration of the income of the household"
3. What the offsec staff tells you to do more of

Enumeration

The art of evaluating the attack surface of a system

Every data point can be important

People get stuck when they jump the gun

Just because something “seems” vulnerable...

“Windows 2000! Easy!!! SMB exploit!”

Two hours later... Default password on FTP + ASP Shell -- sigh

Don't assume. Evaluate the entire attack surface, rack and stack, then exploit.

Enumeration - Example (not all inclusive)

1. Staged nmap scan
 - a. Common ports (21,22,80,443 etc)
 - b. Less common ports (8080,10443, etc)
 - c. All ports (UDP ugh)
2. Enumerate each port (banners can lie)
3. poke/prod each port (nc, browser, etc)
4. Default passwords
5. Enumerate Services (nikto, web apps, etc)

Capture screenshots/notes as you go!

“Exploitation”

Misconfigurations are as common as (or common than) exploits

Practice your screenshot techniques (you'll need them for the exam)

Primarily will use exploitdb (searchsploit)

Not every box is able to be attacked directly...

Document everything you try!

Versions matter... most of the time

Don't make assumptions

Exploitation

Some tools I used:

- Browser Developer Tools
- Tamper Data
- Cookie Editor
- Curl
- Bash
- Burp
- Python
- Perl
- Metasploit (but always go back and do it “manually”)
- Burp (like once)
- nc

Privilege Escalation (look familiar?)

Misconfigurations are just as common as exploits

Practice your screenshot techniques (you'll need them for the exam)

Document everything you try!

Don't make assumptions

Some tutorials/primers that were good reads:

<http://www.fuzzysecurity.com/tutorials/16.html>

<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>

Post Exploitation - The other half of work

Proper Post exploitation may take you longer than the hack

Did you?

- Grab/Crack hashes
 - Make wordlist/userlist
 - Spray network
- Search for interesting files (passwords, php config, asp, interesting scripts apps)?
- Enumerate the application?
- Enumerate SQL? (I missed this way too many times)
- See what connects to the box?
 - Pivot
 - Exploitation point?
- Backups?
- Grab config files

My Methodology (finalized)

1. Make an enumeration checklist, stick with it
2. Make a separate list: every time you use a new technique (enumeration/exploit/postex) -- add it to that list
3. When you get stuck, look at that list
4. If you get stuck for a long time, move on and come back
5. Don't slack on documentation
6. **Keep track of all the rabbit holes you went down, don't do those things**

My Methodology - Mistakes

Building a checklist or formal process to start

Hunches will work sometimes, other times you'll waste 8 hours on your hunch

Go slow, be methodical (speed comes with practice)

Collect every bit of information you can (ENUMERATE)

Capture all the documentation

Look at the big picture

Spoilers

Don't give them. Don't ask for them.

Asking for help:

Wrong: “this very specific exploit that I randomly downloaded and ran without even looking at is not working on this specific lab machine, why not????!!!!111”

More wrong: “I'm stuck, I've TRIED EVERYTHING, I need a hint”

The goal is to teach you to figure out the answer, not give you the answer

Some of the best help I had was someone asking what I tried

Resources

Get a study buddy (helping to teach others can help you, someone to talk to when you're frustrated)

Offsec Forums - <https://forums.offensive-security.com>

Netsecfocus Slack - <https://netsecfocus.herokuapp.com/> (ask for inv to OSCP)

Offsec IRC - <https://www.offensive-security.com/offsec-irc-guide/>

OSCP Like machines: <https://www.vulnhub.com/>

Just remember, offsec is always watching, if it's a spoiler-- don't post it. If you're unsure -- ask the staff -- <https://www.offensive-security.com/contact-us/>

<https://gist.github.com/natesubra/5117959c660296e12d3ac5df491da395>

Last thoughts

1. Make a checklist
2. The main Goal of the lab is to learn. Not just root boxes. Use every method you can, multiple times
3. Using Metasploit is OK, just go back and repeat using the manual method
4. Recap: How could I have done it better. Where did I get stuck and why?
5. **Failure == Successfully identifying a method that does not work**



**WHENEVER I'M ABOUT
TO DO SOMETHING, I THINK
"WOULD AN IDIOT DO THAT?"
AND IF THEY WOULD
I DO NOT DO THAT THING**

- DWIGHT SCHRUTE