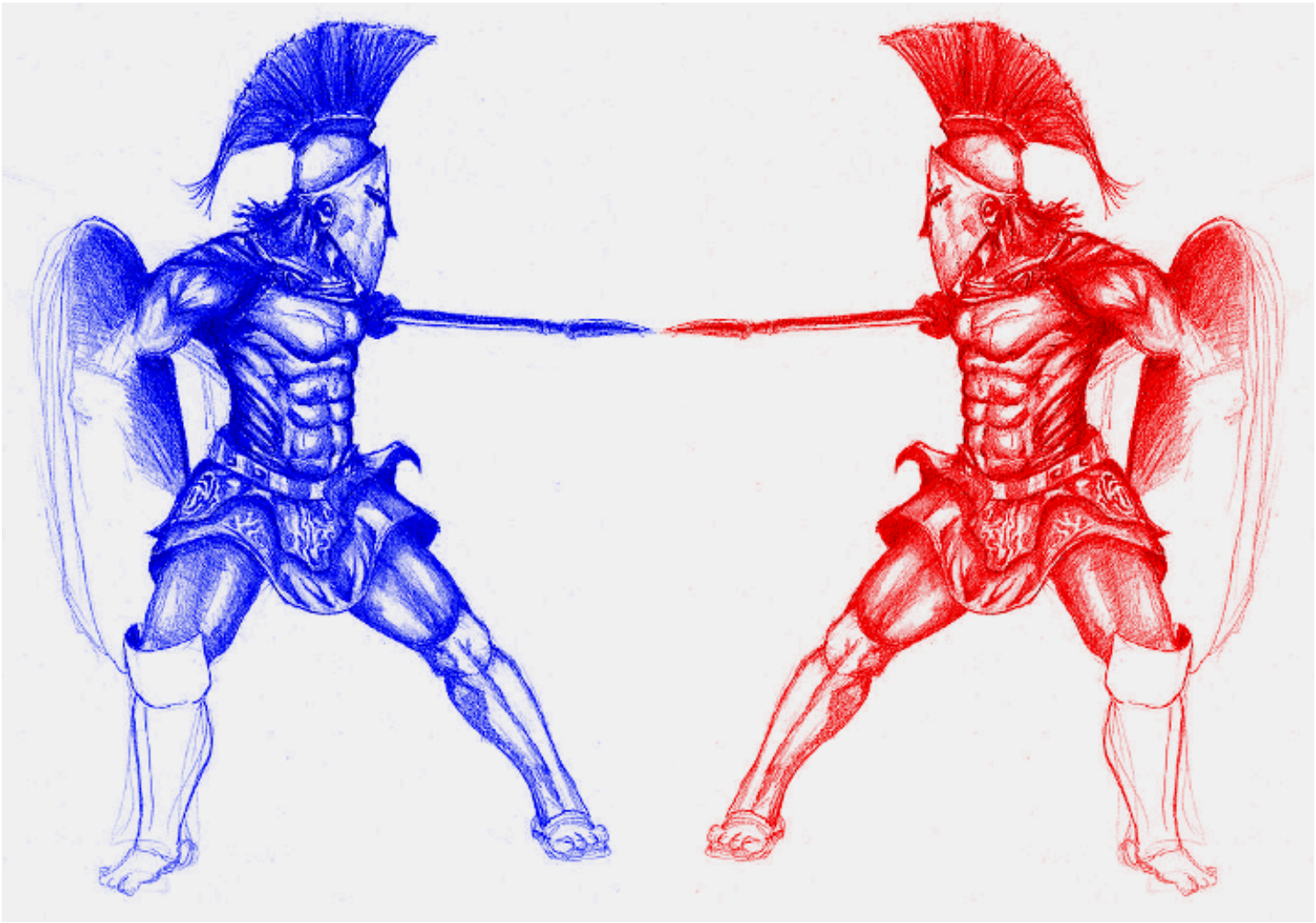
Blue Team Crash Course



This repo contains a growing tutorial/guide for college-level blue teaming.

- Chapter 0 - What's Blue Teaming? - What's this all about?
- Chapter 1 - Getting Started - Using Virtualbox and installing Linux.
- Chapter 2 - Linux Basics - Using Linux.
- Chapter 3 - Networking Basics - Some networking basics.
- Chapter 4 - More Networking - More networking stuff.
- Chapter 5 - Being an Admin - Some basics of managing your new server.
- Chapter 6 - DNS - The domain name system protocol and configuring your first service.

Credits

layout: default title: Chapter 0 - What's Blue Teaming?

Assumptions

You're a college student looking at completing in a blue team competition.

What You'll Learn

- What's Blue Teaming?
- What's the goal of all this?

What's Blue Teaming?

Overall, being on the "blue" team is to defend a computer network from a "red" team that performs attacks on the network similar to how real-world adversaries might attack real systems. As a blue teamer, you have to try stop the hackers on red team from hacking your stuff, manage a network, configure systems, and learn stuff, all at the same time.

Yeah, that's a lot, but its all to give you a rapid-fire, action-packed, and short-term taste of real-world defensive cybersecurity activities. Real-life defending may not usually happen as fast as what happens at a competition (real hackers act slowly so as not raise the alarm), but it'll give you an intense boot-camp of what to do while working alongside your peers. This can give you new perspectives on computer security, and is a great experience, even if you don't end up defending networks yourself. It could give you ideas to program new defensive tools, communications skills for teams in high-pressure environments, or even help learning the other side, the red team (like me!).

So What am I Learning?

Blue team competitions give you a taste of a few different cybersecurity jobs, here's some of them:

- Incident Response - Incident response is what happens when something malicious has happened on a network, and a incident response team is called in to identify the issue and help fix the vulnerabilities. This is basically what happens when you first arrive at the event. The red team is usually given an initial advantage, with pre-placed vulnerabilities and backdoors that give them easy access to your stuff, and your job is to clean up the mess as best and as fast as you can. For more info: https://www.forcepoint.com/cyber-edu/incident-response

- Systems Administration - This is a big chunk of the competition, you have a bunch of systems, and you need to keep them running. You might be even asked to add new systems and services to the network. System administrators (or sysadmins) do this all the time in the real-world, configuring and maintaining systems while ensuring they stay up and available. Sysadmins must be careful though, because changes like as patches and security configurations could cause problems, making applications and systems to go down! It's a delicate balance.
- Threat Hunting - This is pretty self explanatory, you're looking for the bad guys on your network. This kind goes along with system administration, as it's finding the abnormal stuff, the suspicious activities and connections that might lead you to red team actions. Although the real-world might have tools to help you look for bad stuff, your best tools are a sharp eye, good investigation skills, and knowledge of the system and services running. You'll follow clues to see if something is bad or harmless. Again, drastic and rapid responses may do more harm than good, so you must be careful what you do.

So What Should I Do?

I wrote an article that helps with this here

What Should I Keep in Mind?

- Don't give up! You won't learn anything that way.
- Blue team involves a lot of awareness. Be aware of what's on your systems, what's running, and what's going on. Find tools that can help you do this.
- Ask questions of the red team (usually after the event...). Ask what they did and what you could do better. They may now give you the details (a magician never gives away his secrets), but you'll get an idea of what to prepare for next time.
- Back up your stuff. Make sure you have something to fall back to if you accidentally mess stuff up (it happens!) or red team messes something up (we'll do that!).
- Be kind to your event coordinators and event staff (usually called white team and/or black team). They're doing a lot to make the event happen. Be mean to them and you might have red team taking vengeance upon you and your team.

What's the Red Team Doing?

The goal of the red team is to emulate what hackers in the real-world might do to your services. Hackers are sneaky and stealthy, so the red team uses clever tactics to hide their tracks and

activities. Hackers might use multiple paths for attacks, so the red team doesn't just use a single method to attack blue team systems.

We're here to help you learn, so be sure to communicate with them to make the most of your experience.

Who am I?

My name is Jacob Hartman. I've been in the professional cybersecurity field for 3+ years, with 3 years in collegiate cybersecurity competitions while studying in college. Now I'm usually on the red team for CNY Hackathon and help out with the cybersecurity club at SUNY Polytechic Institute. The content here is based on my experiences on both sides of this type of competition.

	Navigation	
	Home	Ch 1 >

{% include footer.html %}

layout: default title: Chapter 1 - Getting Started

Assumptions

This page assumes:

- You're on Windows. (If you're on Linux, install VirtualBox with your package manager and install the server, we'll need it later)

What are Virtual Machines

Virtual machines are essentially virtual systems with real operating systems running inside another system. VirtualBox is a *hypervisor* which allows us to do this. Virtual Machines (or VMs) are everywhere in cybersecurity and IT these days, so it's something you'll really want to get familiar with. We'll be using VMs to create a small number of interconnected VMs that can talk with each other.

Installing VirtualBox.

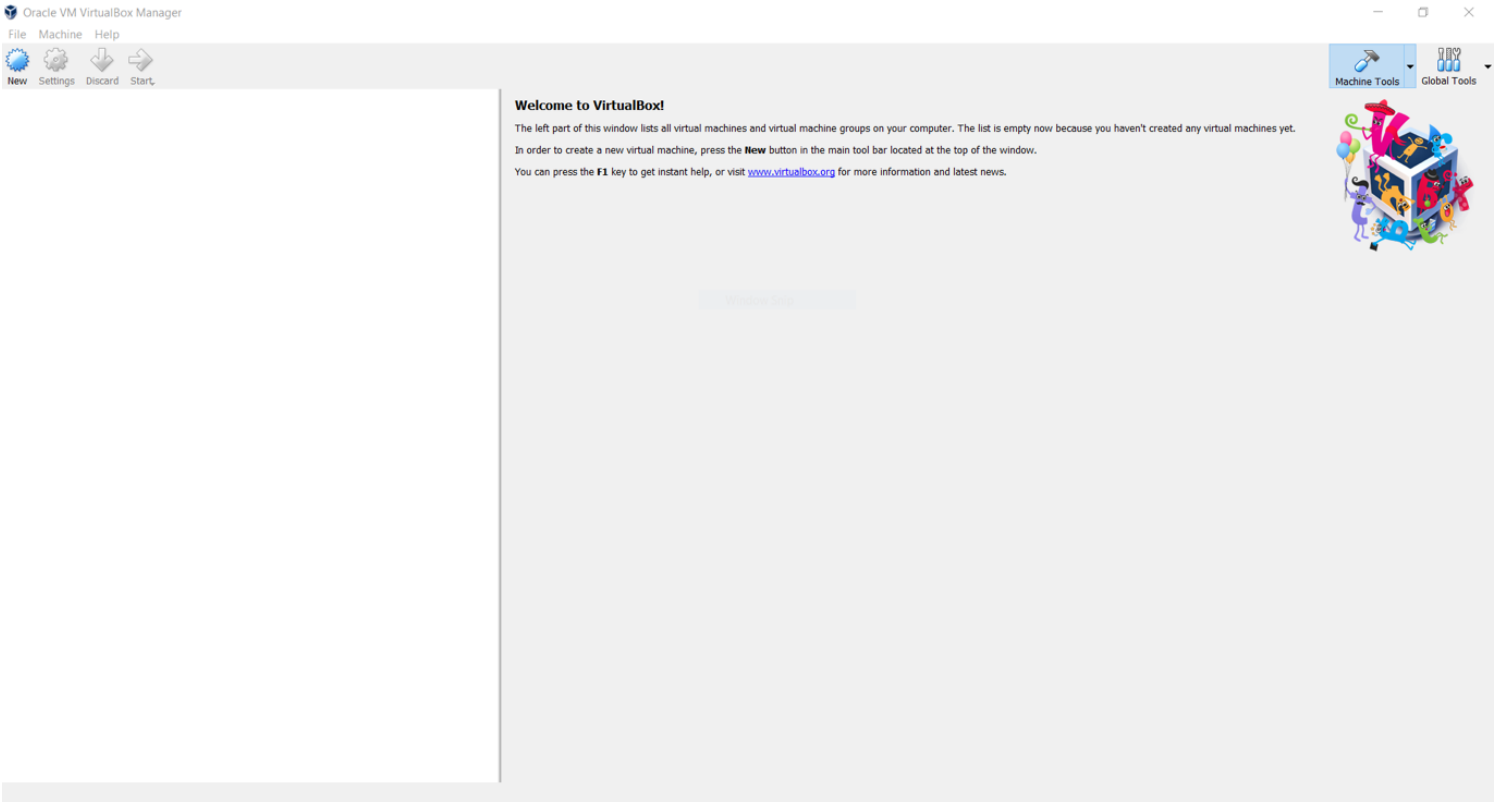
The wizard is pretty straight forward, just take the defaults and be sure to accept the installation of the drivers.

Getting an ISO

The ISO file is used as a virtual CD disk with your virtual machine. We'll be using **Ubuntu 18.04**, which you can get here: http://releases.ubuntu.com/18.04/ubuntu-18.04.1-live-server-amd64.iso

Create a virtual machine

Open VirtualBox. You should see something like this:



On the the main interface on the left, will be a list of virtual machines (VMs) once we create one. The right panel will give details on a VM when one is selected on the left.

To create a VM select the "New" button on the top left.

1. The wizard will prompt you for a name and operating system type. Make the name whatever you want, but set the Type to "Linux" and keep the default "Ubuntu (64-bit)".

2. On the next screen, keep it at the default 1024 MB.

3. On the next screen, keep the default (Create a virtual hard disk now) and just press Create.
4. On the Hard disk file type screen, keep the default VDA setting.
5. On the next screen, keep the default "Dynamically allocated"
6. On the next screen, set the size to 20 GB, then click "Create"

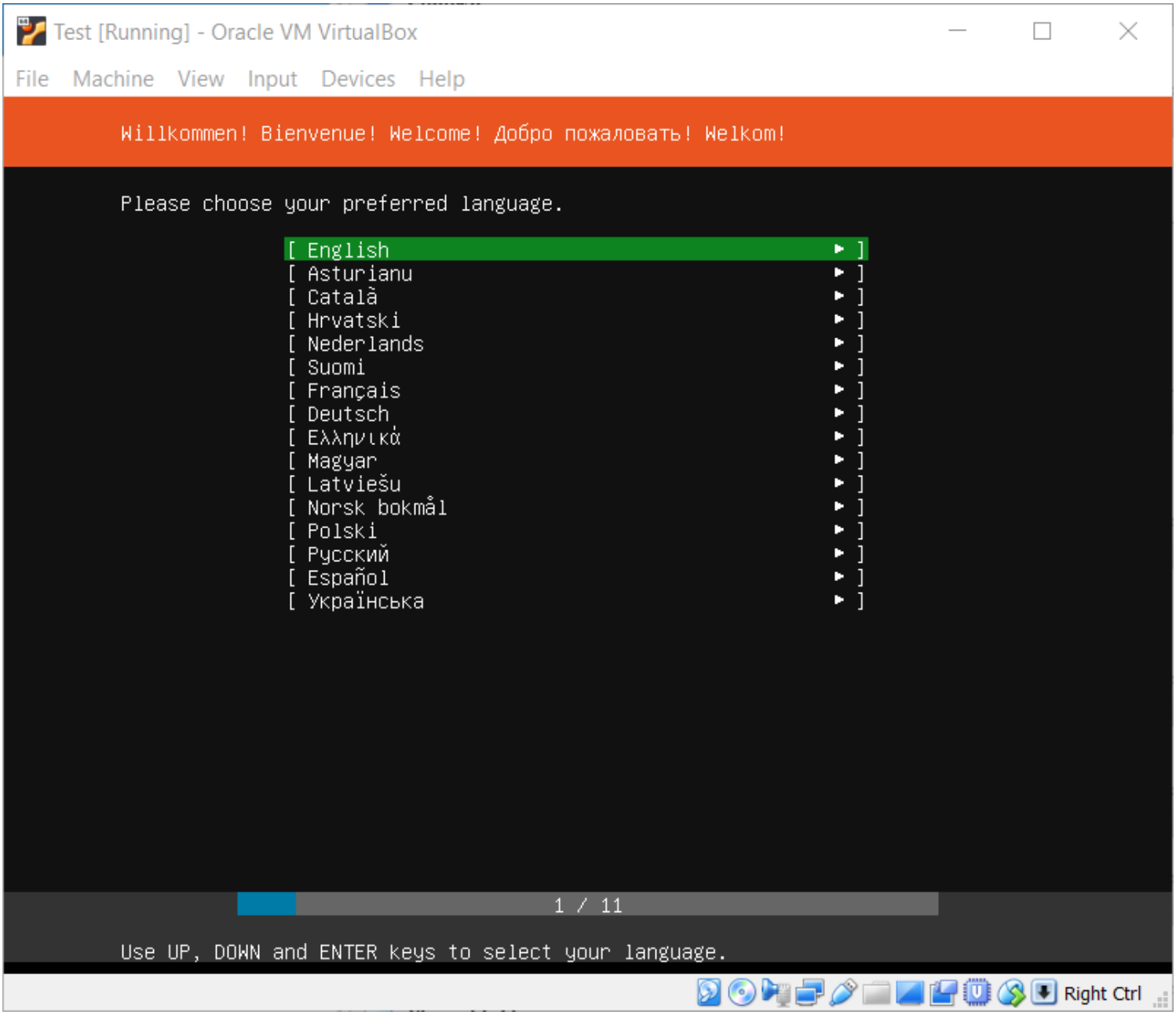
Now the VM is created, select it on the left and click the "Details" button on the right. These are your VMs details.

To start the VM, press the "Start" button. A window should appear. A prompt will pup up asking for a start-up disk. The virtual hard disk we created has no OS, so we must install it ourselves.

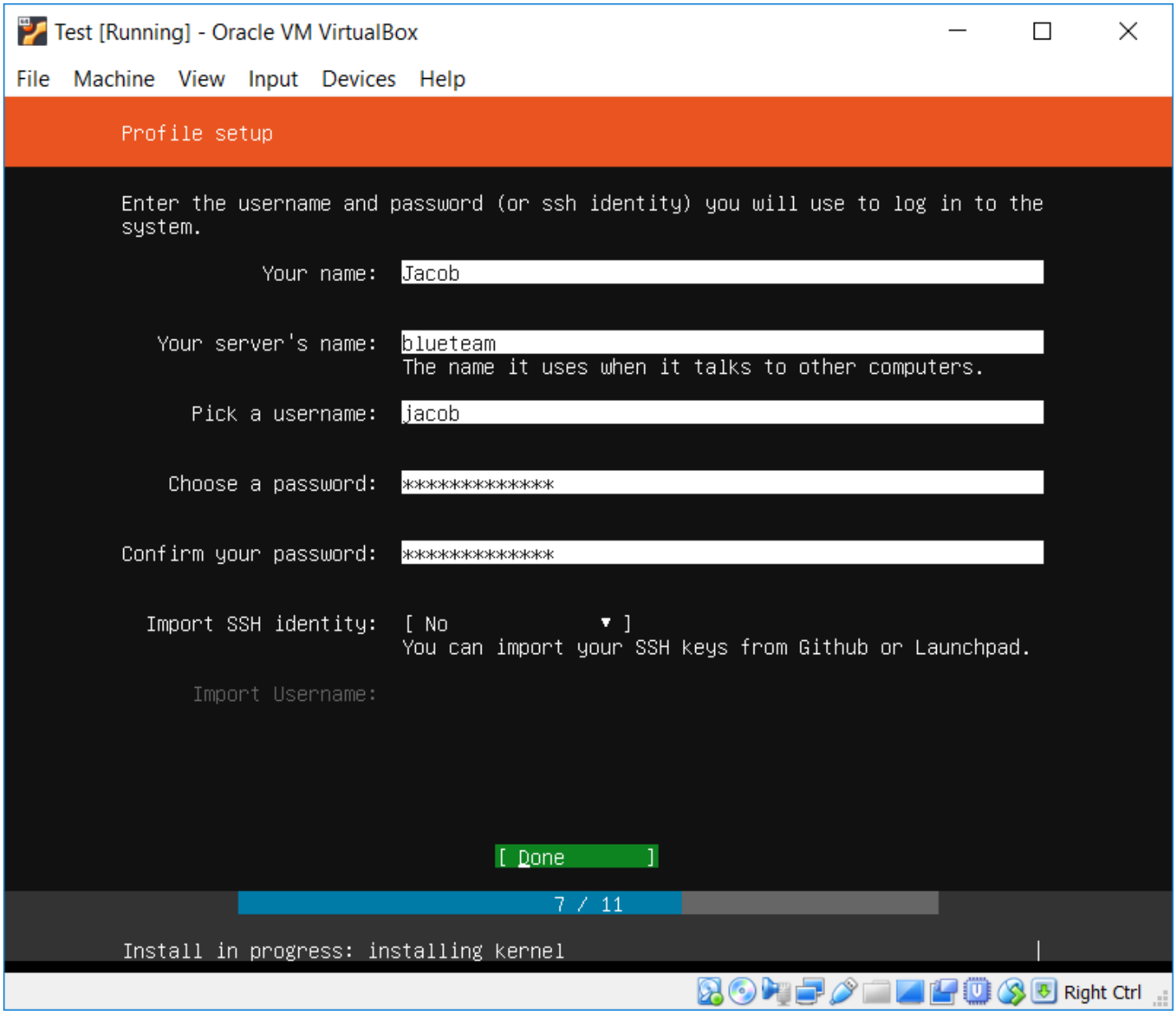
Click the folder icon next to the dropdown and browser to where the ISO is located, select it and press "Open." Then press start.

Installing Ubuntu Server

After some text scrolls down the screen, you should eventually come to a prompt to choose your language. Welcome to the Ubuntu install wizard!



1. English is the default language. We'll be doing this tutorial in English, so please press Enter.
2. Set your keyboard. Just press Enter.
3. On the next screen, select "Install Ubuntu" with the arrow keys. When selected, press enter.
4. On the next screen, press Enter
5. On the screen for proxu info, press enter.
6. On the mirror address screen, press enter.
7. On the next screen, select "Use An Entire Disk" using the arrow keys. Then press enter.
8. On the next screen, the disk we created earlier is the default, press enter.
9. On the next screen, press enter to select Done.
10. A warning prompt will appear, ask us to "Confirm destructive action." With the arrow keys, press down to select "Continue." Press enter
11. On the next screen, you'll set up your username and password. Use the arrow keys to move between text boxes. When done, press down until Done is selected, then press Enter.



12. On the next screen with the list of "snaps," press Tab to drop down to the "Done" button. Press enter.
13. On the "Finished install!" screen, select "Reboot now." The VM will reboot. When prompted to remove installation media, just press enter VirtualBox automatically takes care of this for us.

Using Ubuntu

The reboot is complete and the system is running, when only a login prompt is present. If stuff about SSH keys pop up, ignore it. Press enter to get a new prompt.

```
Ubuntu 18.04.1 LTS blueteam tty1

blueteam login: _
```

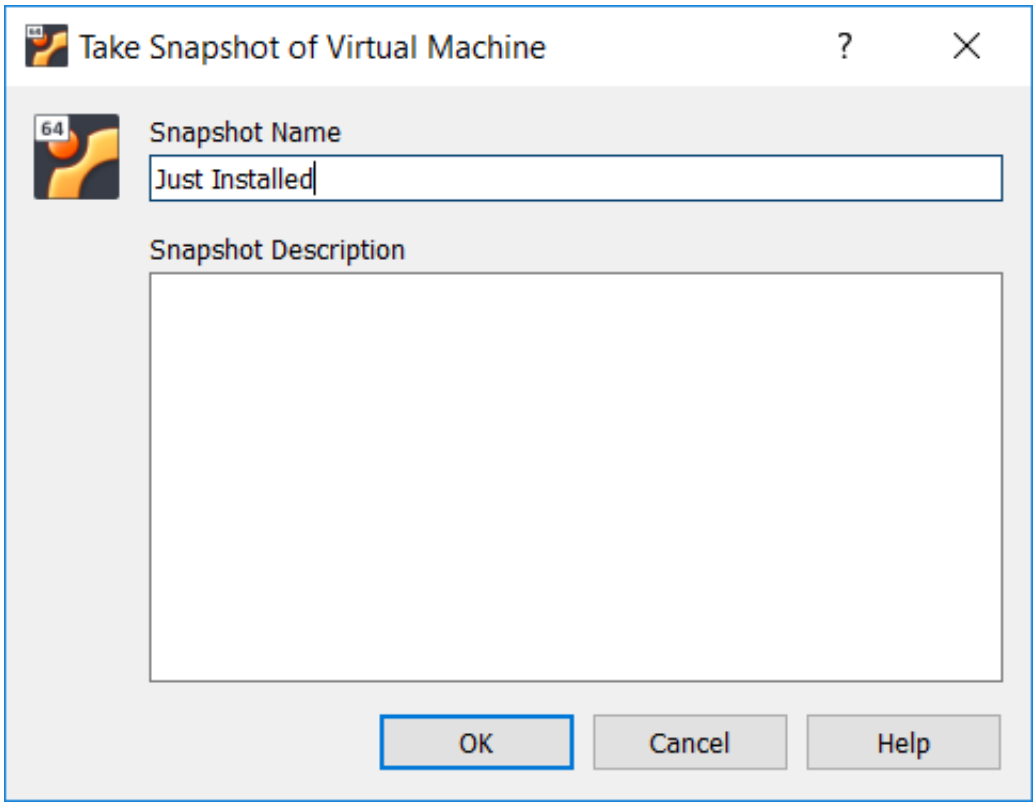
Login using the username and password you set before. Congrats! Ubuntu server is installed. Ubuntu server does NOT have a GUI like Windows. It's meant to be a server accessed remotely. We're mainly using it because part of this guide is to learn to use one of the most powerful parts of Linux, the command line.

Snapshots

Taking a Snapshot

VirtualBox supports snapshots, which stores the memory and disk at a certain point in time. It's essentially a back-in-time action for a VM. (Creating a snapshot does require a little more space on disk, so be careful.) Just in case we mess up somewhere, or want to try something different with our VM, we should take a snapshot now.

Do this by selecting on the window for the VM `Machine > Take Snapshot` . Name the snapshot something like "First" or "Initial" or "Just Installed" and press `Okay` .



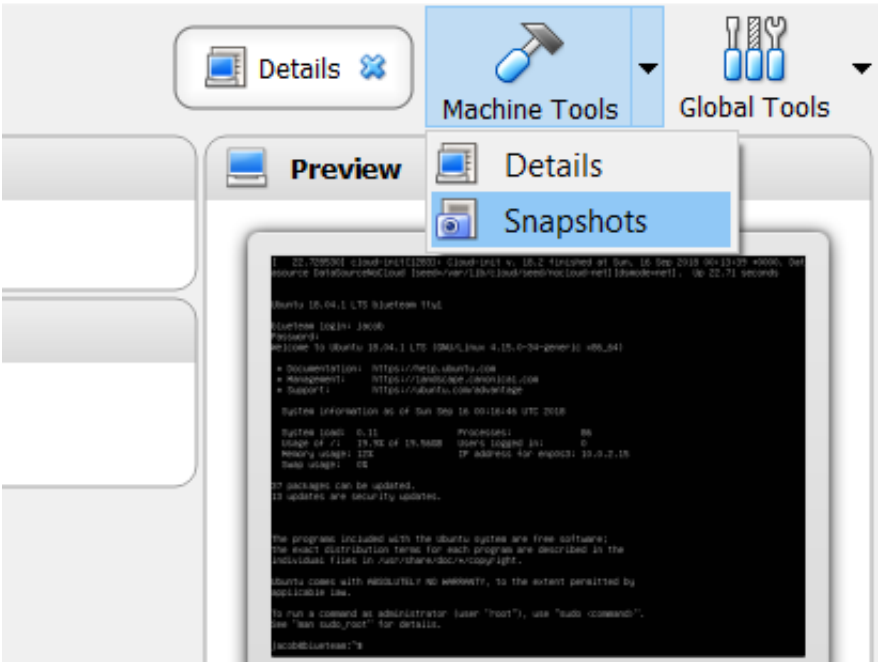
Reverting (Going back) to a Snapshot

To go back to the snapshot and restore a VM to the state of the snapshot, first make sure the VM is shut off. THis can be done with:

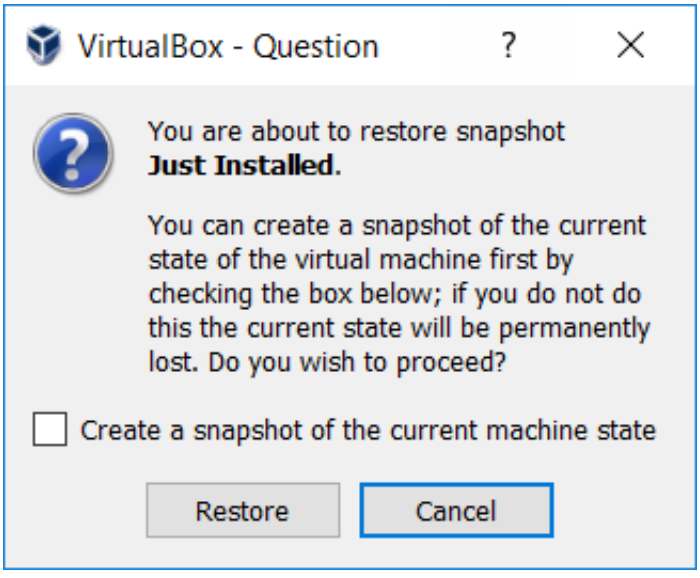
- `sudo poweroff` , type in your password and press enter. This is the recommended method.

- Right click the VM in the main window, and select `Close > Power Off` and then pressing `Power Off`. This is not recommended, as this is essentially pulling the plug on a phyiscal machine.

After the VM is shut off, go to the main screen and click the arrow next to the `Machine Tools` button. This will have the options `Details` and `Snapshots`. Select `Snapshots`.



This will show a list of snapshots in a tree format. (This is because you can take snapshots after snapshots, and then go back and start a new tree of snapshots from a past point.) Select snapshot you made, then press `Restore`, This the window, you save the current state as a new snapshot by having the "Create a snapshot of the current machine state selected," otherwise, unselect it. Then press "Restore."



	Navigation	

{% include footer.html %}

layout: default title: Chapter 2 - Linux Basics

Assumptions

- You got the VM running form the last chapter. Test out the commands you learn while you read.

> Note: You'll starting see these hats around:  < This means to look out for red team stuff surrounding this topic.  < This means this is important to blue teams (you!). Be sure to know stuff about this! (Use Google to find more info)

Basics

The following site provides some good essentials in an easy format. Start here:

- https://ryanstutorials.net/linuxtutorial/

Going Faster

Using "Tab Complete" will attempt to autocomplete commands or file paths. It will complete to as far as is unique (no other commands or files match). If nothing appears when you press tab, this means there are multiple possibilities. Press tab to see your options.

> This is a great way to find commands if you've forgotten exactly how they are spelled.

You can also use the arrow keys to move around the console. Pressing up or down goes back in forth in the command history (it tracks what you typed before. The entire history can be viewed with the `history` command) while pressing left and right moves the cursor on the line (useful if you mis-typed something)

Users

This is mentioned a bit in the "Permissions" page in the tutorial above, but I'll give a little more

detail on users and groups.

Linux Users

Linux supports many users (in fact, your server already has a bunch of users!), and knowing what to do with users and groups is important to managing a Linux system. Linux users depend on their `uid`, or user id. The name doesn't matter, just the uid.

Root

 The most important user in a system is `root`. Root has a uid of 0, and any user with a uid of 0 is root (remember Linux only cares about the uid, and multiple usernames can have the same uid). Root is your administration account, keep it safe! Don't forget the password to access it or remove it! This account can do **anything** on the system, including changing critial applications and configurations.

> The goal of any red team or hacker is to get root!

Ubuntu, by default, does not allow direct root access to keep things a bit safer. To run a command as root in Ubuntu you can use `sudo`. `Sudo` is related to the `su` command, which is short for "switch user." (Note the `su` in `sudo`) On other systems, you could run `su -` (the `-` ensures you login properly as that user), type the password of the root user and become root. Here, we can't do that, so we use `sudo`. To run a command as root using `sudo`, type:

```
sudo <COMMAND>
```

Then type **your** password. `sudo` is nice as it allows you to run `sudo` again without re-entering the password for a period of time. If you want, you could type:

```
sudo su -
```

to switch to root, since you're essentially switching to root from root!

`Sudo` is also useful, since it can restrict users to only running certain commands as root.

Viewing Users

User info is stored in `/etc/passwd`. Password info for users is stored in `/etc/shadow`. These are just text files listing out users, so you can use `cat` to view them and other text tools to edit them. This makes it possible to add and remove users just by editing a file! 

`/etc/passwd` is readable by everybody. (The format can be viewed here). The main thing we care about is that it gives the username and uid.

`/etc/shadow` is readable only by root. (The format can be viewed here). This gives contains out passwords, though put a process called "hashing," which is essentially a one-way encryption. Passwords you type into prompts are hashed with the same the hash process and compared. This allows checking if a password is same without storing the password in the open (in plain-text).

Groups

All users are in one or more groups. These group users together to be given permissions all at once. Group info is stored in `/etc/groups`, which is also a text file. (The format can be viewed here)

Blue Team Tips

 Here's some blue team tips:

- Use `history` to see what commands have been executed before. See if anybody's been up to something.
- Check your environment using `env` to see what variables (like `$SHELL`)
- Keep `/etc/passwd` and especially `/etc/shadow` safe. Watch them for modifications by red team!
- Use `ls -la` to view all files and find dotfiles (`.file`)
- Use `file` to check what type a file is. This is great is for determining if something is not what it should be. Experiment with your system to get an idea of what things should and shouldn't be.

Futher Reading

- More Linux Basics

	Navigation	
< Ch 1	Home	Ch 3 >

{% include footer.html %}

layout: default title: Chapter 3 - Networking Basics

Assumptions

Networking

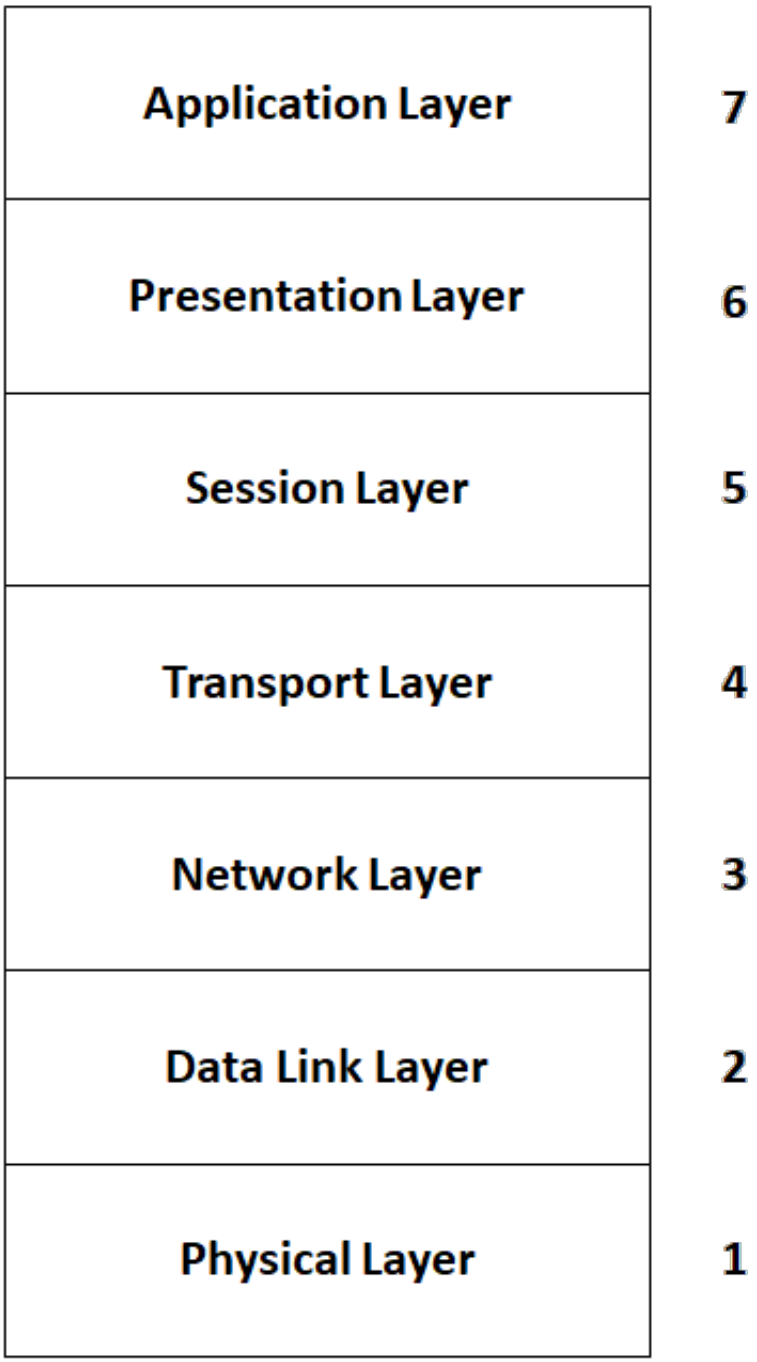
Networking is critical in the modern world, as we use it all the time to connect to Instagram and send emails. (You probably already knew this...)

This chapter is for giving a quick run down of networking and what is involved in moving data around. There's a lot, **lot** more than I'll go through here, but I'll try to give you what you'll need to do the absolute basics in the blue teaming.

What You'll Learn

- The OSI Model
- The different layers of network communications
- IP Addresses
- TCP and UDP
- Network Ports

The OSI Model



(By MichelBakni - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=66641106)

This is the OSI model, its used to divide the networking process in layers. The OSI has 7 layers, but we really care about the bottom 4 (Transport, Network, Data Link, Physical). Having layers allow different protocols to manage different parts of the communication process and be interchangable. (For example, at layer 1, we could use optical (fiber) or copper cables. At layer 2, we could use the protocols for wireless (802.11) or Ethernet. This is all without changing any other layer). The OSI model helps to encourage the abstraction of components to simply changing things around without messing everything else up.

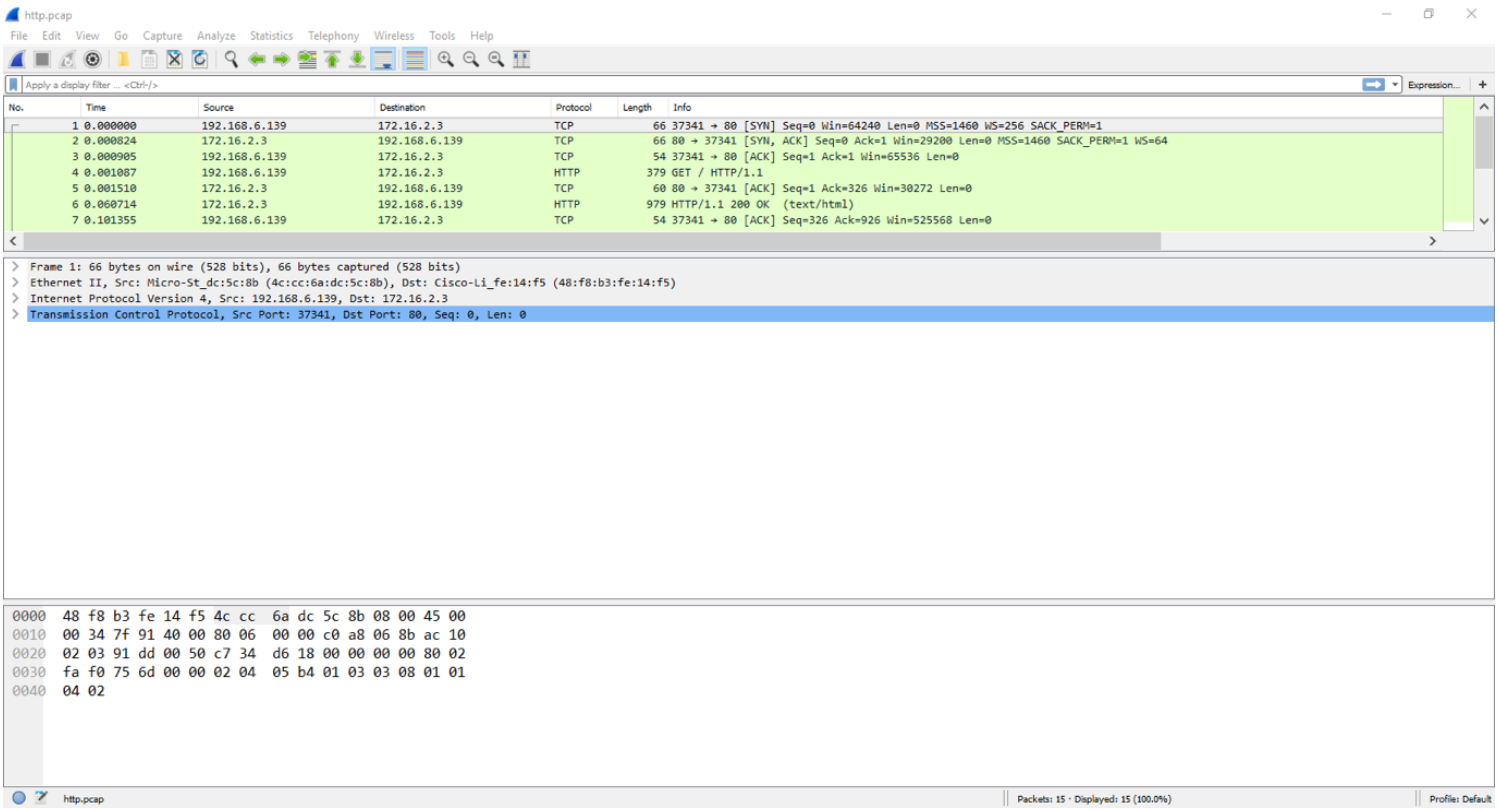
Each layer has its own encapsulation or packaging. The data starts at the top and is packed in

more and more layer information until it is sent. The layer information is used to move the data to its destination, where it is unpacked from all the layer information.

Seeing Network Communications

A tool commonly used to view network traffic as it enters and leaves a system is Wireshark. It's usually referred to as a packet sniffer. It has a commandline-based cousin called `tcpdump`.  . Packet sniffers are usefully for see everything that's coming in or out.

Wireshark gives us a nice interface, and displays the packet chopped up into layers. We'll use this to visualize network traffic to learn about the layers. If you want, you can download Wireshark, install it, and download this pcap (it stores packet capture data). You should be able to double-click the pcap once Wireshark is installed to open it.



The top is a list of packets. Click one to view details about it. The details and the layer sections appear in the middle. Click the arrows on the left to get more details on each section. The bottom of the screen is the raw bytes.

Physical Layer

This is the stuff we actually send on. This is usually a cable or wireless signals that carry the data from point A to point B. In virtualization, this is virtual as well.

Data Link Layer

```
Ethernet II, Src: Micro-St_dc:5c:8b (4c:cc:6a:dc:5c:8b), Dst: Cisco-Li_fe:14:f5 (48:f8:b3:fe:14:f5)
    Destination: Cisco-Li_fe:14:f5 (48:f8:b3:fe:14:f5)
    Source: Micro-St_dc:5c:8b (4c:cc:6a:dc:5c:8b)
    Type: IPv4 (0x0800)
```

The most common protocol you'll find here is Ethernet. In Wireshark, this is the second section from the top section. It should say `Ethernet II`. The main thing we'll focus on here is the two addresses marked by `Destination` and `Source`. These are MAC addresses. They are used to identify computers at layer 2. Layer 2 only cares about local communications between computers on the same network (for this, we'll be defining this as a collection of connected devices). It's kinda like apartment numbers. It's only relevent to a particular group.

MAC addresses are assigned on the Ethernet hardware, so you don't usually have to worry about giving a computer a MAC address. (You can, however, change the MAC address or make new MAC addresses for virtual network interfaces) A switch device will use a MAC address to move frames (what the packaging at this layer is called) from once device to another.

The `Type` field is used to indicate what's in the next layer up, in the Network layer.

Network Layer

```
Internet Protocol Version 4, Src: 192.168.6.139, Dst: 172.16.2.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x7f91 (32657)
    Flags: 0x4000, Don't fragment
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.6.139
    Destination: 172.16.2.3
```

The next section down in Wireshark is layer 3, the Network layer. The protocol that is almost exclusively used here is the Internet Protocol (IP)... protocol. There's two versions, IPv4, which uses addresses like `192.168.122.1`, and IPv6, which uses addresses like `FE80:0000:0000:0000:0202:B3FF:FE1E:8329` (Try memorizing that!) IPv4 is the most common (and easiest to remember). IPv6 was/is supposed to replace IPv4, as it has many many many more addresses you can use, but IPv4 is still the most common. You'll meet up with IPv4 most often, but don't ignore IPv6! It's still usable, and many commands have IPv6 versions.  The source and destination addresses are in the `Destination` and `Source` fields.

Layer 3 only cares about end-to-end communications, kinda like a street address. Similar to sending a letter around the world (or next door) with a street address, IP addresses can be used to send across the world (or the network next door).

This address is the one you'll be setting a lot and be messing around with the most. So we have a dedicated section to them.

IPv4 Addresses

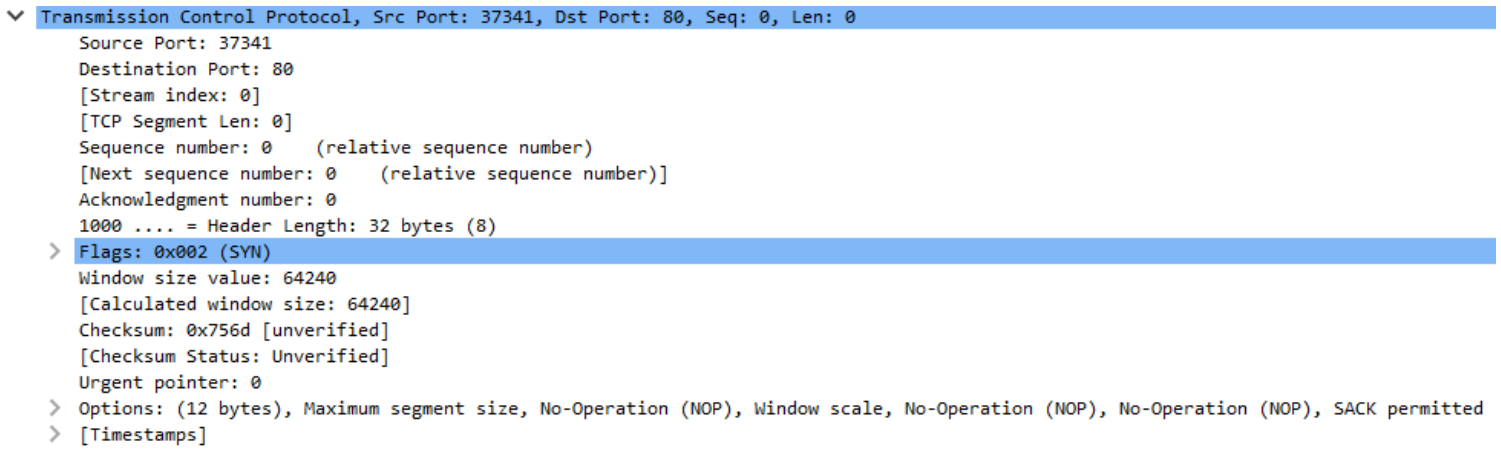
IP addresses are divided into octets (this is because there 8 binary digits per section): `<octect>.<octect>.<octect>.<octect>` . The biggest number that can go in the octet is `255` (the biggest number 8 binary digits can represent).

When using an IP addresses on a computer or network device, its always paired with what's called a subnet mask. This subnet mask is not sent with the data, but used by devices as a sort of guide. Subnet masks are used by devices to determine what's in their local network and what's not. They usually look something like this: `255.255.255.0` (similar to an IP address) or in "slash notation" `/24` . (This number is the total number of bits in the subnet mask, remember each section has 8, so `8 + 8 + 8 = 24`) The number of bits in a subnet mask can be increased and decreased to make the local network bigger or smaller in a process called subnetting. We won't get into subnetting here, it's a bit complicated. (If your interested, there's plenty of resources online)

There are two special addresses in a local network. The first address numerically (the `.0` when using `/24`) is reserved to represent the entire network as the "network address". Don't assign it to a device. The second address is the "broadcast address." This is the last address (`.255`) and it is used to send to everybody on the local network. Also do not assign this address to a device.

There's also another special address, `255.255.255.255` . This is the broadcast address of everything and any network. If this is the destination address, then its basically addressed to everybody! Don't assign this address to any device.

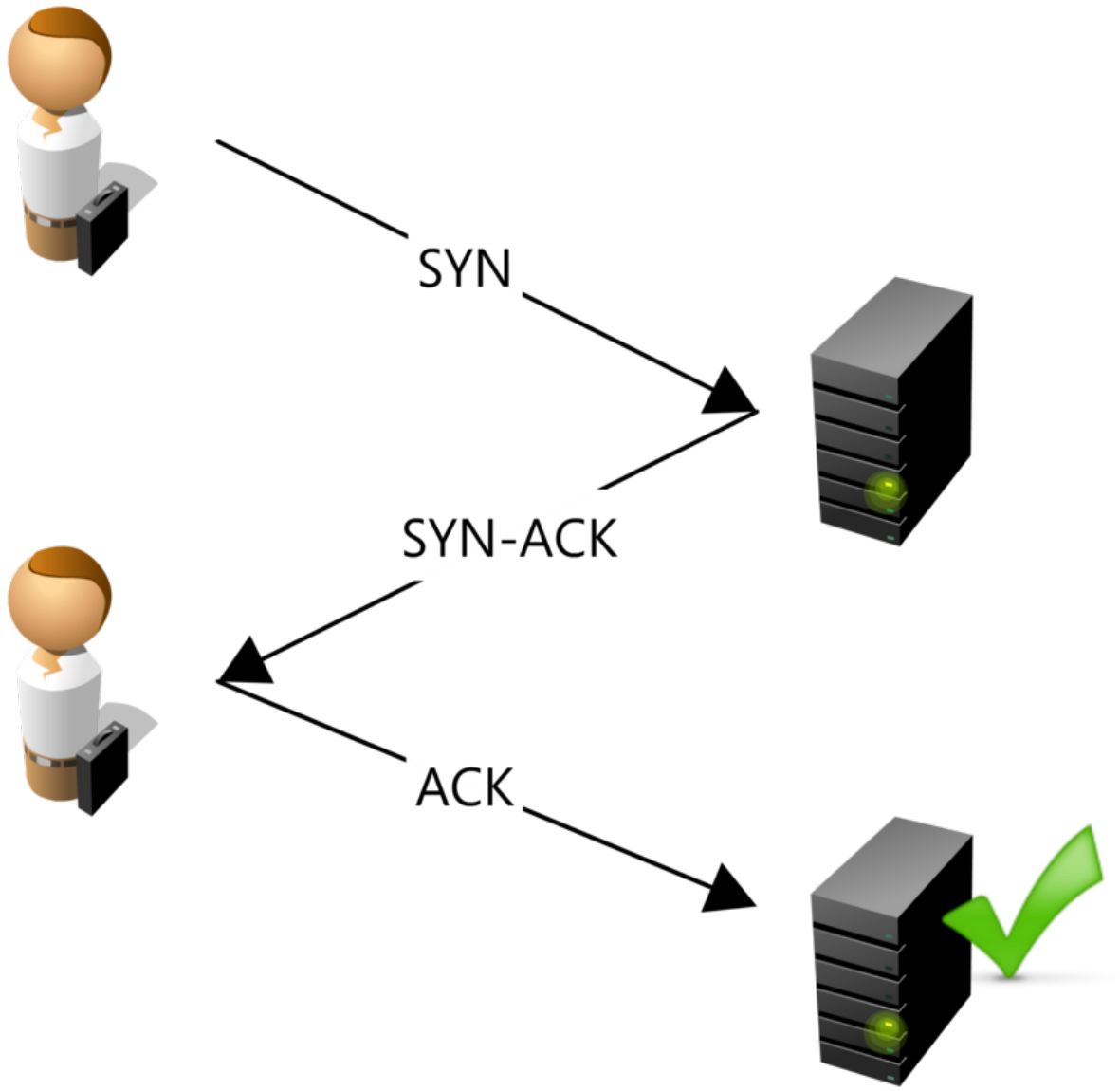
Transport Layer



The next layer is the Transport Layer. It doesn't have addresses, but is used to identify what service the data is destined for. This layer also manages how the whole data arrives. There are two main protocols for this layer, `TCP` and `UDP`. Our example uses `TCP`.

TCP

TCP ensures two sides are talking and the data arrives intact and in the right order. It uses the "three-way-handshake" (the first three lines in Wireshark is this handshake) to negotiate the connection.



(By N-21 - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=7972116)

TCP has flags to indicate the state of the connection. A sender sends a packet with a SYN flag, and gets a packet with the flags SYN and ACK (acknowledgment) set. A packet with ACK is sent in response to complete the handshark. TCP uses sequence numbers to keep things in order, with packets sent with ACK flag set to acknowledge data up to a certain point has been received. When its done, TCP uses another handshake (the last four lines in Wireshark) with FIN (finished) and ACK flags to tell each side they are done.

UDP

The other protocol, `UDP` is much easier. It doesn't care if a connection is made and essentially

just sends the data without bothering to check if it gets there.

Ports

Ports are used by both TCP and UDP to determine what service data is destined for. (e.g. is the data for a web server? a mail server?) Services will reserve "listening" on a port, waiting for data destined for that port. Both ends will have a port. Services, like web servers, listen on "well-known" ports that have been assigned to that type of servive. For example, web servers listen on port 80 (You can see this in the Wireshark above as the `Destination Port`). The port of the sender is usually randomized from ports of a larger port number (maybe something between 32768 or 61000, the Wireshark image above has port `37341`) and identifies the sending program in the response.

> On most systems, using ports below 1024 requires admin privileges 

Basically, any program can listen on a port, so it's a good idea to always see what's listening for data. This can be done with the `netstat` or `ss` commands with the options `-tunap`. 

Conclusion

There's a lot more to networking than this. Hopefully, this will get you started on the basics of networking.

	Navigation	
< Ch 2	Home	Ch 4 >

{% include footer.html %}

layout: default title: Chapter 4 - More Networking

Assumptions

- You read the last chapter
- You have a basic understanding of IP addresses and subnet masks.

More Networking

Since networking is so critical to managing your environment, here's another chapter.

What You'll Learn

- Routing Basics
- NAT
- ARP
- Setting Addresses

Routers and Routes

Most networks will have at least one router. A router can be a Cisco device, a small home router, or even a linux box with more than one network interface. It does what the name suggests, and helps route stuff to its destination.

A network needs a router because it is implausible that device will know how where to send data to every other system in the world, so this job is passed off into a series of routers. These routers know where to send data for certain destinations or know who to hand it off to if they don't.

Default Gateway

When any system doesn't know where to send some data, it will refer to its "default gateway." Systems will need to configure their default gateway or get it from DHCP.

Setting the default gateway manually:

- Centos
- Ubuntu <16.04
- Ubuntu >=18.04

Routing Table

A system will determine this by looking at its "routing table." This holds a map of what paths to destinations the system knows about. The routing table will always have networks the system is directly connected to (all its local networks) because it obviously knows where to send data for those networks, just out the interface connected to that local network. Routing tables can also

be given "routes" by manual insertion or by "routing protocols" that transmit paths to destinations. These routes tell the system who to hand off data for a certain destination to. Routes also have subnet masks, which is used by the system to determine if a packet is bound for that network or not.

A system's routing table can be viewed on a system using the `route` or `ip route` (Linux only) command.

Linux:

```
jacob@web:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.2.1      0.0.0.0         UG    0      0        0 ens3
172.16.2.0      0.0.0.0         255.255.255.0   U     0      0        0 ens3
```

OR

```
jacob@web:~$ ip route
default via 172.16.2.1 dev ens3 onlink
172.16.2.0/24 dev ens3  proto kernel  scope link  src 172.16.2.3
```

Windows:

```
PS C:\Users\jacob> route print
===========================================================================
Interface List
 16...4c cc 6a dc 5c 8b ......Qualcomm Atheros AR8171/8175 PCI-E Gigabit Ethernet
...
 23...30 e3 7a ee 7d c5 ......Intel(R) Dual Band Wireless-AC 3168 #2
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      192.168.6.1    192.168.6.139     25
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
...
===========================================================================
```

Routing Process

When a system wants to send something, it'll go through a process like this:

- Check the IP it wants to go to against the routing table.
  - If it has the network in the routing table, and its a local network, it will send it to the device on that local network.
  - If it has the network in the routing table, and its a route, it will pass the data on to the other device in the route
  - If it does NOT have the network in the routing table, it will pass it off to the default gateway.

Essentially, this process happens until the data reaches its destination. Then the destination has to do the exact same process again, just in reverse.

 Routing is a two-way street, you must know how to get there, but they also need to know how to get back!

Network Address Translation (NAT)

Network Address Translation (NAT) is a widely used process on converting addresses from private IP addresses, which can't be routed on the internet, to public IP addresses, which can be routed on the internet. (If you've used a home router, you've used NAT before.) This is why your IP address that appears in sites like https://www.whatismyip.com/ might be different than the address you get when running `ipconfig` or `ifconfig`.

ARP

ARP is a protocol that is used on a local network to find MAC addresses for a given IP address. A system will use ARP request to ask which system has a given IP and what their MAC address, so the system can send stuff on layer 2(Data Link Layer). The system that has that IP will response with an ARP response.

>  This protocol has no verification mechanism, so any system can pretend to be another system by responding to the ARP request. Using tools such as Ettercap, the red team can intercept local network traffic (only on a local network) by pretending to be the destination system. Lookup "ARP man-in-the-middle-attacks" and Ettercap for more info.

Giving a machine a IP Address

To communicate, we need to give our systems IP addresses. This can be done in two ways.

DHCP

DHCP is a network protocol that allocates and communicates IP addresses to systems. A device will send out a message indicating they want an address, and the DHCP server will respond, assigning them an IP address, which is the device's responsibility to assign to itself, and providing the default gateway, network DNS servers and other network information. DHCP records these address allocations and makes sure nobody gets a duplicate IP address.

DHCP is usually used by desktop and workstations. Servers, like the ones you will be defending, will not use DHCP, and you must assign them their address.

Static IPs

Static IPs are when you manually give a system an IP address.  This means you have to track your addresses yourself.

>  A system will NOT usually check if they have been given an static IP address that is already been given out. If you are getting intermittent connection problems, you may have allocated the same IP address on two systems. To test this, you can use the `arping` tool to see if two systems respond to your ARP message. See here and here for more info.

Here are some guides on how to set static IPs on a few Linux types:

- CentOS 7
- Ubuntu <=16.04 (Note you can also put `dns-nameservers <DNS_IP>` after the `gateway` line in `/etc/network/interfaces` to set the DNS server.)
- Ubuntu 17.04+

	Navigation	
< Ch 3	Home	Ch 5 >

{% include footer.html %}

layout: default title: Chapter 5 - Being an Admin

Assumptions

- You read the last chapter
- You have a basic understanding of the networking topics we have covered.
- You have a server setup with an IP address.

Accessing your System

To configure your server, you'll most likely use SSH. SSH stands for Secure Shell, and is a very common way to remotely access the command line interface of a system. SSH communications are encrypted, so the commands you type can't be seen if intercepted.

To SSH into a box (people do use this as a noun), on a Linux system type:

```
ssh <USER>@<IP_OF_SYSTEM>
```

On Windows, use the PuTTY client to ssh into a system.

For both of these, a message will appear indicating if you trust this system. This happens when you first connect to a server, and indicate you want to continue (`yes` or `OK`). This will store the fingerprint of the server on your system, which is used to identify it in the future. This shouldn't change (you'll get a nasty error message if it does), unless you recreate the box (which creates a new fingerprint) or delete the server's fingerprint data.

Looking Around

When you SSH in, you'll be placed into the user's home directory. You have already seen this, but you can easily refer to the user's home directory with the `~` symbol, so `~/Desktop` means the `Desktop` directory in the current user's home directory. The home directories are usually in `/home/<USERNAME>` , or for root, `/root` .

The Shell

Usually the shell you are using is `bash` or `sh` . `bash` is actually a upgrade/successor to `sh` , but `sh` is usually still around. You can view your shell by running:

```
echo $SHELL
```

> Lookup bash scripting on how to utilize all the cool features of bash, which is almost a full programming language and allows you to automate tasks on the system.

Your User

The current user should usually be displayed in the shell something like this:

```
<USER>@<HOSTNAME> $
```

If there is no username in the prompt, run the `id` command to view your current user, the user's uid and the groups the user is in. You can also do

```
id <USERNAME>
```

to view the same info for that user.

Managing Users

 As we've seen before, the simplest way to view existing users on the system is by reading `/etc/passwd`. You can view users that are logged in or have logged in in multiple ways:

- `lastlog`: Prints out the last time the user logged in
- `last`: Prints a list of user logins and system reboots. More Info
- `w`: Prints out who is currently logged in.
- Files `/var/log/secure` (CentOS) or `/var/log/auth/log` (Ubuntu): These files contain the logging output in regards to logins locally and remotely.

Changing User Passwords

This is an important element of your user management, changing their passwords.  Do this especially quick at an event since the existing passwords on the system are usually known to the red team or very easy to guess.

To change your own password, you need to know the existing one. If you don't, well, you're out of luck, unless you know the password for root. Root is the only user that doesn't need to enter

the existing password to change it. Root can change anybody's password.

To change the password, run the following command:

```
passwd <USERNAME>
```

> Note: During the next few prompts, no text will appear as you type, this is intentional to hide the password while you type.

If no username is given, the current user is assumed. If you are not root, you will first be prompted for your existing password. After this or if you are root, you will need to enter the new password twice, once to set it and the next to verify it. Now your new password should be in effect.

Services

Usually, applications such as web servers and email are run on a system as services. Services are processes that run in the background and controlled through a service manager. This service manager helps ensure any processes the service needs are running and that files exist. This also allows services to be centrally managed.

In most recent (as of 2019) Linux distributions, services are managed through two commands. `systemctl` and `service` (which is mostly a wrapper for the `systemctl` command these days.)

Service Names

Services are referenced by name, which can be different across Linux distributions, for example, the SSH server is referred to as the `ssh` service on Ubuntu, while referred to as `sshd` on CentOS. On Ubuntu, the Apache web server is `apache2` while on CentOS, it is `httpd`.

Mainly you'll have to get used to these naming changes. You can usually find out the service name by looking up that service for that Linux distro. You could also use the following commands to list out the services to see if you can identify the service you are looking for.

```
service --status-all
systemctl list-unit-files
```

Service Control

By running applications as services gives you centralized control over these servers. This allows

you to start, stop and restart service process much easier. If you didn't use the service manager, you'd have to manually start the server process, ensuring the right command line options are given and manually stopping the process (by name or figuring out its process id, or PID).

Basic Controls

To start a service:

```
service <SERVICE_NAME> start
```

or

```
systemctl start <SERVICE_NAME>
```

To stop a service:

```
service <SERVICE_NAME> stop
```

or

```
systemctl stop <SERVICE_NAME>
```

Restarting

Almost all service need to be restarted after modifying their configuration files. The easiest way to do this is through the service manager.

To restart a service:

```
service <SERVICE_NAME> restart
```

or

```
systemctl restart <SERVICE_NAME>
```

Service Status

Another handy feature of service managers is getting the status of a service. This can tell you if

it is running, if the service crashed, and other info.

To get the status of a service:

```
service <SERVICE_NAME> status
```

or

```
systemctl status <SERVICE_NAME>
```


	Navigation	
< Ch 4	Home	Ch 6 >

layout: default title: Chapter 6 - DNS

Assumptions

- You read the last chapter
- You have a basic understanding of managing services.
- You can use a editor such as `vi` to edit files on a server.

DNS

DNS is the service that turns names (like `google.com`) into IP addresses. It's kinda like a phone book, which is used to map people's names to their phone number. A system will use a variety of resources to "resolve" or convert name into an IP. A more detailed look at DNS info can be read about here in an article from CloudFlare. I recommend you read it.

DNS Host Configuration

Hosts File

The simplest way a system converts names to IPs is through the hosts file. Both Linux and

Windows have this, and its a simple and easy way to add name mappings. For Windows, the file is at `C:\Windows\System32\Drivers\etc\hosts`, while on Linux, it is just `/etc/hosts`.

The format is simple:

```
<IP> <NAME>
```

A system always checks this hosts file first, so it's a good way to short-circuit the resolution process. This is used to redirect ad sites to block adds, or override your resolution to critical sites, so you can't get there or go a fake one! 

Setting a Nameserver

After the hosts file, the system will attempt to to turn a name into an IP, also known as resolving the name, through a configured nameserver. Essentially, the system is asking that DNS server to do the mapping of name to IP. (We'll make our own DNS server we can point our systems to later) This can be configured in a few different ways.

`/etc/resolv.conf`

The main file that contains what options the system will use for DNS resolution is `/etc/resolv.conf`. The format of the file is as follows:

```
domain <DOMAIN>
nameserver <SOME_IP>
```

The `domain` option is not required, and is used to append to names. This would allow you to use only a part of the name instead of the full name (or Full Qualified Domain Name, FQDN for short).

The more important setting is `nameserver`, which sets the server the system will send DNS requests to resolve the name.

> In many modern systems though, `/etc/resolv.conf` is generated by the network management service, so if you edit it, your changes may not stick around for long. Do it as a last resort. Use the following methods to configure the DNS name.

CentOS 7

In the same file you used to set the static IP on your CentOS 7 system, add the following if you haven't already. You only require 1 DNS server to resolve, but having a second is a good backup

in case the first DNS server fails (for example, if you set the first to be your DNS server and the second to something else, if your DNS server goes down, you might have a backup).

```
DNS1=<NAMESERVER_IP_1>
DNS2=<NAMESERVER_IP_2>
```

Ubuntu 16.04

In `/etc/network/interfaces`, under the interface you set the static IP on, add the following:

```
dns-nameservers <NAMESERVER_IP_1> <NAMESERVER_IP_2>
```

Note the space between the addresses.

Ubuntu 18.04

In the `/etc/netplan/01-netcfg.yaml` file you edited to set the static IP, add the following at the same indentation after the addresses and gateway:

```
nameservers:
    addresses: [<NAMESERVER_IP_1>, <NAMESERVER_IP_2>]
```

Note the commas after the first IP.

Building a DNS Server

To practice managing and configuring a service, we'll dive into making a DNS server. Thankfully, default configurations will give us a basis to work upon and overall the process is pretty straight forward.

You can configure your DNS server in different "modes":

- **Authoritative**: This DNS server holds actual information for a domain. DNS queries will usually finally arrive here, and the authoritative server will return a result, either the name exists, doesn't exist, or speak to another server.
- **Recursive**: This DNS server will do the searching for you. If it doesn't have the name cached, it will work its way up the domain portions (the parts separated by `.`) from root, who tells it where to find info on the Top Level Domain (TLD, like `.com` or `.net`), to the TLD server, which tells it where to find the DNS server for the next part and so on.

- **Forwarding**: This DNS server will just forward requests, but also cache the results for later.

We'll be doing a Authoritative and Forwarding server on **Ubuntu 16.04**.

Install the Service Packages

On a command line (connected through SSH is recommended), run the following command to install the necessary service package for BIND, a popular DNS server, (`bind9`) and some tools (`bind9utils` and `dnsutils`):

```
sudo apt-get install bind9 bind9utils dnsutils
```

When prompted, type `y` and Enter to continue the installation.

Locating the Configuration Files

A majority of service configuration files are stored in `/etc/` . The config files for BIND on this version of Ubuntu are in the `/etc/bind/` directory.

> Note that other Linux distributions may have these files at different locations, so research the location if you can't find where the files are.

The main config file is `/etc/bind/named.conf` . Note the name of the file is not related to BIND, but a combination of `name` , for being a domain name server, and `d` for daemon, which is another name for a service process that runs in the background. Services may do this sometimes.

Editing the Configuration Files

Almost all configuration files for services can only be edited by `root` , so be sure to be `root` or use `sudo` .

> Hopefully you know `vi` ...

Let's open the `/etc/bind/named.conf` file. Note though that there's not much here:

```
sudo vi /etc/bind/named.conf
```

The contents:

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

The lines with `//` in front are comments, similar to the C language. The lines with `include` mean that we are including other config files into the main config file. Many services have ways of including configurations from other files, so you don't have one massive config file that's hard to manage and script.

> Note that the configuration format that BIND uses is not universal, different services have different formats, so be sure to research how to edit their formats correctly.

Setting up Forwarding

To setup forwarding to forward our DNS requests, we need to edit the `/etc/bind/named.conf.options` included into the main config file.

```
sudo vi /etc/bind/named.conf.options
```

In the file, uncomment the following portion of the file by removing the `//` in front (With `vi`, use the arrow keys to navigate and use the delete key to remove them):

```
        // forwarders {
        //      0.0.0.0;
        // };
```

It should look like this:

```
        forwarders {
                0.0.0.0;
        };
```

However, `0.0.0.0` is invalid to send DNS requests to, so we need to set this to another IP. If you are in a shared lab, set this to a local DNS server (ask your systems administrator). If you are at home, set this to something like `8.8.8.8`, which is Google's public DNS resolver.

> Be sure to add that `;` at the end! It gets me every time...

For example:

```
        forwarders {
                172.16.10.10;
        };
```

Save and exit out of `vi`, and then restart the service to enable the changes.

> Note: Most services won't reflect the changes until they are restarted!

But, uh, we don't know the service name... Using the `sudo service --status-all` command can list out the services and we can see if we can identify it. Your output will probably be different.

```
jacob@testdns:~$ sudo service --status-all
 [ - ]  bind9
 [ - ]  bootmisc.sh
 [ - ]  checkfs.sh
 [ - ]  checkroot-bootclean.sh
 [ - ]  checkroot.sh
 [ - ]  console-setup
 [ + ]  cron
 [ - ]  hostname.sh
 [ - ]  hwclock.sh
 [ - ]  keyboard-setup
 [ - ]  killprocs
 [ + ]  kmod
 [ - ]  mountall-bootclean.sh
 [ - ]  mountall.sh
 [ - ]  mountdevsubfs.sh
 [ - ]  mountkernfs.sh
 [ - ]  mountnfs-bootclean.sh
 [ - ]  mountnfs.sh
 [ + ]  networking
 [ + ]  ondemand
 [ + ]  procps
 [ + ]  rc.local
 [ + ]  resolvconf
```

```
 [ + ]  rsyslog
 [ - ]  sendsigs
 [ + ]  ssh
 [ + ]  udev
 [ - ]  umountfs
 [ - ]  umountnfs.sh
 [ - ]  umountroot
 [ + ]  urandom
```

There's a service called `bind9` near or at the top, same as the package we installed! We can guess this is the service name we want to use to restart.

```
sudo service bind9 restart
```

To check the status of our service, change the `restart` to `status` to get the service's status (Your output will be different):

```
jacob@testdns:~$ sudo service bind9 status
* bind9.service - BIND Domain Name Server
   Loaded: loaded (/lib/systemd/system/bind9.service; enabled; vendor preset: ena
  Drop-In: /run/systemd/generator/bind9.service.d
           └─50-insserv.conf-$named.conf
   Active: active (running) since Thu 2019-10-24 03:24:46 UTC; 4s ago
     Docs: man:named(8)
 Main PID: 1627 (named)
   CGroup: /system.slice/bind9.service
           └─1627 /usr/sbin/named -f -u bind

Oct 24 03:24:46 testdns named[1627]: command channel listening on 127.0.0.1#953
Oct 24 03:24:46 testdns named[1627]: configuring command channel from '/etc/bind/
Oct 24 03:24:46 testdns named[1627]: command channel listening on ::1#953
Oct 24 03:24:46 testdns named[1627]: managed-keys-zone: loaded serial 0
Oct 24 03:24:46 testdns named[1627]: zone 0.in-addr.arpa/IN: loaded serial 1
Oct 24 03:24:46 testdns named[1627]: zone 127.in-addr.arpa/IN: loaded serial 1
Oct 24 03:24:46 testdns named[1627]: zone 255.in-addr.arpa/IN: loaded serial 1
Oct 24 03:24:46 testdns named[1627]: zone localhost/IN: loaded serial 2
Oct 24 03:24:46 testdns named[1627]: all zones loaded
Oct 24 03:24:46 testdns named[1627]: running
```

It should say `loaded` and `Active: active (running)`. If not, the text on the bottom is log output for the service, which is helpful to see what went wrong.

Testing out Forwarding

It's always good to test your own services to make sure they are running as you expected. They may start, but they may not give the results you want.

To test BIND, we'll use the `dig` command, a common tool for testing DNS. It has many more options than we'll explore to day, so be sure to look it up sometime.

We'll test to see if can resolve `google.com`. The `@127.0.0.1` forces `dig` to use a certain DNS server, in this case our local one. Otherwise, `dig` will use the DNS server address is `/etc/resolv.conf`.

```
dig google.com @127.0.0.1
```

`dig` returns a status in its output, look at the `status` field in the result. If things aren't going well, things may hang as `dig` or the DNS server waits for a result.

For example, this is when a request times out:

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.com @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 49581
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.            IN  A

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Oct 24 03:32:50 UTC 2019
;; MSG SIZE  rcvd: 39
```

A successful result may look similar to this:

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> google.com @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50143
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
```

```
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     299 IN  A   172.217.12.142

;; AUTHORITY SECTION:
.           261290  IN  NS  b.root-servers.net.
.           261290  IN  NS  m.root-servers.net.
.           261290  IN  NS  f.root-servers.net.
.           261290  IN  NS  h.root-servers.net.
.           261290  IN  NS  e.root-servers.net.
.           261290  IN  NS  j.root-servers.net.
.           261290  IN  NS  a.root-servers.net.
.           261290  IN  NS  g.root-servers.net.
.           261290  IN  NS  d.root-servers.net.
.           261290  IN  NS  i.root-servers.net.
.           261290  IN  NS  c.root-servers.net.
.           261290  IN  NS  l.root-servers.net.
.           261290  IN  NS  k.root-servers.net.

;; Query time: 50 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Oct 24 03:38:02 UTC 2019
;; MSG SIZE  rcvd: 266
```

If you get a successful response, you're all set for this section. If not, use the logs the `status` to diagnose the problem. More log output is available in `/var/log/syslog` (only readable as root). Use `cat` and `tail` to view portions of the log and watch for output when the service is restarted.

Setting up a Domain

Now let's set up our own domain. If your at home, you can use something like `YOURNAME.local`. If your in a shared lab, the system administrator or instructor might have on for you.

First, edit the `/etc/bind/named.conf.local` file as root:

```
sudo vi /etc/bind/named.conf.local
```

Add the following lines to the file: with `DNS.NAME` being the domain name you selected:

```
zone "DNS.NAME" {
    type master;
    file "/etc/bind/DNS.NAME.fwd";
```

```
    allow-transfer {none;};
};
```

- The first line sets the block for our zone, which is basically everything in and under the domain name.
- The second line indicates we are configuring the "master" for the domain. This means we will edit and configure the zone here. There can be "slave" DNS servers that just store what the "master" servers have.
- The next line sets the file containing the zone data to `/etc/bind/DNS.NAME.fwd`. The `fwd` is short for foward, as this is a forward zone, meaning we're doing NAME->IP, a "reverse" zone does IP->NAME, which is useful on occasion.
- The next line blocks "zone transfers". A zone transfer essentially is a report of all the information about the zone.  While this may be useful when transferring DNS info between servers, attackers can use this information to learn about your systems, so its good to restrict access to transfers for block them all, as we are doing here.

Once we save and exit, we need to create the zone file and add entries called "records."

```
sudo vi /etc/bind/DNS.NAME.fwd
```

Headers and SOA record

To start, add the following lines as a header and first record (note, keep the `.` on the end when you edit the domain):

```
$TTL 86400
@   IN  SOA     DNS.NAME. root.DNS.NAME. (
        0000  ;Serial
        3600        ;Refresh
        1800        ;Retry
        604800      ;Expire
        86400       ;Minimum TTL
)
```

These values are required in a zone file, giving administrator contact info, time-to-live (TTL) which indicates how long results from this domain should be stored, and other data. Research `dns SOA records` for more info. The main field we are interested here is the serial. This is used to indicate the "version" of the zone file, and should be changed with each edit. In a larger setup, this would tell slave DNS servers that there is an update to the zone, and they need to update their own zone information. If you don't change the serial, other servers won't know they

need to update.

A common pattern is `<4-DIGIT-YEAR><2-DIGIT-MONTH><2-DIGIT-DAY><2-DIGIT-INCREMENTAL-VALUE>`, such as `2019102401`, which is the first serial for Oct 21, 2019.

NS records

The next record we need to have is setting the nameserver, `NS` records. This tells what server is authoritative for the domain, and would list other nameservers if they existed for this domain. Note the trailing `.`, it needs to be there! (`@` stands for the base domain or domain root, which is `DNS.NAME`)

```
@       IN  NS          ns1.DNS.NAME.
```

`ns1` is a common name for the nameserver, we'll set who will get this name next.

A records

The `A` records are the ones that map a name to an IP. They are formatted like this:

```
<NAME>      IN  A   <IP>
```

- The `<NAME>` is the name we are adding to the domain, so setting `<NAME>` to `stuff` means we would create the full name of `stuff.DNS.NAME`
- `IN` means this is an internet address
- `A` is our record type
- `<IP>` is the IP we want to map to the name,

First we add one for the nameserver `NS` record, which is the server we are configuring.:

```
ns1         IN  A   <YOUR_SYSTEMS_IP>
```

Then we can add whatever names we want in new records! For example:

```
coolstuff   IN  A   192.168.1.2
```

Sets `coolstuff.DNS.NAME` to map to `192.168.1.2`.

When you've added the names you wanted, save and exit, then restart the service.

Testing our Domain

Use `dig` again to test our domain.

```
dig ns1.DNS.NAME @127.0.0.1
```

If everything is set up right, you should get a result and congrats, you have a DNS server! All you need to do is point your other systems to your DNS server so they can resolve the domain you've configured. Be sure if you have a firewall, you've opened the necessary ports, UDP and TCP ports 53.

Conclusion

There's much, much more on DNS. This was a quick look to get you familiar with the protocol and managing a service. More research will lead to things such as reverse zones, DNS cache poisoning, open DNS resolvers, MX records for mail, DNSSEC, and more. DNS is a critical service for the Internet, so you should learn more about it.

	Navigation	
< Ch 5	Home	

{% include footer.html %}

- Red Team Hat: GPL, https://commons.wikimedia.org/w/index.php?curid=451026
- Spartan Warrior: SAWg3rd [CC0], via Wikimedia Commons