Meow Write-up

Prepared by: 0ne-nine9

Setting Up

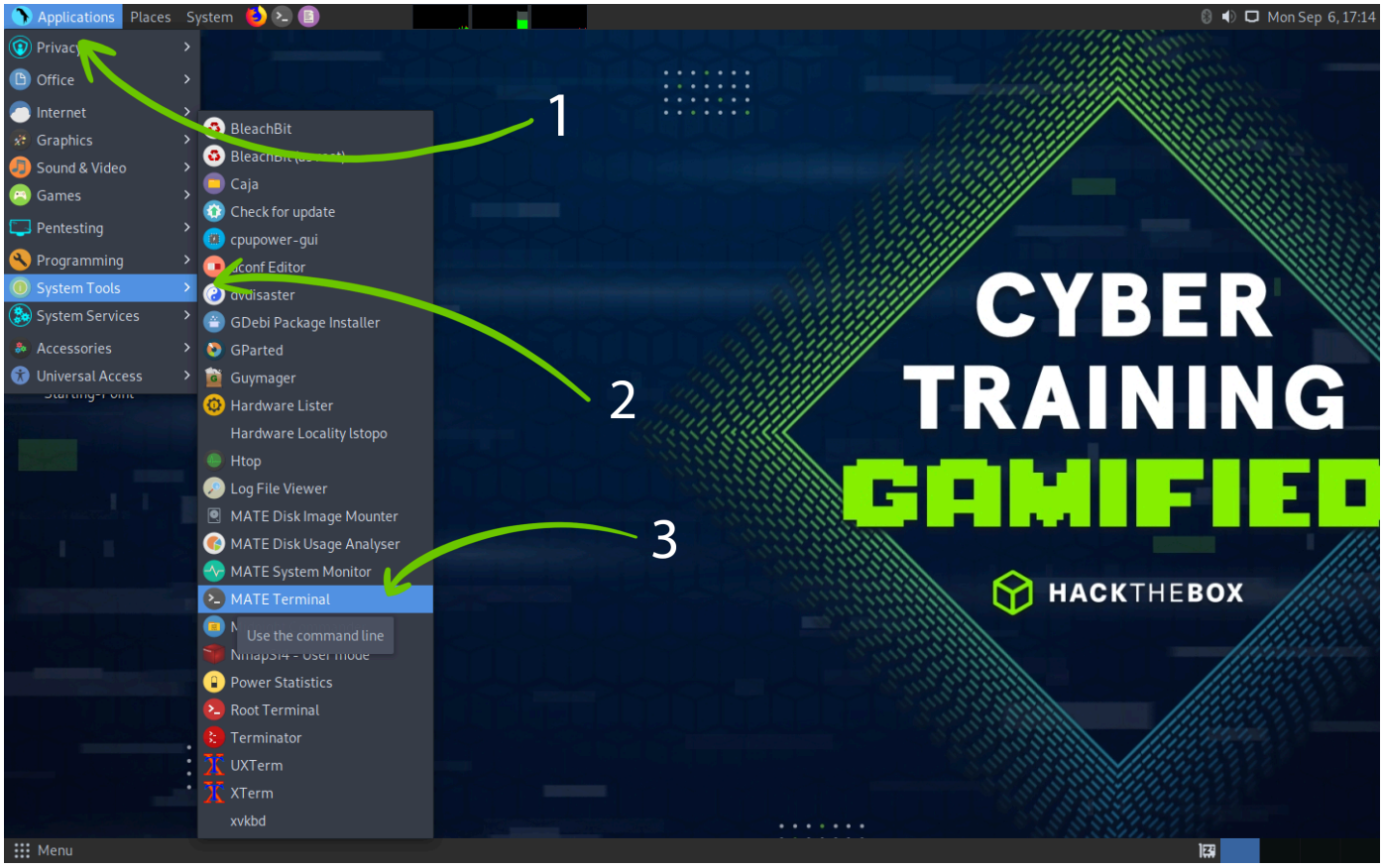
Welcome to Hack The Box!

Before we start with your very first vulnerable machine, let us make sure you are connected to the target's network and know your way around a terminal. When visiting the Starting Point lab's page, you might have been prompted to pick between a Pwnbox connection or a VPN configuration file that you can download and run on your Virtual Machine. If you have not learned how to set up a Virtual Machine yet, check out the Setting Up module on HTB Academy.

Running Pwnbox is straightforward, and you do not require any additional steps to connect to the target machine. If you boot up a new instance of Pwnbox under the Starting Point option, you will be automatically placed in the same network as the target. You can read more about Pwnbox in this article.

If you choose to download the VPN (.ovpn) configuration file on your Virtual Machine, then here is how to use it to connect to the target's network. In order to open a terminal window, you can click on the terminal icon on your Desktop.

Alternatively, you can navigate to the System Tools menu and select the terminal from there. In this case, we are using a MATE terminal. Ultimately, it does not matter what terminal you use as long as you do not get lost. Hovering over the terminal option, you can see the description of the tool: `Use the command line`, which is precisely what we will be doing next.



After selecting our terminal window, we will need to navigate where the .ovpn file was downloaded. In most cases, this is in the Downloads folder. In order to get there properly, let's start with a `cd` command, which stands for `change directory`, to make sure you're in your home folder for your current logged in user. Running this command without a specified location to navigate to will simply place you in your `home` directory. This way, we can make sure everyone reading this is at the right starting point (no pun intended). The next command we can type is `ls`, which will show us the home directory folders, some of them being `Desktop`, `Documents`, `Downloads` and more. `Downloads` is what we're interested in, and we use the `cd Downloads` command to navigate inside of it.

After navigating to the Downloads directory, type in `ls` to make sure the .ovpn file is present on the system, followed by the command to launch your OpenVPN client and connect to the Hack The Box internal network: `sudo openvpn {filename}.ovpn`, where {filename} should be replaced with the name of your .ovpn file for the Starting Point lab. The text marked in green and curly brackets `{}` is a replacement for your own version of input. This will be a recurring sight in the Starting Point write-ups, so keep that in mind!

After running the command, you will be prompted to input your super-user password, the same as the current password for your Operating System account. Don't worry if you can't see anything being typed into the terminal once you are typing your password. It's a security measure of Linux to stop other from shoulder-surfing you. Finish inputting your password and hit the Enter key once done to initialize the

openVPN connection.

Let the configuration script run until you see the `Initialization Sequence Completed` message at the very end of the output. Once that is present, make sure that there is no mention of multiple tunnel interfaces, such as `tun1`, `tun2`, and so forth. Having multiple tunnel interfaces can ruin the stability of your connection to the target and create routing conflicts on your Operating System, which would only bring frustration. There should only be `tun0` mentioned in the output as marked in the image below.

```
$ cd

$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

$ cd Downloads

$ ls
{filename}.ovpn

$ sudo openvpn {filename.ovpn}
[sudo] password for {username}: {your_VM_password}

2021-09-24 20:20:41 WARNING: Compression for receiving enabled. Compression has been used in the past to break encryption. Sent
packets are not compressed unless "allow-compression yes" is also set.
2021-09-24 20:20:41 DEPRECATED OPTION: --cipher set to 'AES-128-CBC' but missing in --data-ciphers (AES-256-GCM:AES-128-GCM).
Future OpenVPN version will ignore --cipher for cipher negotiations. Add 'AES-128-CBC' to --data-ciphers or change --cipher
'AES-128-CBC' to --data-ciphers-fallback 'AES-128-CBC' to silence this warning.
2021-09-24 20:20:41 OpenVPN 2.5.1 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on
May 14 2021


[..] Output omitted [..]

2021-09-24 20:20:41 net_addr_v6_add: {dead:beef:IPV6} dev tun0
2021-09-24 20:20:41 net_route_v4_add: 10.10.10.0/23 via 10.10.14.1 dev [NULL] table 0 metric -1
2021-09-24 20:20:41 net_route_v4_add: {IPV4} via 10.10.14.1 dev [NULL] table 0 metric -1
2021-09-24 20:20:41 add_route_ipv6(dead:beef::/64 -> {IPV6} metric -1) dev tun0
2021-09-24 20:20:41 net_route_v6_add: dead:beef::/64 via :: dev tun0 table 0 metric -1
2021-09-24 20:20:41 WARNING: this configuration may cache passwords in memory -- use the auth-nocache option to prevent this
2021-09-24 20:20:41 Initialization Sequence Completed
```

If you feel lost, you'll probably need to brush up on your Linux skills. In order to learn more about navigating and using Linux as a new user, you should check our our Linux Fundamentals module on HTB Academy. You'll be on the right track after completing it and you'll be able to return to Starting Point with a fresh set of skills that will tone down the frustration many new users feel when picking up pentesting for the first time with no prior contact with Linux, command line interfaces and big features.

After making sure everything in the output is in order, you can open a new terminal tab or window. Leave the current one running; otherwise, you will lose the connection to the target. You are now ready to start.

Introduction

When first starting a penetration test or any security evaluation on a target, a primary step is known as `Enumeration`. This step consists of documenting the current state of the target to learn as much as possible about it.

Since you are now on the same Virtual Private Network (VPN) as the target, you can directly access it as any user would. If the target is a web server, running a public web page, you can navigate to its IP address to see what the page contains. If the target is a storage server, you can connect to it using the same IP address to explore the files and folders stored on it, provided that you have the necessary credentials. The question is, how do you find these services? You cannot manually search for them because it would take a long time.

Every server uses `ports` in order to serve data to other clients. The first steps in the Enumeration phase involve scanning these open ports to see the purpose of the target on the network and what potential vulnerabilities might appear from the services running on it. In order to quickly scan for ports, we can use a tool called `nmap`, which we will detail more in the Enumeration chapter of this write-up.

After finding the open ports on the target, we can manually access each of them using different tools to find out if we have access to their contents or not. Different services will use different tools or scripts to be accessed. These can be discovered and learned by a beginner penetration tester only with time and practice (and some diligent Googling). 90% of penetration testing consists of research done on the internet about the product you are testing. Since the technological ecosystem is continuously evolving, it is impossible to know everything about everything. The key is to know how to look for the information you need. The ability to

research effectively is the skill you need to continuously adapt and evolve into your top quality.

The objective here is not speed but meticulousness. If a resource on the target is missed during the Enumeration phase of your test, you might lose a vital attack vector which would have potentially cut your worktime on the target in half or even less.

Enumeration

After our VPN connection is successfully established, we can ping the target's IP address to see if our packets reach their destination. You can take the IP address of your current target from the Starting Point lab's page and paste it into your terminal after typing in the `ping` command as illustrated below.

```
$ ping {target_IP}

PING {target_IP} ({target_IP}) 56(84) bytes of data.
64 bytes from {target_IP}: icmp_seq=1 ttl=63 time=20.4 ms
64 bytes from {target_IP}: icmp_seq=2 ttl=63 time=22.0 ms
64 bytes from {target_IP}: icmp_seq=3 ttl=63 time=20.2 ms
64 bytes from {target_IP}: icmp_seq=4 ttl=63 time=19.8 ms
^C
--- {target_IP} ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 19.788/20.603/22.026/0.849 ms
```

After four successful replies from the target, we can determine that our connection is formed and stable. We can cancel the ping command by pressing the `CTRL+C` combination on our keyboard, which will be displayed in the terminal as `^C` marked above in green. This will return control of the terminal tab to us, from where we can proceed with the next step - scanning all of the target's open ports to determine the services running on it. In order to start the scanning process, we can use the following command with the `nmap` script. `nmap` stands for Network Mapper, and it will send requests to the target's ports in hopes of receiving a reply, thus determining if the said port is open or not. Some ports are used by default by certain services. Others might be non-standard, which is why we will be using the service detection flag `-sV` to determine the name and description of the identified services. The text marked in green and curly brackets `{}` is a replacement for your own version of input. In this case, you will need to replace the `{target_IP}` part with the IP address of your own target.

```
$ sudo nmap -sV {target_IP}

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-24 20:36 BST
Nmap scan report for {target_IP}
Host is up (0.050s latency).
Not shown: 999 closed tcp ports (reset)

PORT   STATE SERVICE VERSION
23/tcp open  telnet  Linux telnetd
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 1.82 seconds
```

Following the completion of the scan, we have identified `port 23/tcp` in an `open` state, running the `telnet` service. Following a quick Google search of this protocol, we find out that telnet is an old service used for remote management of other hosts on the network. Since the target is running this service, it can receive telnet connection requests from other hosts in the network (such as ourselves). Usually, connection requests through telnet are configured with username/password combinations for increased security. We can see this is the case for our target, as we are met with a Hack The Box banner and a request from the target to authenticate ourselves before being allowed to proceed with remote management of the target host.

```
$ telnet {target_IP}

Trying {target_IP}...
Connected to {target_IP}.
Escape character is '^]'.

 Hack The Box

Meow login:
```

We will need to find some credentials that work to continue since there are no other ports open on the target that we could explore.

Foothold

Sometimes, due to configuration mistakes, some important accounts can be left with blank passwords for the sake of accessibility. This is a significant issue with some network devices or hosts, leaving them open to simple brute-forcing attacks, where the attacker can try logging in sequentially, using a list of usernames with no password input.

Some typical important accounts have self-explanatory names, such as:

- admin
- administrator
- root

A direct way to attempt logging in with these credentials in hopes that one of them exists and has a blank password is to input them manually in the terminal when the hosts request them. If the list were longer, we could use a script to automate this process, feeding it a wordlist for usernames and one for passwords. Typically, the wordlists used for this task consist of typical people names, abbreviations, or data from previous database leaks. For now, we can resort to manually trying these three main usernames above.

The first two were not so lucky for us. When things look down, it is essential to keep going, be persistent. We can't succeed unless we attempt all possibilities. Let us try the last one.

```
Meow login: root
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon 06 Sep 2021 03:15:22 PM UTC

  System load:  0.0               Processes:             195
  Usage of /:   41.7% of 7.75GB   Users logged in:       0
  Memory usage: 4%                IPv4 address for eth0: {target_IP}
  Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

72 updates can be applied immediately.
29 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Wed Jul  7 10:55:01 UTC 2021 on tty1
```

Success! We have logged into the target system. We can now go ahead and take a look around the directory we landed in using the `ls` command. There is a possibility we might find what we are looking for.

```
# ls
flag.txt  snap

# cat flag.txt
b40abdfe23665f766f9c61ecba8a4c19
```

The `flag.txt` file is our target in this case. Most of Hack The Box's targets will have one of these files, which will contain a hash value called `a flag`. The naming convention for these targeted files varies from lab to lab. For example, weekly and retired machines will have two flags, namely `user.txt` and `root.txt`. CTF targets and other labs will have `flag.txt`. Challenges will, most of the time, not contain an actual file, but rather offer you snippets of the flag as you solve it, the respective parts being embedded into the challenge more homogeneously (text hidden in an image, or other examples).

You can read the file to have the hash value displayed in the terminal using the `cat` command. Copying the flag and pasting it into the Starting Point lab's page will grant you ownership of this machine, completing your very first task.

Congratulations!

Fawn Write-up

Prepared by: 0ne-nine9

Introduction

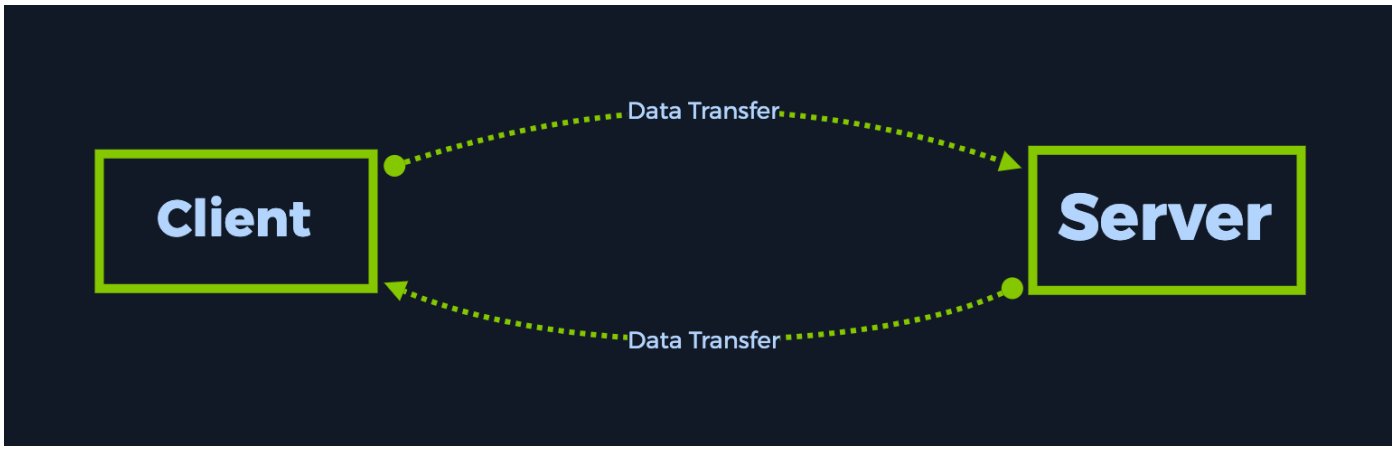
Sometimes, when we are asked to enumerate the services of specific hosts on the client network, we will be met with file transfer services that may have high chances to be poorly configured. The purpose of this exercise is to familiarize yourself with the File Transfer Protocol (FTP), a native protocol to all host operating systems and used for a long time for simple file transfer tasks, be they automated or manual. FTP can be easily misconfigured if not correctly understood. There are cases where an employee of the client company we are assessing might want to bypass file checks or firewall rules for transferring a file from themselves to their peers. Considering the many different mechanisms for controlling and monitoring data flow within an enterprise network today, this scenario becomes a substantial and viable case we might meet in the wild.

At the same time, FTP can be used to transfer log files from one network device to another or a log collection server. Suppose the network engineer in charge of handling the configuration forgets to secure the receiving FTP server properly or does not put enough importance on the information contained within the logs and decides to leave the FTP service unsecured intentionally. In that case, an attacker could gain leverage of the logs and extract all kinds of information from them, which can later be used to map out the network, enumerate usernames, detect active services, and more.

Let's take a look at what FTP is, according to definition on Wikipedia:

```
The File Transfer Protocol (FTP) is a standard communication protocol used to transfer
computer files from a server to a client on a computer network. FTP is built on a
client–server model architecture using separate control and data connections between
the client and the server. FTP users may authenticate themselves with a clear-text
sign-in protocol, generally in the form of a username and password. However, they can
connect anonymously if the server is configured to allow it. For secure transmission
that protects the username and password and encrypts the content, FTP is often secured
with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).
```

From the first lines of the excerpt above, we can see mention of the client-server model architecture. This refers to the roles hosts in the network have during the act of transferring data between them. Users can download and upload files from the client (their own host) to the server (a centralized data storage device) or vice versa. Conceptually speaking, the client is always the host that downloads and uploads files to the server, and the server always is the host that safely stores the data being transferred.



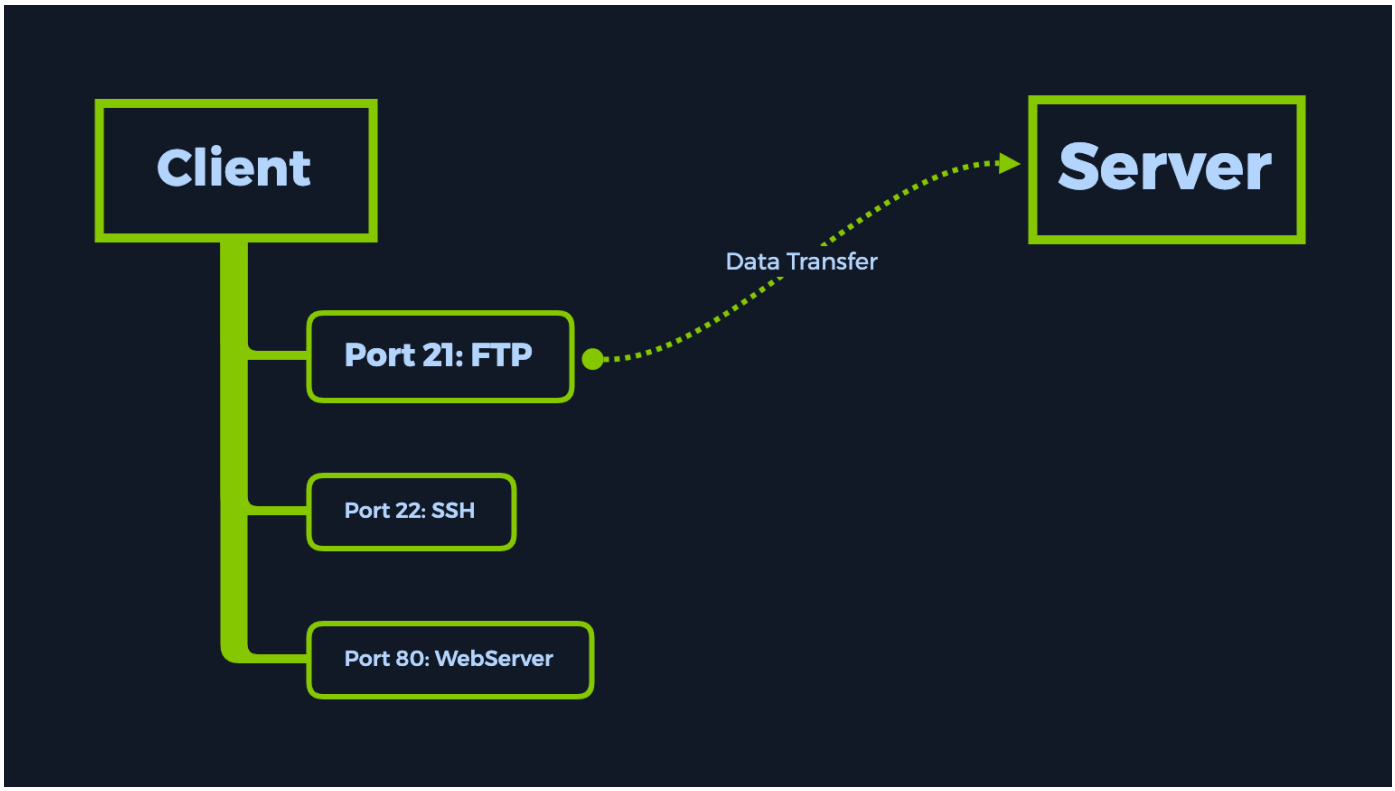
Clients can also browse the available files on the server when using the FTP protocol. From a user's terminal perspective, this action will seem like browsing their own operating system's directories for files that they need. FTP services also come with a GUI (Graphical User Interface), akin to Windows OS Programs, enabling easier navigation for beginners. An example of a well-known GUI-oriented FTP Service is FileZilla. However, let's first understand what it means for a port to be running a service openly.

A port running an active service is a reserved space for the IP address of the target to receive requests and send results from. If we only had IP addresses or hostnames, then the hosts could only do 1 task at a time. This means that if you wanted to browse the web and play music from an application on your computer simultaneously, you could not, because the IP address would be used for handling either the first or the latter, but not both at the same time. By having ports, you can have one IP address handling multiple services, as it adds another layer of distinction.

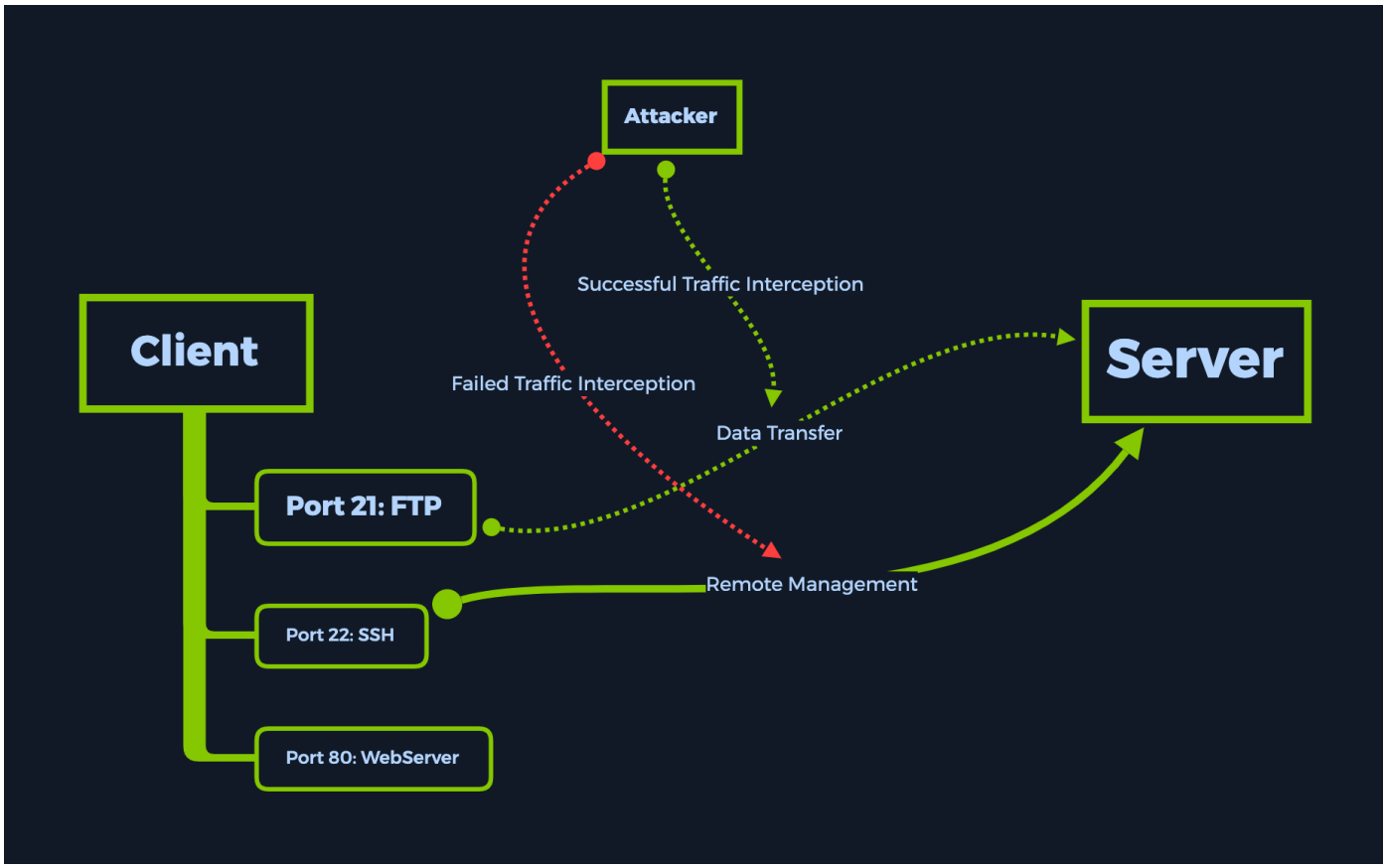
In the case shown below, we can see FTP being active on port 21. However, let's add some extra services like SSH (Secure Shell Protocol) and HTTPD (Web Server) in order to explore a more typical example. With this type of configuration, a network administrator has set up a rudimentary core web server configuration, allowing them to achieve the following, all at the same time if need be:

- Receive and send files that can be used to configure the webserver or serve logs to an external source
- Be able to be logged into for remote management from a distant host, in case any configuration changes are needed
- Serve web content that can be accessed remotely through another host's web browser

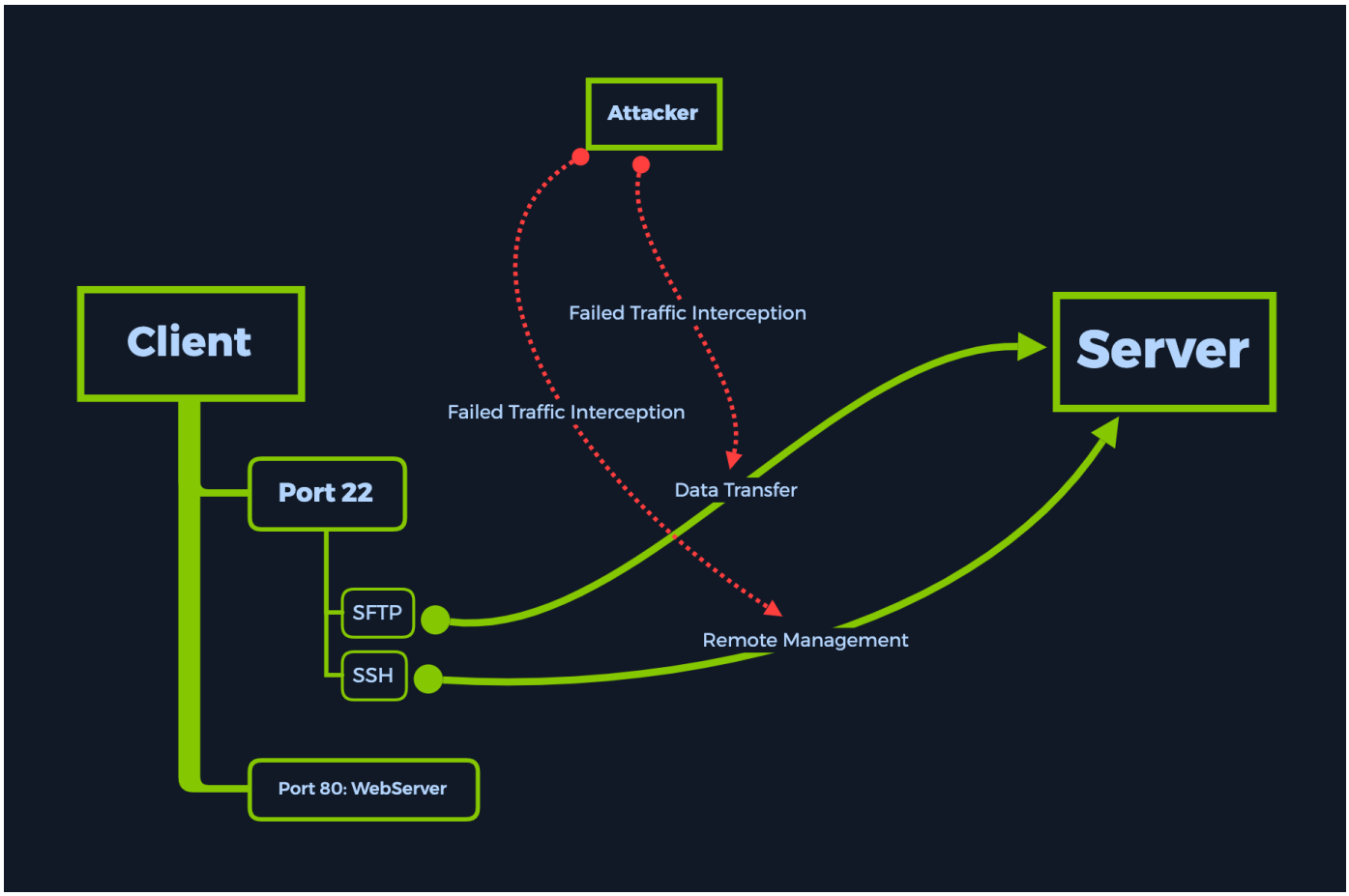
From the graph below, you can see where FTP sits in the logical structure of the host, together with other services that could potentially be running on it at the same time.



The Wiki article shows that it is considered non-standard for FTP to be used without the encryption layer provided by protocols such as SSL/TLS (FTPS) or SSH-tunneling (SFTP). FTP by itself does have the ability to require credentials before allowing access to the stored files. However, the deficiency here is that traffic containing said files can be intercepted with what is known as a Man-in-the-Middle Attack (MitM). The contents of the files can be read in plaintext (meaning unencrypted, human-readable form).



However, if the network administrators choose to wrap the connection with the SSL/TLS protocol or tunnel the FTP connection through SSH (as shown below) to add a layer of encryption that only the source and destination hosts can decrypt, this would successfully foil most Man-in-the-Middle attacks. Notice how port 21 has disappeared, as the FTP protocol gets moved under the SSH protocol on port 22, thus being tunneled through it and secured against any interception.



However, the situation we are dealing with in this case is much simpler. We are only going to interact with the target running a simple, misconfigured FTP service. Let us proceed and analyze how such a service running on an internal host would look like.

Enumeration

Firstly, let us check if our VPN connection is established. Using the ping protocol can help with this since it is a low-overhead method of reaching the target to get a response, thus confirming our connection is established, and the target is reachable. Low-overhead means that very little data is sent to the target by default, allowing us to quickly check the status of the connection without having to wait for a whole scan to complete beforehand. The ping protocol can be invoked from the terminal using the `ping {target_IP}` command, where {target_IP} is the IP address of your instance of the Fawn machine, as displayed on the Hack The Box webpage, under the Starting Point lab.

Note that this might not always work in a large-scale corporate environment, as firewalls usually have rules to prevent pinging between hosts, even in the same subnet (LAN), to avoid insider threats and discover other hosts and services.

```
$ ping {target_IP}

PING {target_IP} ({target_IP}) 56(84) bytes of data.
64 bytes from {target_IP}: icmp_seq=1 ttl=63 time=49.2 ms
64 bytes from {target_IP}: icmp_seq=2 ttl=63 time=47.1 ms
64 bytes from {target_IP}: icmp_seq=3 ttl=63 time=60.6 ms
64 bytes from {target_IP}: icmp_seq=4 ttl=63 time=41.0 ms
^C
--- {target_IP} ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 41.020/49.486/60.572/7.076 ms
```

We can cancel the `ping` command by pressing `CTRL+C` on our keyboard, otherwise it will run infinitely. Following the output from the command, we can see that responses are being received from the target host. This means that the host is reachable through the VPN tunnel we formed. We can now start scanning the open services on the host.

```
$ sudo nmap {target_IP}

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-24 22:30 BST
Nmap scan report for {target_IP}
Host is up (0.048s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp

Nmap done: 1 IP address (1 host up) scanned in 1.10 seconds
```

Scanning using our previously used command, we can see the FTP service open and running on port 21. However, what if we would like to know the actual version of the service running on this port? Could scanning it with different switches present us with the needed information?

```
$ sudo nmap -sV {target_IP}

Starting Nmap 7.92 ( https://nmap.org ) at 2021-09-24 22:31 BST
Nmap scan report for {target_IP}
Host is up (0.050s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
Service Info: OS: Unix

Service detection performed. Please report any incorrect results
at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 2.28 seconds
```

In our case, the `-sV` switch stands for version detection. Using this switch will consequently make our scan take longer but will offer us more insight into the version of the service running on the previously detected port. This means that at a glance, we would be able to tell if the target is vulnerable due to running outdated software or if we need to dig deeper to find our attack vector.

We will not be looking at exploiting the service per sé. We will take small steps towards our goals, and the next one will involve simply interacting with the service as-is to learn more about how we should approach targets. However, having the service version always helps us gain more insight into what is running on the scanned port.

Foothold

It is time we interacted with the target.
In order to access the FTP service, we will use the `ftp` command on our own host. It's good practice to have a quick check that your `ftp` is up to date and installed properly. Running the command below will display the same output as pictured if your `ftp` service is installed. Otherwise, it will continue with the installation. The `-y` switch at the end of the command is used to accept the installation without interrupting the process to ask you if you'd like to proceed.

```
$ sudo apt install ftp -y

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ftp is already the newest version (0.17-34.1.1).
```

After it has done installing, you can run the `ftp -h` command to see what the service is capable of.

```
$ ftp -h

    Usage: { ftp | pftp } [-46pinegvtd] [hostname]
       -4: use IPv4 addresses only
       -6: use IPv6, nothing else
       -p: enable passive mode (default for pftp)
       -i: turn off prompting during mget
       -n: inhibit auto-login
       -e: disable readline support, if present
       -g: disable filename globbing
       -v: verbose mode
       -t: enable packet tracing [nonfunctional]
       -d: enable debugging
```

From the excerpt above, we can see that we can connect to the target host using the command below. This will initiate a request to authenticate on the FTP service running on the target, which will return a prompt back to our host:

```
$ ftp {target_IP}
Connected to {target_IP}.
220 (vsFTPd 3.0.3)
Name ({target_IP}:{username}):
```

The prompt will ask us for the username we want to log in with. Here is where the magic happens.

A typical misconfiguration for running FTP services allows an `anonymous` account to access the service like any other authenticated user. The `anonymous` username can be input when the prompt appears, followed by any password whatsoever since the service will disregard the password for this specific account.

```
$ ftp {target_IP}
Connected to {target_IP}.
220 (vsFTPd 3.0.3)
Name ({target_IP}:{username}): anonymous
331 Please specify the password.
Password: anon123
```

Hitting `Enter` after filling in the password, we can see that we are logged in successfully. Our terminal changes in order to show us that we can now issue `ftp` commands.

```
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Typing in the `help` command allows us to view which commands are available. You will be able to see this pattern with every script and service that you have access to. Typing either the `-h`, `--help`, or `help` commands will always issue a list of all the commands available to you as a user, with descriptions occasionally included. If you would like to learn about a specific command in more depth, you can use a different command: `man {commandName}`. However, for now, let us get back to our target.

```
ftp> help
Commands may be abbreviated. Commands are:

!           dir         mdelete     qc          site
$           disconnect  mdir        sendport    size
account     exit        mget        put         status
append      form        mkdir       pwd         struct
ascii       get         mls         quit        system
bell        glob        mode        quote       sunique
binary      hash        modtime     recv        tenex
bye         help        mput        reget       tick
case        idle        newer       rstatus     trace
cd          image       nmap        rhelp       type
cdup        ipany       nlist       rename      user
chmod       ipv4        ntrans      reset       umask
close       ipv6        open        restart     verbose
cr          lcd         prompt      rmdir       ?
delete      ls          passive     runique
debug       macdef      proxy       send
ftp>
```

Some of the commands listed here seem familiar to us. We already know how to use `ls` and `cd`. Let us issue the first command and view the contents of the folder.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0              32 Jun 04 03:25 flag.txt
226 Directory send OK.
ftp>
```

As you can notice from the output, the operation of FTP services also issue the status for the commands you are sending to the remote host. The meaning of status updates are as follows:

```
200 : PORT command successful. Consider using PASV.
150 : Here comes the directory listing.
226 : Directory send OK.
```

Now, we can proceed to download the `flag.txt` to our host (Virtual Machine). In order to do so, we can use the `get` command, followed by the name of the file we want to download. In our case, it would look like this:

```
ftp> get flag.txt
local: flag.txt remote: flag.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for flag.txt (32 bytes).
226 Transfer complete.
32 bytes received in 0.00 secs (33.7838 kB/s)
ftp>
```

This will trigger the download of the file to the same directory you were in when you issued the `ftp {machineIP}` command. If we exit the FTP service, we will see the same file on our host now.

```
ftp> bye
421 Timeout.

$ ls
flag.txt  Starting-Point

$ cat flag.txt
035db21c881520061c53e0536e44f815
```

We can now take the flag and submit it on the platform in order to own the box!

Nice work!