TRYHACKME-Beginners Edition

This file was created on July 26,2022.
This file is to keep track of basic entry level THM ROOMS.

Day-01:

Room No. 1- OpenVpn

Visit tryhackme.com and practice this room to learn, how to connect with THM Network. This Room will guide you thoroughly so I am not going to put those details here.

Room No. 2-welcome

Same as above room, visit https://tryhackme.com/jr/welcome , and explore on your own.

DAY-02:

Room No. 3:

Information Gathering: How to search effectly on internet to learn about theproblems and techniques.

StegnoGraphy: Stegnography is a technique to hide/embed some data insideother data(jpg, txt etc)

Steghide: This is a tool to perform stegnography in linux. Theuse is really simple and there are two main modes of steghide(embed and extract).

Embed: To embed some data use "steghide embed" and then pass the arguments to specify the files, -ef(embeding file) the data we are suppose tohide inside other file -cf(cover file)
syntax: steghide embed -cf file.jpg -ef file.txt
Then it will ask to enter passphrase to encrypt message, The file ef(file.txt) will be embeded inside cf(file.jpg).

Extract: To extract the data from received_file you have to simply type "steghide extract" and then pass the arguments. -sf(stegofile) is theargument to pass after these parameters.

Syntax: steghide extract -sf file.jpg

Then it will ask you to enter passphrase which will be used to decrypt the data.
After passing correct passphrase the secret file ef(embeded file) will be restored in the folder as it was previously.

Burpsuite: Whether you are a developer or a security professional, understanding how applications are attacked is the key to defending them. Burp Suite is an integrated platform and graphical tool for performing security testingof web applications, it supports the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and exploitingsecurity vulnerabilities.

WebGoat Training Application: manually verifying a XSS (cross-site scripting vulnerability found in the WebGoat training application. The example uses a versionof “WebGoat” taken from OWASP’s Broken Web Application Project

Burp Repeater OR simply Repeater: Mode to send request manually repeatedly.

Vulnerability Searching: Often in hacking you'll come across software that might be open to exploitation. For example, Content Management Systems (such as Wordpress, FuelCMS, Ghost, etc) are frequently used to make setting up a website easier, and many of these are vulnerable to various attacks. So where would we lookif we wanted to exploit specific software?

The answer to that question lies in websites such as:ExploitDB

NVD
CVE Mitre
NVD keeps track of CVEs (Common Vulnerabilities and Exposures) -- whether or not there is an exploit publicly available -- so it's a really good place to look if you're researching vulnerabilities in a specific piece of software.
CVEs take theform: CVE-YEAR-IDNUMBER.
ExploitDB tends to be very useful for hackers, as it often actually contains exploits that can be downloaded and used straight out of the box. It tends to beone of the first stops when you encounter software in a CTF or pentest.

If you're inclined towards the CLI on Linux, Kali comes pre-installed with a tool called "searchsploit" which allows you to search ExploitDB from your own machine. This is offline, and works using a downloaded version of the database, meaning thatyou already have all of the exploits already on your Kali Linux!

SCP is a tool used to copy files from one computer to another.

fdisk is a command used to view and alter the partitioning scheme used on your harddrive.

Netcat is a basic tool used to manually send and receive network requests.

DAY 03:

The Process that pentesters follow is summarized in the following steps:Reconnaissance
Enumeration/Scanning
Gaining Access Privilege
EscalationCovering Tracks
Reporting

Reconnaissance is all about collecting information about your target.
Generally speaking, reconnaissance usually involves no interaction with thetarget(s) or system(s).
Reconnaissance is a pretty simple concept, think about what tools we can use on theinternet to gather information about people.
What websites and technology come to mind to gather information about a targetorganization, technology, or set of individuals?

There are some specialized tools that we can utilize but for thisintroduction, it is good to know the following tools.

Google (specifically Google Dorking)Wikipedia
PeopleFinder.com
who.is
sublist3r
hunter.io
builtwith.com
wappalyzer

The second phase of the Hacker Methodology is Scanning and Enumeration.
This is where a hacker will start interacting with (scanning and enumerating) thetarget to attempt to find vulnerabilities related to the target.
This is where more specialized tools start to come in to the arsenal. Tools like nmap, dirb, metasploit, exploit-db, Burp Suite and others are very useful to helpus try to find vulnerabilities in a target.

In the scanning and enumeration phase, the attacker is interacting with the targetto determine its overall attack surface.
The attack surface determines what the target might be vulnerable to in the Exploitation phase. These vulnerabilities might be a range of things: anything froma webpage not being properly locked down, a website leaking information, SQL Injection, Cross Site Scripting or any number of other vulnerabilities.

NMAP: Nmap is a tool which can scan a target and tell us a wide variety of things:What ports are open (if you don't know anything about ports I highly recommend watching this: https://www.youtube.com/watch?v=PpsEaqJV_A0 and https://www.youtube.com/watch?v=qsZ8Qcm6_8k)
The operating system of the target (Windows, Linux, MacOS, etc. including whatversion of the Operating System)
What services are running and what version of the service (for example, just sayingFTP (File Transfer Protocol) isn't enough - nmap can attempt to fingerprint and determine the exact VERSION of FTP which may enable us to find a specific vulnerability in the target)

Metasploit Project: One common tool used for exploitation is called Metasploitwhich has many built-in scripts to try to keep life simple.
The Metasploit Project is a computer security project that provides information about security vulnerabilities and aids in penetration testing and IDS signaturedevelopment.

You can also used tools like Burp Suite and SQLMap to exploit web applications.There are tools such as msfvenom (for building custom payloads), BeEF (browser-based exploitation), and many many others.

SQL Injection in ID Parameter: Placing a payload of 1' OR '1'='1 into the ID parameter of the website allowed the viewing of all cat names in the cat Table ofthe database. Furthermore, a UNION SELECT SQL statement allowed the attacker to view all usernames and passwords stored in the Accounts table.

DAY 04:

Linux Basics:

commands:

echo-> print given text as output to terminal.whoami-> print linux username

ls-> list all the files/folders in directory you are working currently

pwd-> print the name of present working directorycd-> change directory

cat-> cat command allows us to create single or multiple files, view content of a file, concatenate files and redirect output interminal or files.

find-> directories can contain even more directorieswithin themselves. It becomes a headache when we're having to look through every

single one just to try and look for specific files. We can use find to do just thisfor us! Let's start simple and assume that we already know the name of the file we're looking for — but can't remember where it is exactly! In this case, we're looking for "passwords.txt"
If we remember the filename, we can simply use find -name passwords.txt where thecommand will look through every folder in our current directory for that specificfile like so. But let's say that we don't know the name of the file, or want to search for every file that has an extension such as ".txt". Find let's us do thattoo!
We can simply use what's known as a wildcard (*) to search for anything that has .txt at the end. In our case, we want to find every .txt file that's in ourcurrent directory. We will construct a command such as find -name *.txt . Where
"Find" has been able to find every .txt file and has then given us the location ofeach one:

grep-> We can use grepto search the entire contents of this file for any entries of the value that we are searching for. Going with theexample of a web server's access log, we want to see everything that the IP address"81.143.211.90" has visited (note that this is fictional). "Grep" has searched through this file and has shown us any entries of what we've provided and that is contained within this log file for the IP.

Operators:

operator"&"-> This operator allows you torun commands in the background of your terminal.For example, let's say we want to copy a large file. This will obviously take quite a long time and will leave us unable to do anything else until the file successfully copies. The "&" shell operator allows us to execute a command and have it run in thebackground (such as this file copy) allowing us to do other things!

Operator "&&"-> This shell operator is a bit misleading in the sense of how familiar is to its partner "&". Unlike the "&" operator, we can use "&&" to make a list of commands to run for example command1 &&command2. However, it's worth noting that command2 will only run if command1 was successful.

Operator ">"-> This operator is what'sknown as an output redirector. What this essentially means is that we take the output from a command we run and send that output to somewhere else.
A great example of this is redirecting the output of the echo command that we learned in Task 4. Of course, running something such as echo howdy will return "howdy" back to our terminal — that isn't super useful. What we can do instead, isredirect "howdy" to something such as a new file!
Let's say we wanted to create a file named "welcome" with the message "hey". We canrun echo hey > welcome where we want the file created with the contents "hey" like so.

Operator ">>" -> This operator is also an output redirector like in the previous operator (>) we discussed. However, what makes this operator different is that rather than overwriting any contents within afile, for example, it instead just puts the output at the end.
Following on with our previous example where we have the file "welcome" that has the contents of "hey". If were to use echo to add "hello" to the file using the >operator, the file will now only have "hello" and not "hey". The >> operator allows to append the output to the bottom of the file — rather thanreplacing the contents like so.

SSH:-> Secure Shell or SSH simply is a protocol betweendevices in an encrypted form. Using cryptography, any input we send in a human- readable format is encrypted for travelling over a network -- where it is then unencrypted once it reaches the remote machine.

SSH allows us to remotely execute commands on another

device remotely. Any data sent between the devices is encrypted when it is sentover a network such as the Internet.

Using SSH to Login to Your Linux Machine:

The syntax to use SSH is very simple. We only need to provide two things:
1. The IP address of the remote machine
2. Correct credentials to a valid account to login with on the remote machine

we will be logging in as "tryhackme", whose password is "tryhackme" without the quotation ("") marks. Let's use the IP address of the machine displayed in the cardat the top of the room as the IP address and this user, to construct a command to log in to the remote machine using SSH. The command to do so is ssh and then the username of the account, @ the IP address of the machine.
But first, we need to open a terminal on the TryHackMe AttackBox. There is an iconplaced on the desktop named "Terminal". And now, we can proceed to input commands.For example: ssh tryhackme@10.10.202.70 . Replacing the IP address with the IP address for your Linux target machine. Once executed, we will then be asked to trust the host and then provide a password for the "tryhackme" account, which is also "tryhackme".

Now that we are connected, any commands that we execute will now execute on theremote machine -- not our own.
Note: When you type a password into an ssh login prompt there is no visible feedback -- you will not be able to see any text or symbols appear as you type thepassword. It is still working, however, so just type the password and press enter to login.

SYNTAX: ssh username@ipAddress
do you want to establish a connection(yes/no): yesenter the
password for the remote machine:

DAY 05:

Flags and Switches:

A majority of commands allow for arguments to be provided. These arguments are identified by a hyphen and a certain keyword known asflags or switches.

When using a command, unless otherwise specified, it will perform its default behaviour. For example, ls lists the contents of the working directory. However, hidden files are not shown. We can use flags and switches to extend the behaviour of commands, switches of ls like -a -l etc are additionally used to show the hidden files or show all the files with permission to3groups(user, groups and others).

Determining File Type:

What is often misleading and often catches people out is making presumptions from files as to what their purpose or contents may be.Files usually have what's known as an extension to make this easier. For example, text files usually have an extension of ".txt". But this is not necessary.
So far, the files we have used in our examples haven't had an extension. Without knowing the context of why the file is there -- we don't really know its purpose.Enter the file command. This command takes one argument. For example, we'll use file to confirm whether or not the "note" file in our examples is indeed a text file, like so file note.

FILES PERMISSION SYSTEM:

There are basically three type of users in linux which has access to files and folders. Root user, Groups and Others. Permission to files and folders very from one to others. To check the permissions of all these onthe pwd we use the command "ls -l" which contains the first column of 9 dashesh which represents the Read, Write and Execute permissions of all the files in order of (USER, GROUPS, OTHERS). It actually contains the 10 dashes from which the very left first dash show whether it is a directory or file, blank dash means file and din first dash means directory.

Briefly: The Differences Between Users & Groups

We briefly explored this inLinux fundamentals part 1 (namely, the differences between a regular user and a system user). The great thing about Linux is that permissions can be so granular, that whilst a user technically owns a file, if the permissions have been set, then a group of users can also have either the same or a different set of permissions tothe exact same file without affecting the file owner itself.

Let's put this into a real-world context; the system user that runs a web server must have permissions to read and write files for an effective web application. However, companies such as web hosting companies will have to want to allow theircustomers to upload their own files for their website without being the webserversystem user -- compromising the security of every other customer.

We'll learn the commands necessary to switch between users below.

Switching Between Users:

S witching between users on a Linux install is easy work thanks to the su command. Unless you are the root user (or using root permissions through sudo), then you are required to know two things tofacilitate this transition of user accounts:

The user we wish to switch toThe
user's password
The su command takes a couple of switches that may be of relevance to you. For example, executing a command once you log in or specifying a specific shell to use.I encourage you to read the man page for su to find out more. However, I will coverthe -l or --login switch.
Simply, by providing the -lswitch to su, we start a shell that is much more similarto the actual user logging into the system - we inherit a lot more properties of the new user, i.e., environment variables and the likes.

```
tryhackme@linux2:~$ su user2
Password:
user2@linux2:/home/tryhackme$
```

Common Directories in linux:

/etc:

This root directory is one of the most important root directories on your system. The etc folder (short for etcetera) is a commonplace location to store system files that are used by your operating system.For example, the sudoers file highlighted in the screenshot below contains a list of the users & groups that have permission to run sudo or a set of commands as theroot user.
Also highlighted below are the "passwd" and "shadow" files. These two files are

special for Linux as they show how your system stores the passwords for each userin encrypted formatting called sha512.

/var:

The "/var" directory, with "var" being short for variable data, is one of the main root folders found on a Linux install. This folder stores data that is frequently accessed or written by services or applications running on the system. For example, log files from running services and applications are written here (/var/log), or other data that is not necessarilyassociated with a specific user (i.e., databases and the like).

/root:

Unlike the /home directory, the /root folder is actually the home for the "root" system user. There isn't anything more to this folder other than just understanding that this is the home directory for the "root"user. But, it is worth a mention as the logical presumption is that this user wouldhave their data in a directory such as "/home/root" by default.

/tmp:

This is a unique root directory found on a Linux install. Short for "temporary", the /tmp directory is volatile and is used to storedata that is only needed to be accessed once or twice. Similar to the memory on your computer, once the computer is restarted, the contents of this folder are cleared out.
What's useful for us in pentesting is that any user can write to this folder bydefault. Meaning once we have access to a machine, it serves as a good place tostore things like our enumeration scripts.

DAY 06:

Linux fundamentals part 3 final:best text

Text Editors: Vim and Nano are the 2 editors, available in linux.

Useful Utilities:

wget:

This command allows us to download files from the web via HTTP -- as if you were accessing the file in your browser. We simply need to provide the address of the resource that wewish to download. For example, if I wanted to download a file named "myfile.txt" onto my machine, assuming I knew the web address it -- it would look something likethis:

wget https://assets.tryhackme.com/additional/linux-fundamentals/part3/myfile.txt

SCP(secure copy protocol):

Secure copy, or SCP, is just that -- a means of securely copying files. Unlike the regular cp command, this command allows you to transfer files between two computersusing the SSH protocol to provide both authentication and encryption.
Working on a model of SOURCE and DESTINATION, SCP allows you to:

Copy files & directories from your current system to a remote systemCopy files & directories from a remote system to your current system

Provided that we know usernames and passwords for a user on your current system anda user on the remote system.

for example on my local machine I have a file named file1 and I want to send thecopy of it to the remotely connected machine or any machine which credantials I know. The syntax will be:

scp file1 remoteUserName@remoteIP:/path/to/store/file/fileNameOnRemoteMachine

And now let's reverse this and layout the syntax for using scpto copy a file from aremote computer that we're not logged into:

scp remoteUserName@remoteIP:/path/of/file/fileName nameOfFileToStoreOnLocalMachine

Serving Files From Your Host - WEB:

Ubuntu machines come pre-packaged with python3. Python helpfully provides a lightweight and easy-to-use module called "HTTPServer". This module turns your computer into a quick and easy web server that you can use to serve your own files,where they can then be downloaded by another computing using commands such as curland wget.

Python3's "HTTPServer" will serve the files in the directory that you run the command, but this can be changed by providing options that can be found in the manual pages. Simply, all we need to do is run python3 -m http.server to start themodule!
Now, let's use wgetto download the file using the computer's IP address and the name of the file. One flaw with this module is that you have no way of indexing, soyou must know the exact name and location of the file that you wish to use.

wget http://127.0.0.1:8000/file

wget http://10.10.248.218:8000/file(for a remote deployed machine server)

Processes 101:

Processes are the programs that are running on your machine. They are managed by the kernel, where each process will have an ID associated with it, also known as its PID. The PID increments for the order In which the process starts. I.e. the 60th process will have a PID of 60. We can use the friendly ps command to provide a list of the running processes as our user's session and some additional information such as itsstatus code, the session that is running it, how much usage time of the CPU it is using, and the name of the actual programs that are being executed. To see the processes run by other users and those that don't run from a session (i.e. system processes), we need to provide aux to the ps command like so: ps aux.

Managing processes:

You can send signals that terminate processes; there are a variety of types of

signals that correlate to exactly how "cleanly" the process is dealt with by the kernel. To kill a command, we can use the appropriately named kill command and theassociated PID that we wish to kill. i.e., to kill PID 1337, we'd use kill 1337.

Enter the use of systemctl -- this command allows us to interact with the systemd process/daemon. Continuing on with our example, systemctl is an easy to use commandthat takes the following formatting: systemctl [option] [service]

For example, to tell apache to start up, we'll use systemctl start apache2. Seems simple enough, right? Same with if we wanted to stop apache, we'd just replace the[option] with stop (instead of start like we provided)

We can do four options with systemctl:Start
Stop
Enable
Disable

DAY 07:

NETWORKIN
G:

OSI(Open Systems Interconnection):

The OSI (Open Systems Interconnection) Model is a standardised model which we use to demonstrate the theory behind computer networking. In practice, it's actually the more compact TCP/IP model that real-world networking is based off; however the OSImodel, in many ways, is easier to get an initial understanding from.

The OSI model consists of seven layers:

Table showing the OSI layers: Application, Presentation, Session, Transport,Network, Data Link, Physical

There are many mnemonics floating around to help you learn the layers of the OSImodel -- search around until you find one that you like.

I personally favour: Anxious Pale Shakespeare Treated Nervous Drunks PatientlyLet's briefly take a look at each

of these in turn:

Layer 7 -- Application:

The application layer of the OSI model essentially provides networking options toprograms running on a computer. It works almost exclusively with applications, providing an interface for them to use in order to transmit data. When data is given to the application layer, it is passed down into the presentation layer.

Layer 6 -- Presentation:

The presentation layer receives data from the application layer. This data tends tobe in a format that the application understands, but it's not necessarily in a standardised format that could be understood by the application layer in the receiving computer. The presentation layer translates the data into a standardised format, as well as handling any encryption, compression or other transformations tothe data. With this complete, the data is passed down to the session layer.

Layer 5 -- Session:

When the session layer receives the correctly formatted data from the presentation layer, it looks to see if it can set up a connection with the other computer acrossthe network. If it can't then it sends back an error and the process goes no further. If a session can be established then it's the job of the session layer to maintain it, as well as co-operate with the session layer of the remote computer inorder to synchronise communications. The session layer is particularly important asthe session that it creates is unique to the communication in question. This is what allows you to make multiple requests to different endpoints simultaneously without all the data getting mixed up (think about opening two tabs in a web browser at the same time)! When the session layer has successfully logged a connection between the host and remote computer the data is passed down to Layer 4:the transport Layer.

Layer 4 -- Transport:

The transport layer is a very interesting layer that serves numerous important functions. Its first purpose is to choose the protocol over which the data is to betransmitted. The two most common protocols in the transport layer are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol); with TCP the transmission is connection-based which means that a connection between the computers is established and maintained for the duration of the request. This allows for a reliable transmission, as the connection can be used to ensure that the packets all get to the right place. A TCP connection allows the two computers to remain in constant communication to ensure that the data is sent at an acceptable speed, and that any lost data is re-sent. With UDP, the opposite is true; packets of data are essentially thrown at the receiving computer -- if it can't keep up then that's its problem (this is why a video transmission over something like Skype can be pixelated if the connection is bad). What this means isthat TCP would usually be chosen for situations where accuracy is favoured over speed (e.g. file transfer, or loading a webpage), and UDP would be used in situations where speed is more important (e.g. video streaming).

With a protocol selected, the transport layer then divides the transmission up intobite-sized pieces (over TCP these are called segments, over UDP they're called datagrams), which makes it easier to transmit the message successfully.

Layer 3 -- Network:

The network layer is responsible for locating the destination of your request. For example, the Internet is a huge network; when you want to request information from a webpage, it's the network layer that takes the IP address for the page and figures out the best route to take. At this stage we're working with what is referred to as Logical addressing (i.e. IP addresses) which are still software controlled. Logical addresses are used to provide order to networks, categorising them and allowing us to properly sort them. Currently the most common form of logical addressing is the IPV4 format, which you'll likely already be familiar with(i.e 192.168.1.1 is a common address for a home router).

Layer 2 -- Data Link:

The data link layer focuses on the physical addressing of the transmission. It receives a packet from the network layer (that includes the IP address for the remote computer) and adds in the physical (MAC) address of the receiving endpoint. Inside every network enabled computer is a Network Interface Card (NIC) which comeswith a unique MAC (Media Access Control) address to identify it.

MAC addresses are set by the manufacturer and literally burnt into the card; they

can't be changed -- although they can be spoofed. When information is sent across anetwork, it's actually the physical address that is used to identify where exactly to send the information.

Additionally, it's also the job of the data link layer to present the data in aformat suitable for transmission.

The data link layer also serves an important function when it receives data, as itchecks the received information to make sure that it hasn't been corrupted during transmission, which could well happen when the data is transmitted by layer 1: thephysical layer.

Layer 1 -- Physical:

The physical layer is right down to the hardware of the computer. This is where theelectrical pulses that make up data transfer over a network are sent and received. It's the job of the physical layer to convert the binary data of the transmission into signals and transmit them across the network, as well as receiving incoming signals and converting them back into binary data.

Encapsulation in Networking:

As the data is passed down each layer of the model, more information containing details specific to the layer in question is added on to the start of the transmission. As an example, the header added by the Network Layer would include things like the sourceand destination IP addresses, and the header added by the Transport Layer would include (amongst other things) information specific to the protocol being used. Thedata link layer also adds a piece on at the end of the transmission, which is used to verify that the data has not been corrupted on transmission; this also has the added bonus of increased security, as the data can't be intercepted and tampered with without breaking the trailer. This whole process is referred to as encapsulation; the process by which data can be sent from one computer to another.

Notice that the encapsulated data is given a different name at different steps of the process. In layers 7,6 and 5, the data is simply referred to as data. In the transport layer the encapsulated data is referred to as a segment or a datagram (depending on whether TCP or UDP has been selected as a transmission protocol). At the Network Layer, the data is referred to as a packet. When the packet gets passeddown to the Data Link layer it becomes a frame, and by the time it's transmitted across a network the frame has been broken down into bits.

When the message is received by the second computer, it reverses the process -- starting at the physical layer and working up until it reaches the application layer, stripping off the added information as it goes. This is referred to as de- encapsulation. As such you can think of the layers of the OSI model as existing inside every computer with network capabilities. Whilst it's not actually as clearcut in practice, computers all follow the same process of encapsulation to send data and de-encapsulation upon receiving it.

The processes of encapsulation and de-encapsulation are very important -- not leastbecause of their practical use, but also because they give us a standardised methodfor sending data. This means that all transmissions will consistently follow the same methodology, allowing any network enabled device to send a request to any other reachable device and be sure that it will be understood -- regardless of whether they are from the same manufacturer; use the same operating system; or any other factors.

TCP/IP Model(physically applied model in Networking):

The TCP/IP model is, in many ways, very similar to the OSI model. It's a few years older, and serves as the basis for real-world networking. The TCP/IP model consistsof four layers: Application, Transport, Internet and Network Interface. Between them, these cover the same range of functions as the seven layers of the OSI Model.

The layesrs of the TCP/IP model: Application, Transport,Internet, Network Interface

Note: Some recent sources split the TCP/IP model into five layers -- breaking the Network Interface layer into Data Link and Physical layers (as with the OSI model).This is accepted and well-known; however, it is not officially defined (unlike the original four layers which are defined in RFC1122). It's up to you which version you use -- both are generally considered valid.

You would be justified in asking why we bother with the OSI model if it's not actually used for anything in the real-world. The answer to that question is quitesimply that the OSI model (due to being less condensed and more rigid than the TCP/IP model) tends to be easier for learning the initial theory of networking.

The processes of encapsulation and de-encapsulation work in exactly the same way with the TCP/IP model as they do with the OSI model. At each layer of the TCP/IP model a header is added during encapsulation, and removed during de-encapsulation.

Now let's get down to the practical side of things.

A layered model is great as a visual aid -- it shows us the general process of howdata can be encapsulated and sent across a network, but how does it actually happen?

When we talk about TCP/IP, it's all well and good to think about a table with fourlayers in it, but we're actually talking about a suite of protocols -- sets of rules that define how an action is to be carried out. TCP/IP takes its name from the two most important of these: the Transmission Control Protocol (which we touched upon earlier in the OSI model) that controls the flow of data between two endpoints, and the Internet Protocol, which controls how packets are addressed andsent. There are many more protocols that make up the TCP/IP suite; we will cover some of these in later tasks. For now though, let's talk about TCP.

As mentioned earlier, TCP is a connection-based protocol. In other words, before you send any data via TCP, you must first form a stable connection between the twocomputers. The process of forming this connection is called the three-way handshake.

When you attempt to make a connection, your computer first sends a special request to the remote server indicating that it wants to initialise a connection. This request contains something called a SYN (short for synchronise) bit, which essentially makes first contact in starting the connection process. The server willthen respond with a packet containing the SYN bit, as well as another "acknowledgement" bit, called ACK. Finally, your computer will send a packet that contains the ACK bit by itself, confirming that the connection has been setup successfully. With the three-way handshake successfully completed, data can be reliably transmitted between the two computers. Any data that is lost or corrupted on transmission is re-sent, thus leading to a connection which appears to be lossless.

History:

It's important to understand exactly why the TCP/IP and OSI models were originally created. To begin with there was no standardisation -- different manufacturers followed their own methodologies, and consequently systems made by different manufacturers were completely incompatible when it came to networking. The TCP/IP model was introduced by the American DoD in 1982 to provide a standard -- somethingfor all of the different manufacturers to follow. This sorted out the inconsistencyproblems. Later the OSI model was also introduced by the International Organisationfor Standardisation (ISO); however, it's mainly used as a more comprehensive guide for learning, as the TCP/IP model is still the standard upon which modern networking is based.

Networking Tool "PING":

The ping command is used when we want to test whether a connection to a remote resourceis possible. Usually this will be a website on the internet, but it could also be for a computer on your home network if you want to check if it's configured correctly. Ping works using the ICMP protocol, which is one of the slightly less well-known TCP/IP protocols that were mentioned earlier. The ICMP protocol works onthe Network layer of the OSI Model, and thus the Internet layer of the TCP/IP model. The basic syntax for ping is ping <target>. In this example we are using ping to test whether a network connection to Google is possible:

Ping google.com

Notice that the ping command actually returned the IP address for the Google serverthat it connected to, rather than the URL that was requested. This is a handy secondary application for ping, as it can be used to determine the IP address of the server hosting a website. One of the big advantages of ping is that it's prettymuch ubiquitous to any network enabled device. All operating systems support it outof the box, and even most embedded devices can use ping!
Use "man ping" command on linux to see the syntax and application of pingtool/utility.

Networking Tool "traceroute":

he logical follow-up to the ping command is 'traceroute'. Traceroute can be used tomap the path your request takes as it heads to the target machine.

The internet is made up of many, many different servers and end-points, all networked up to each other. This means that, in order to get to the content you actually want, you first need to go through a bunch of other servers. Tracerouteallows you to see each of these connections -- it allows you to see every intermediate step between your computer and the resource that you requested. Thebasic syntax for traceroute on Linux is this:
traceroute <destination>

By default, the Windows traceroute utility (tracert) operates using the same ICMP protocol that ping utilises, and the Unix equivalent operates over UDP. This can bealtered with switches in both instances.

Networking Tool "WHOIS":

Domain Names -- the unsung saviours of the internet.

Can you imagine how it would feel to remember the IP address of every website youwant to visit? Horrible thought.

Fortunately, we've got domains.

We'll talk a little bit more about how this works in the next task, but for now suffice to know that a domain translates into an IP address so that we don't need to remember it (e.g. you can type tryhackme.com, rather than the TryHackMe IP address). Domains are leased out by companies called Domain Registrars. If you wanta domain, you go and register with a registrar, then lease the domain for a certainlength of time.

Enter Whois.

Whois essentially allows you to query who a domain name is registered to. In Europepersonal details are redacted; however, elsewhere you can potentially get a great deal of information from a whois search.

There is a web version of the whois tool if you're particularly adverse to thecommand line. Either way, let's get started!

(Note: You may need to install whois before using it. On Debian based systems thiscan be done with sudo apt update && sudo apt-get install whois)

Whois lookups are very easy to perform. Just use whois <domain> to get a list ofavailable information about the domain registration:

Networking Tool "DIG":

We talked about domains in the previous task -- now lets talk about how they work.

Ever wondered how a URL gets converted into an IP address that your computer canunderstand? The answer is a TCP/IP protocol called DNS (Domain Name System).

At the most basic level, DNS allows us to ask a special server to give us the IP address of the website we're trying to access. For example, if we made a request towww.google.com, our computer would first send a request to a special DNS server (which your computer already knows how to find). The server would then go looking for the IP address for Google and send it back to us. Our computer could then send the request to the IP of the Google server.

Let's break this down a bit.

You make a request to a website. The first thing that your computer does is check its local cache to see if it's already got an IP address stored for the website; ifit does, great. If not, it goes to the next stage of the process.

Assuming the address hasn't already been found, your computer will then send a request to what's known as a recursive DNS server. These will automatically be known to the router on your network. Many Internet Service Providers (ISPs) maintain their own recursive servers, but companies such as Google and OpenDNS alsocontrol recursive servers. This is how your computer automatically knows where to send the request for information: details for a recursive DNS server are stored in your router. This server will also maintain a cache of results for popular domains;however, if the website you've requested isn't stored in the cache, the recursive server will pass the request on to a root name server.

Before 2004 there were precisely 13 root name DNS servers in the world. These days

there are many more; however, they are still accessible using the same 13 IP addresses assigned to the original servers (balanced so that you get the closestserver when you make a request). The root name servers essentially keep track ofthe DNS servers in the next level down, choosing an appropriate one to redirect your request to. These lower level servers are called Top-Level Domain servers.

Top-Level Domain (TLD) servers are split up into extensions. So, for example, if you were searching for tryhackme.com your request would be redirected to a TLD server that handled .com domains. If you were searching for bbc.co.uk your request would be redirected to a TLD server that handles .co.uk domains. As with root name servers, TLD servers keep track of the next level down: Authoritative name servers.When a TLD server receives your request for information, the server passes it down to an appropriate Authoritative name server.

Authoritative name servers are used to store DNS records for domains directly. In other words, every domain in the world will have its DNS records stored on an Authoritative name server somewhere or another; they are the source of the information. When your request reaches the authoritative name server for the domainyou're querying, it will send the relevant information back to you, allowing your computer to connect to the IP address behind the domain you requested.

When you visit a website in your web browser this all happens automatically, but wecan also do it manually with a tool called dig . Like ping and traceroute, dig should be installed automatically on Linux systems.
This is a lot of information. We're currently most interested in the ANSWER sectionfor this room; however, taking the time to learn what the rest of this means is a very good idea. In summary, that information is telling us that we sent it one query and successfully (i.e. No Errors) received one full answer -- which, as expected, contains the IP address for the domain name that we queried.

Another interesting piece of information that dig gives us is the TTL (Time To Live) of the queried DNS record. As mentioned previously, when your computer queries a domain name, it stores the results in its local cache. The TTL of the record tells your computer when to stop considering the record as being valid --
i.e. when it should request the data again, rather than relying on the cached copy.

The TTL can be found in the second column of the answer section:Results demonstrating

that the TTL of the DNS record is 157

It's important to remember that TTL (in the context of DNS caching) is measured inseconds, so the record in the example will expire in two minutes and thirty-seven seconds.

DAY 08:

NMAP(Basics):

When it comes to hacking, knowledge is power. The more knowledge you have about a target system or network, the more options you have available. This makes it imperative that properenumeration is carried out before any exploitation attempts are made.

Say we have been given an IP (or multiple IP addresses) to perform a security auditon. Before we do anything else, we need to get an idea of the “landscape” we are attacking. What this means is that we need to establish which services are running on the targets. For example, perhaps one of them is running a webserver, and another is acting as a Windows Active Directory Domain Controller. The first stage in establishing this “map” of the landscape is something called port scanning. Whena computer runs a network service, it opens a networking construct called a “port” to receive the connection. Ports are necessary for making multiple network

requests or having multiple services available. For example, when you load several webpages at once in a web browser, the program must have some way of determining which tab is loading which web page. This is done by establishing connections to the remote webservers using different ports on your local machine. Equally, if you want a server to be able to run more than one service (for example, perhaps you want your webserver to run both HTTP and HTTPS versions of the site), then you needsome way to direct the traffic to the appropriate service. Once again, ports are the solution to this. Network connections are made between two ports – an open portlistening on the server and a randomly selected port on your own computer. For example, when you connect to a web page, your computer may open port 49534 to connect to the server's port 443.

As in the previous example, the diagram shows what happens when you connect to numerous websites at the same time. Your computer opens up a different, high- numbered port (at random), which it uses for all its communications with the remoteserver.

Every computer has a total of 65535 available ports; however, many of these are registered as standard ports. For example, a HTTP Webservice can nearly always befound on port 80 of the server. A HTTPS Webservice can be found on port 443.
Windows NETBIOS can be found on port 139 and SMB can be found on port 445. It is important to note; however, that especially in a CTF setting, it is not unheard of for even these standard ports to be altered, making it even more imperative that weperform appropriate enumeration on the target.

If we do not know which of these ports a server has open, then we do not have a hope of successfully attacking the target; thus, it is crucial that we begin any attack with a port scan. This can be accomplished in a variety of ways – usually using a tool called nmap, which is the focus of this room. Nmap can be used to perform many different kinds of port scan – the most common of these will be introduced in upcoming tasks; however, the basic theory is this: nmap will connect to each port of the target in turn. Depending on how the port responds, it can be determined as being open, closed, or filtered (usually by a firewall). Once we knowwhich ports are open, we can then look at enumerating which services are running oneach port – either manually, or more commonly using nmap.

So, why nmap? The short answer is that it's currently the industry standard for areason: no other port scanning tool comes close to matching its functionality (although some newcomers are now matching it for speed). It is an extremely powerful tool – made even more powerful by its scripting engine which can be usedto scan for vulnerabilities, and in some cases even perform the exploit directly!Once again, this will be covered more in upcoming tasks.

For now, it is important that you understand: what port scanning is; why it is necessary; and that nmap is the tool of choice for any kind of initial enumeration.

NMAP Switches:

Like most pentesting tools, nmap is run from the terminal. There are versions available forboth Windows and Linux. For this room we will assume that you are using Linux; however, the switches should be identical. Nmap is installed by default in both Kali Linux and the TryHackMe Attack Box.

Nmap can be accessed by typing nmap into the terminal command line, followed bysome of the "switches" (command arguments which tell a program to do different things) we will be covering below.

All you'll need for this is the help menu for nmap (accessed with nmap -h) and/or the nmap man page (access with man nmap). For each answer, include all parts of theswitch unless otherwise specified. This includes the hyphen at the start (-).

first switch listed in the help menu for a 'Syn Scan'= -sSWhich switch would you

use for a "UDP scan"?= -sU

If you wanted to detect which operating system the target is running on, whichswitch would you use?=-O

Nmap provides a switch to detect the version of the services running on the target.What is this switch?=-sV

The default output provided by nmap often does not provide enough information for apentester. How would you increase the verbosity?=-V

Verbosity level one is good, but verbosity level two is better! How would you setthe verbosity level to two? (Note: it's highly advisable to always use at least this option)= -vv

We should always save the output of our scans -- this means that we only need torun the scan once (reducing network traffic and thus chance of detection), and gives us a reference to use when writing reports for clients.

What switch would you use to save the nmap results in three major formats?= -oAWhat switch would you use to

save the nmap results in a "normal" format?= -oN

A very useful output format: how would you save results in a "grepable" format?= -oG

Sometimes the results we're getting just aren't enough. If we don't care about howloud we are, we can enable "aggressive" mode. This is a shorthand switch that activates service detection, operating system detection, a traceroute and common script scanning.

How would you activate this setting?= -A

Nmap offers five levels of "timing" template. These are essentially used to increase the speed your scan runs at. Be careful though: higher speeds are noisier,and can incur errors!

How would you set the timing template to level 5?= -T5We can also choose

which port(s) to scan.

How would you tell nmap to only scan port 80?= -p 80

How would you tell nmap to scan ports 1000-1500?= -p 1000-1500

A very useful option that should not be ignored:
How would you tell nmap to scan all ports?= -p-

How would you activate a script from the nmap scripting library (lots more on thislater!)?= --script

How would you activate all of the scripts in the "vuln" category?= --script=vuln

SCAN TYPES:

When port scanning with Nmap, there are three basic scan types. These are:

TCP Connect Scans (-sT)
SYN "Half-open" Scans (-sS)UDP
Scans (-sU)
Additionally there are several less common port scan types, some of which we willalso cover (albeit in less detail). These are:

TCP Null Scans (-sN)TCP
FIN Scans (-sF) TCP
Xmas Scans (-sX)
Most of these (with the exception of UDP scans) are used for very similar purposes,however, the way that they work differs between each scan. This means that, whilst one of the first three scans are likely to be your go-to in most situations, it's worth bearing in mind that other scan types exist.

In terms of network scanning, we will also look briefly at ICMP (or "ping")scanning.

TCP Connect Scans:

To understand TCP Connect scans (-sT), it's important that you're comfortable with theTCP three-way handshake. If this term is new to you then completing Introductory Networking before continuing would be advisable.

As a brief recap, the three-way handshake consists of three stages. First the connecting terminal (our attacking machine, in this instance) sends a TCP requestto the target server with the SYN flag set. The server then acknowledges this packet with a TCP response containing the SYN flag, as well as the ACK flag.
Finally, our terminal completes the handshake by sending a TCP request with the ACKflag set.
This is one of the fundamental principles of TCP/IP networking, but how does itrelate to Nmap?

Well, as the name suggests, a TCP Connect scan works by performing the three-way handshake with each target port in turn. In other words, Nmap tries to connect to each specified TCP port, and determines whether the service is open by the responseit receives.

For example, if a port is closed, RFC 793 states that:

"... If the connection does not exist (CLOSED) then a reset is sent in response to any incoming segment except another reset. In particular, SYNs addressed to a non-existent connection are rejected by this means."

In other words, if Nmap sends a TCP request with the SYN flag set to a closed port,the target server will respond with a TCP packet with the RST (Reset) flag set. By this response, Nmap can establish that the port is closed.

If, however, the request is sent to an open port, the target will respond with a TCP packet with the SYN/ACK flags set. Nmap then marks this port as being open (andcompletes the handshake by sending back a TCP packet with ACK set).

This is all well and good, however, there is a third possibility.What if the port is open, but

hidden behind a firewall?

Many firewalls are configured to simply drop incoming packets. Nmap sends a TCP SYNrequest, and receives nothing back. This indicates that the port is being protectedby a firewall and thus the port is considered to be filtered.

That said, it is very easy to configure a firewall to respond with a RST TCP packet. For example, in IPtables for Linux, a simple version of the command wouldbe as follows:

iptables -I INPUT -p tcp --dport <port> -j REJECT --reject-with tcp-reset

This can make it extremely difficult (if not impossible) to get an accurate readingof the target(s).

Which RFC defines the appropriate behaviour for the TCP protocol?= RFC 793

If a port is closed, which flag should the server send back to indicate this?= rst

SYN Scan-TCP:

As with TCP scans, SYN scans (-sS) are used to scan the TCP port-range of a target or targets;however, the two scan types work slightly differently. SYN scans are sometimes referred to as "Half-open" scans, or "Stealth" scans.

Where TCP scans perform a full three-way handshake with the target, SYN scans sendsback a RST TCP packet after receiving a SYN/ACK from the server (this prevents the server from repeatedly trying to make the request). In other words, the sequence for scanning an open port looks like this:

This has a variety of advantages for us as hackers:

It can be used to bypass older Intrusion Detection systems as they are looking out for a full three way handshake. This is often no longer the case with modern IDS solutions; it is for this reason that SYN scans are still frequently referred to as"stealth" scans.
SYN scans are often not logged by applications listening on open ports, as standard practice is to log a connection once it's been fully established. Again, this playsinto the idea of SYN scans being stealthy.
Without having to bother about completing (and disconnecting from) a three-way handshake for every port, SYN scans are significantly faster than a standard TCPConnect scan.

There are, however, a couple of disadvantages to SYN scans, namely:

They require sudo permissions[1] in order to work correctly in Linux. This is

because SYN scans require the ability to create raw packets (as opposed to the fullTCP handshake), which is a privilege only the root user has by default.
Unstable services are sometimes brought down by SYN scans, which could proveproblematic if a client has provided a production environment for the test. All in all, the pros outweigh the cons.

For this reason, SYN scans are the default scans used by Nmap if run with sudo permissions. If run without sudo permissions, Nmap defaults to the TCP Connect scanwe saw in the previous task.

When using a SYN scan to identify closed and filtered ports, the exact same rulesas with a TCP Connect scan apply.

If a port is closed then the server responds with a RST TCP packet. If the port is filtered by a firewall then the TCP SYN packet is either dropped, or spoofed with aTCP reset.

In this regard, the two scans are identical: the big difference is in how theyhandle open ports.

[1] SYN scans can also be made to work by giving Nmap the CAP_NET_RAW, CAP_NET_ADMIN and CAP_NET_BIND_SERVICE capabilities; however, this may not allowmany of the NSE scripts to run properly.

There are two other names for a SYN scan, what are they?= half-open scan, stealth

UDP SCAN:

Unlike TCP, UDP connections are stateless. This means that, rather than initiating a connectionwith a back-and-forth "handshake", UDP connections rely on sending packets to a target port and essentially hoping that they make it. This makes UDP superb for connections which rely on speed over quality (e.g. video sharing), but the lack of acknowledgement makes UDP significantly more difficult (and much slower) to scan.
The switch for an Nmap UDP scan is (-sU)

When a packet is sent to an open UDP port, there should be no response. When this happens, Nmap refers to the port as being open|filtered. In other words, it suspects that the port is open, but it could be firewalled. If it gets a UDP response (which is very unusual), then the port is marked as open. More commonly there is no response, in which case the request is sent a second time as a double- check. If there is still no response then the port is marked open|filtered and Nmapmoves on.

When a packet is sent to a closed UDP port, the target should respond with an ICMP(ping) packet containing a message that the port is unreachable. This clearly identifies closed ports, which Nmap marks as such and moves on.

Due to this difficulty in identifying whether a UDP port is actually open, UDP scans tend to be incredibly slow in comparison to the various TCP scans (in theregion of 20 minutes to scan the first 1000 ports, with a good connection). Forthis reason it's usually good practice to run an Nmap scan with --top-ports <number> enabled. For example, scanning with nmap -sU --top-ports 20 <target>.Will scan the top 20 most commonly used UDP ports, resulting in a much more acceptable scan time.

When scanning UDP ports, Nmap usually sends completely empty requests -- just rawUDP packets. That said, for ports which are usually occupied by well-known

services, it will instead send a protocol-specific payload which is more likely toelicit a response from which a more accurate result can be drawn.

If a UDP port doesn't respond to an Nmap scan, what will it be marked as?= open|filtered

When a UDP port is closed, by convention the target should send back a "portunreachable" message. Which protocol would it use to do so= icmp

NULL, FIN and Xmas Scans:

NULL, FIN and Xmas TCP port scans are less commonly used than any of the others we've covered already, so we will not go into a huge amount of depth here. All three are interlinked and are used primarily as they tend to be even stealthier,relatively speaking, than a SYN "stealth" scan. Beginning with NULL scans:

As the name suggests, NULL scans (-sN) are when the TCP request is sent with no flags set at all. As per the RFC, the target host should respond with a RST if theport is closed.

FIN scans (-sF) work in an almost identical fashion; however, instead of sending acompletely empty packet, a request is sent with the FIN flag (usually used to gracefully close an active connection). Once again, Nmap expects a RST if the portis closed.

As with the other two scans in this class, Xmas scans (-sX) send a malformed TCPpacket and expects a RST response for closed ports. It's referred to as an xmas scan as the flags that it sets (PSH, URG and FIN) give it the appearance of a blinking christmas tree when viewed as a packet capture in Wireshark.

The expected response for open ports with these scans is also identical, and is very similar to that of a UDP scan. If the port is open then there is no response to the malformed packet. Unfortunately (as with open UDP ports), that is also an expected behaviour if the port is protected by a firewall, so NULL, FIN and Xmas scans will only ever identify ports as being open|filtered, closed, or filtered. Ifa port is identified as filtered with one of these scans then it is usually becausethe target has responded with an ICMP unreachable packet.

It's also worth noting that while RFC 793 mandates that network hosts respond to malformed packets with a RST TCP packet for closed ports, and don't respond at all for open ports; this is not always the case in practice. In particular Microsoft Windows (and a lot of Cisco network devices) are known to respond with a RST to any malformed TCP packet -- regardless of whether the port is actually open or not.
This results in all ports showing up as being closed.

That said, the goal here is, of course, firewall evasion. Many firewalls are configured to drop incoming TCP packets to blocked ports which have the SYN flag set (thus blocking new connection initiation requests). By sending requests which do not contain the SYN flag, we effectively bypass this kind of firewall. Whilst this is good in theory, most modern IDS solutions are savvy to these scan types, sodon't rely on them to be 100% effective when dealing with modern systems.

Why are NULL, FIN and Xmas scans generally used?= firewall evasion

Which common OS may respond to a NULL, FIN or Xmas scan with a RST for every port?
=microsoft windows

ICMP SCAN:

On first connection to a target network in a black box assignment, our first objective is toobtain a "map" of the network structure -- or, in other words, we want to see whichIP addresses contain active hosts, and which do not.

One way to do this is by using Nmap to perform a so called "ping sweep". This is exactly as the name suggests: Nmap sends an ICMP packet to each possible IP addressfor the specified network. When it receives a response, it marks the IP address that responded as being alive. For reasons we'll see in a later task, this is not always accurate; however, it can provide something of a baseline and thus is worth covering.

To perform a ping sweep, we use the -sn switch in conjunction with IP ranges whichcan be specified with either a hypen (-) or CIDR notation. i.e. we could scan the 192.168.0.x network using:

nmap -sn 192.168.0.1-254
or

nmap -sn 192.168.0.0/24

The -sn switch tells Nmap not to scan any ports -- forcing it to rely primarily on ICMP echo packets (or ARP requests on a local network, if run with sudo or directlyas the root user) to identify targets. In addition to the ICMP echo requests, the -sn switch will also cause nmap to send a TCP SYN packet to port 443 of the target, as well as a TCP ACK (or TCP SYN if not run as root) packet to port 80 of the target.

How would you perform a ping sweep on the 172.16.x.x network (Netmask: 255.255.0.0)using Nmap? (CIDR notation)= nmap -sn 172.16.0.0/16

NSE(nmap Scripting engine):

The Nmap Scripting Engine (NSE) is an incredibly powerful addition to Nmap, extending its functionality quite considerably. NSE Scripts are written in the Luaprogramming language, and can be used to do a variety of things: from scanning forvulnerabilities, to automating exploits for them. The NSE is particularly useful for reconnaisance, however, it is well worth bearing in mind how extensive the script library is.

There are many categories available. Some useful categories include:safe:- Won't affect the

target
intrusive:- Not safe: likely to affect the targetvuln:- Scan for
vulnerabilities
exploit:- Attempt to exploit a vulnerability
auth:- Attempt to bypass authentication for running services (e.g. Log into an FTPserver anonymously)
brute:- Attempt to bruteforce credentials for running services
discovery:- Attempt to query running services for further information about the

network (e.g. query an SNMP server).

What language are NSE scripts written in?= LUA

Which category of scripts would be a very bad idea to run in a productionenvironment?= intrusive

we looked very briefly at the --script switch for activating NSE scripts from the vuln category using --script=vuln. It should come as no surprise that the other categories work in exactly the same way. If the command --script=safe is run, thenany applicable safe scripts will be run against the target (Note: only scripts which target an active service will be activated).

To run a specific script, we would use --script=<script-name> , e.g. --script=http-fileupload-exploiter.

Multiple scripts can be run simultaneously in this fashion by separating them by acomma. For example: --script=smb-enum-users,smb-enum-shares.

Some scripts require arguments (for example, credentials, if they're exploiting an authenticated vulnerability). These can be given with the --script-args Nmap switch. An example of this would be with the http-put script (used to upload files using the PUT method). This takes two arguments: the URL to upload the file to, andthe file's location on disk. For example:

nmap -p 80 --script http-put --script-args http-put.url='/dav/shell.php',http-put.file='./shell.php'

Note that the arguments are separated by commas, and connected to the correspondingscript with periods (i.e. <script-name>.<argument>).

What optional argument can the ftp-anon.nse script take?= maxlist

Searching for Scripts:

Ok, so we know how to use the scripts in Nmap, but we don't yet know how to findthese scripts.

We have two options for this, which should ideally be used in conjunction with eachother. The first is the page on the Nmap website (mentioned in the previous task) which contains a list of all official scripts. The second is the local storage on your attacking machine. Nmap stores its scripts on Linux at /usr/share/nmap/scripts. All of the NSE scripts are stored in this directory bydefault -- this is where Nmap looks for scripts when you specify them.

There are two ways to search for installed scripts. One is by using the /usr/share/nmap/scripts/script.db file. Despite the extension, this isn't actually a database so much as a formatted text file containing filenames and categories foreach available script.

Nmap uses this file to keep track of (and utilise) scripts for the scripting engine; however, we can also grep through it to look for scripts. For example: grep"ftp" /usr/share/nmap/scripts/script.db.

The second way to search for scripts is quite simply to use the ls command. For example, we could get the same results as in the previous screenshot by using ls -l /usr/share/nmap/scripts/*ftp*:

Note the use of asterisks (*) on either side of the search term

The same techniques can also be used to search for categories of script. Forexample:
grep "safe" /usr/share/nmap/scripts/script.db

Installing New Scripts

We mentioned previously that the Nmap website contains a list of scripts, so, what happens if one of these is missing in the scripts directory locally? A standard sudo apt update && sudo apt install nmap should fix this; however, it's also possible to install the scripts manually by downloading the script from Nmap (sudo wget -O /usr/share/nmap/scripts/<script-name>.nse https://svn.nmap.org/nmap/scripts/<script-name>.nse). This must then be followed upwith nmap --script-updatedb, which updates the script.db file to contain the newly downloaded script.

It's worth noting that you would require the same "updatedb" command if you were tomake your own NSE script and add it into Nmap -- a more than manageable task with some basic knowledge of Lua!

Search for "smb" scripts in the /usr/share/nmap/scripts/ directory using either ofthe demonstrated methods.
What is the filename of the script which determines the underlying OS of the SMBserver?= smb-os-discovery.nse

Firewall Evasion:

We have already seen some techniques for bypassing firewalls (think stealth scans,along with NULL, FIN and Xmas scans); however, there is another very common firewall configuration which it's imperative we know how to bypass.

Your typical Windows host will, with its default firewall, block all ICMP packets.This presents a problem: not only do we often use ping to manually establish the activity of a target, Nmap does the same thing by default. This means that Nmap will register a host with this firewall configuration as dead and not bother scanning it at all.

So, we need a way to get around this configuration. Fortunately Nmap provides anoption for this: -Pn, which tells Nmap to not bother pinging the host before scanning it. This means that Nmap will always treat the target host(s) as being alive, effectively bypassing the ICMP block; however, it comes at the price of potentially taking a very long time to complete the scan (if the host really is dead then Nmap will still be checking and double checking every specified port).

It's worth noting that if you're already directly on the local network, Nmap can

also use ARP requests to determine host activity.

There are a variety of other switches which Nmap considers useful for firewall evasion. We will not go through these in detail, however, they can be found here.

The following switches are of particular note:

-f:- Used to fragment the packets (i.e. split them into smaller pieces) making itless likely that the packets will be detected by a firewall or IDS.
An alternative to -f, but providing more control over the size of the packets: -- mtu <number>, accepts a maximum transmission unit size to use for the packets sent.This must be a multiple of 8.
--scan-delay <time>ms:- used to add a delay between packets sent. This is very useful if the network is unstable, but also for evading any time-based firewall/IDStriggers which may be in place.
--badsum:- this is used to generate in invalid checksum for packets. Any real TCP/IP stack would drop this packet, however, firewalls may potentially respond automatically, without bothering to check the checksum of the packet. As such, thisswitch can be used to determine the presence of a firewall/IDS.