PORT 22 AND YOU

A Match Made In Shell

Graham Helton

Agenda

01 Quick overview of SSH

02 Problems and their SSHolutions

03 Me rambling about SSH until someone physically removes me

04 Some allegedly theoretical stuff

/usr/bin/whoami

- Red Team Specialist @ Google
- Scope = Offensive Architecture & Advisement
- Blog -> GrahamHelton.com
- Socials -> GrahamHelton.com/links

- Linux Enjoyer
  - Hater of Windows
- Big fan of cast iron skillets, deep rabbitholes on random things, and cats

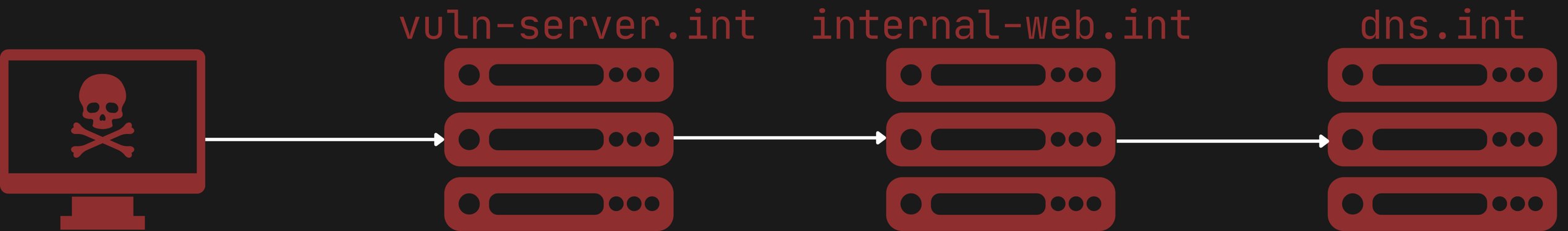
Why this talk

- Understanding how to use SSH tunnels will help you immensely in your day to day shenanigans
  - bhop through jumphosts
  - Connecting to lab and/or client networks
- Security is hard. Spend your valuable brain juice on the unique things, not network movement
- Don’t try to memorize all of it (or do, nerd)
  - grahamhelton.com/blog/ssh-cheatsheet

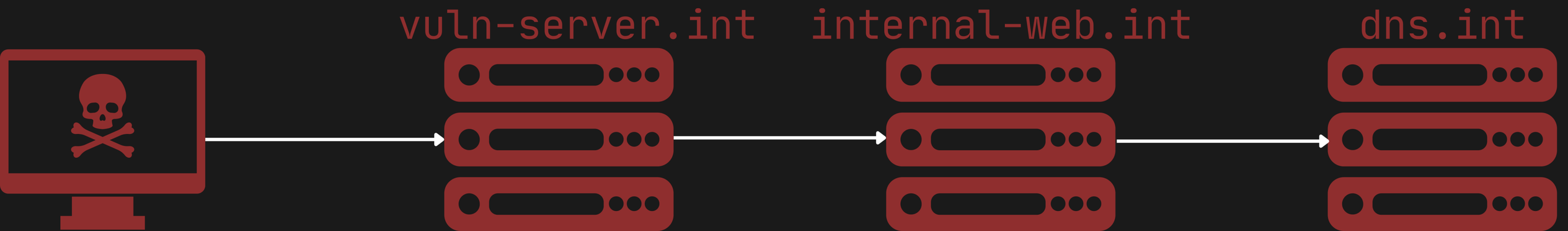
Jumphosts (-J)

- Lets start things out fairly simple...
- Lets send our traffic through a bunch of hosts
  - Jumphosts/Bastions/Edges/Jumpservers...
  - Annoy the SOC (Or evade detection?)
- `ssh -J root@vuln-server.int,root@internal-web.int root@dns.int`

- Fun project: What happens if you were to SSH back to your original machine at the end of your jumps?



```
smores@campfire:~ $ ssh -J root@vuln-server.int,root@internal-web.int root@dns.int
root@vuln-server.int's password:
root@internal-web.int's password:
root@dns.int's password:
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.13.19-1-pve x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

You have new mail.
Last login: Mon Aug 21 01:47:01 2023 from 192.168.1.185
root@Pihole:~# hostname
Pihole
root@Pihole:~#
```



Recap of SSHing with Keys

- Math is cool (I guess)
  - But I don’t really understand it so I’m going to say “it’s out of scope for this talk”

1. Generate a public and private keypair: `ssh-keygen -b 4096` ! ! !
   a. Upload the public key: `ssh-copy-id -i ~/.ssh/nashville_rocks root@bsides-crashville`
   b. The private NEVER leaves your machine
2. Connect to the server with the key: `ssh -i /path/to/private/key root@bsides-crashville`
3. sshd checks `~/.ssh/authorized_keys` for your public key
4. If your public key is in `~/.ssh/authorized_keys`, you’re allowed to connect ! ! !

- But there is an easier way

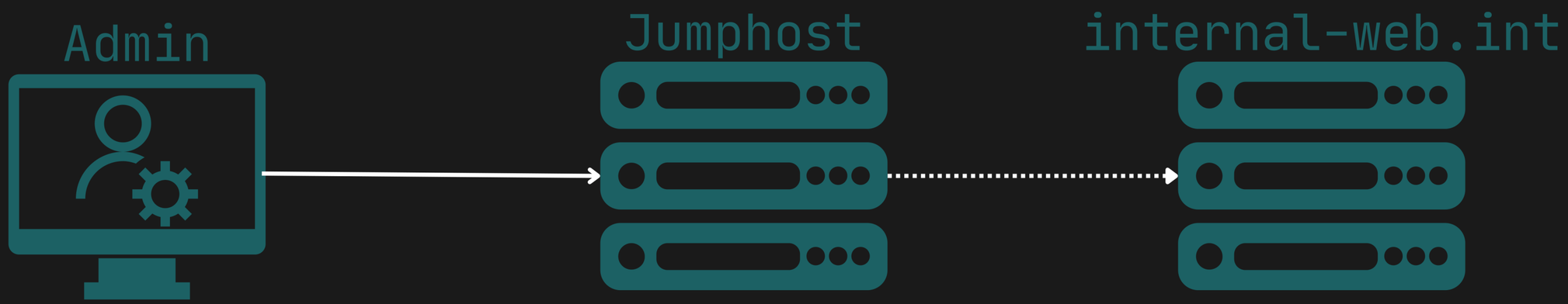
SSH Agent

- Add keys to a service running on your local machine
- Stores your private keys (unencrypted) in memory 🤭🤭🤭
- Now you can utilize the keys in the agent without explicitly calling them with SSH
- Super helpful for “admins” connecting to many machines
- This comes with a risk...

```
smores@campfire:~/ssh_agent $ ssh-add -l
The agent has no identities.
smores@campfire:~/ssh_agent $ ssh-add pihole
Enter passphrase for pihole:
Identity added: pihole (smores@campfire)
smores@campfire:~/ssh_agent $ ssh-add -l
4096 SHA256:llb6AquVVMuenl0tZeOfnCk9GilnVA59YjGMfIzCvhM smores@campfire (RSA)
smores@campfire:~/ssh_agent $
```

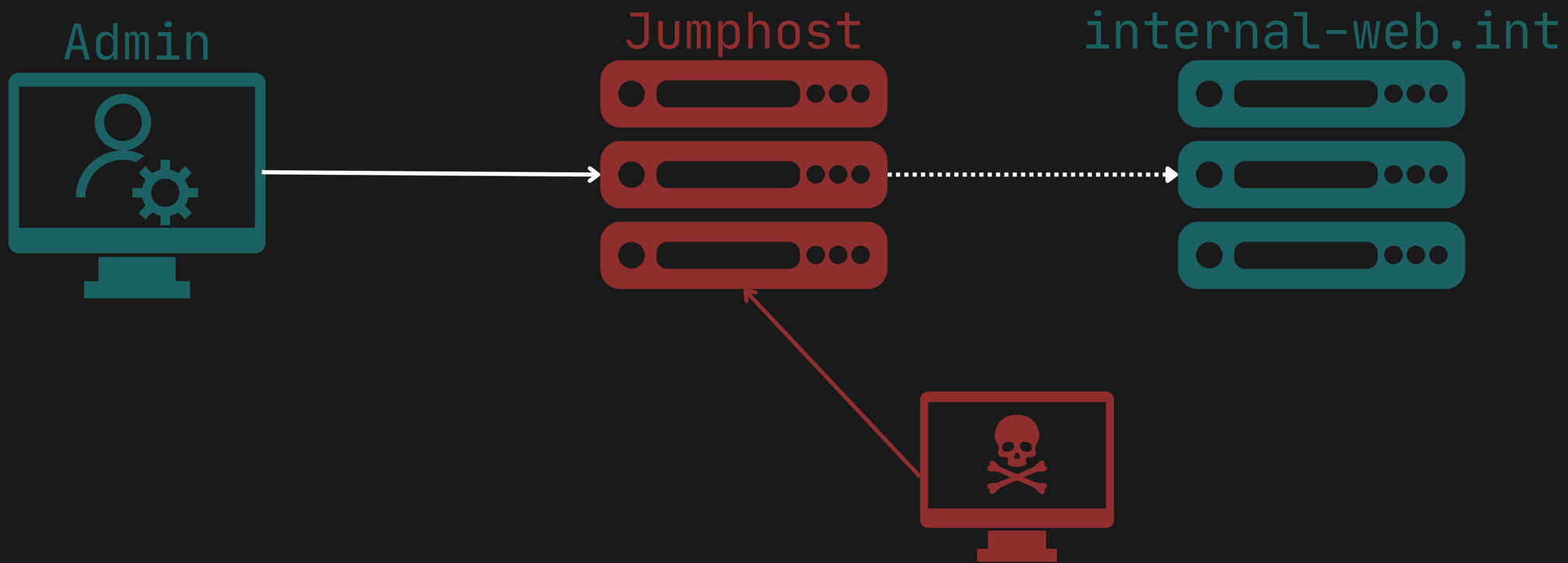
SSH Agent Part 2 Electric Boogaloo

- Agent forwarding, my beloved 🥰
- The legitimate (yet flawed) use case:
  - Admin want’s to access keys stored in SSH agent from the jumphost
  - Uses Agent forwarding (`ssh -A admin@jumphost`)
  - Admin can now access local ssh agent on jumphost



SSH Agent Part 2 Electric Boogaloo

- Attacker compromises jumphost
  - Attacker goal: Move laterally to other hosts in the network
- The attacker can now utilize the admin’s keys to do VERY LEGAL THINGS



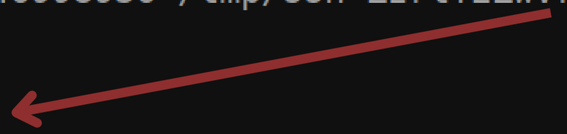
```
root@vuln-server:~# ssh-add -l
ssh-add -l
Could not open a connection to your authentication agent.
```

- Attacker lists keys in agent: `ssh-add -l`
  - No keys :(

```
root@vuln-server:~# ssh-add -l
ssh-add -l
Could not open a connection to your authentication agent.
root@vuln-server:~# lsof -U | grep agent
lsof -U | grep agent
sshd      4145          admin   11u  unix 0x000000007f0fda18      0t0 46998936 /tmp/ssh-ZzrtT2ZwVr/agent.4145 type=STREAM
```

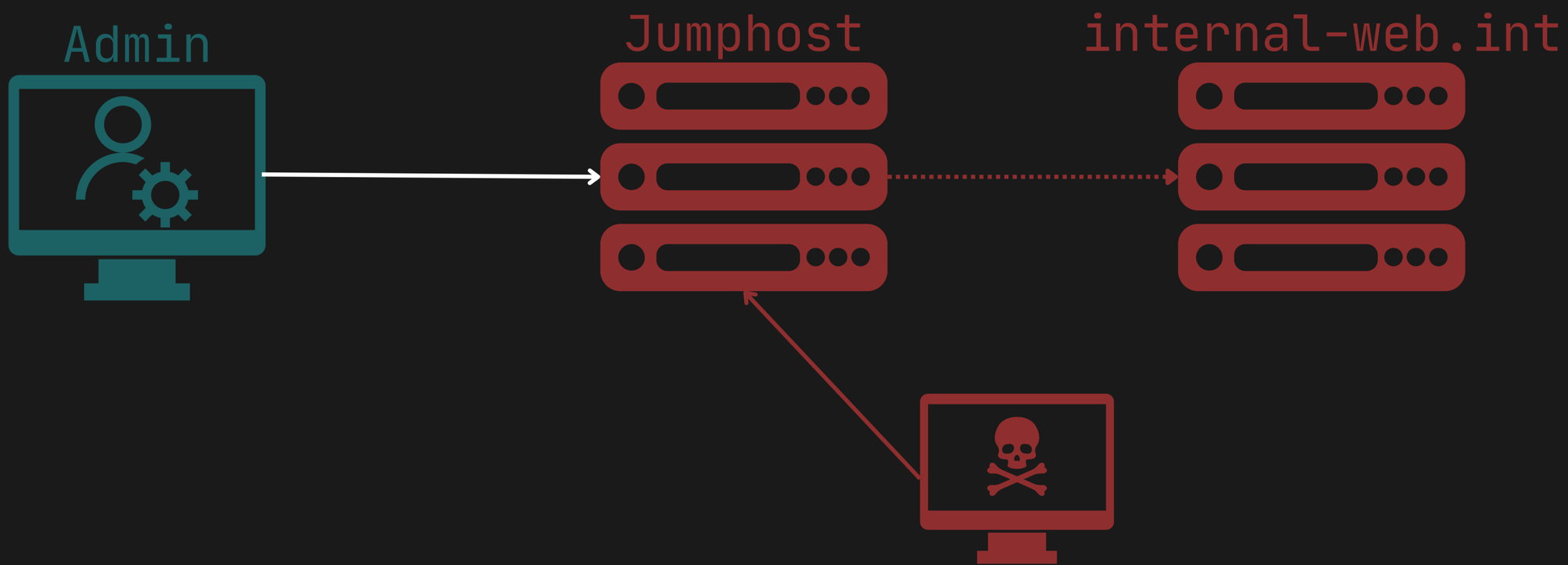
- Attacker looks for agent sockets: `lsof -U | grep agent`
  - SSH agent sockets are stored in `/tmp` and follow a pattern
  - In this case: `/tmp/ssh-ZzrT22wVr/agent.4145`

```
root@vuln-server:~# ssh-add -l
ssh-add -l
Could not open a connection to your authentication agent.
root@vuln-server:~# lsof -U | grep agent
lsof -U | grep agent
sshd      4145           admin   11u  unix 0x000000007f0fda18      0t0 46998936 /tmp/ssh-ZzrtT2ZwVr/agent.4145 type=STREAM
root@vuln-server:~# export | grep SSH_AUTH_SOCK
export | grep SSH_AUTH_SOCK
root@vuln-server:~# export SSH_AUTH_SOCK=/tmp/ssh-ZzrtT2ZwVr/agent.4145
export SSH_AUTH_SOCK=/tmp/ssh-ZzrtT2ZwVr/agent.4145
```



- Set path to agent file as the environment variable $SSH_AUTH_SOCK
  - ????
  - Profit?
  - Yes really it’s that easy

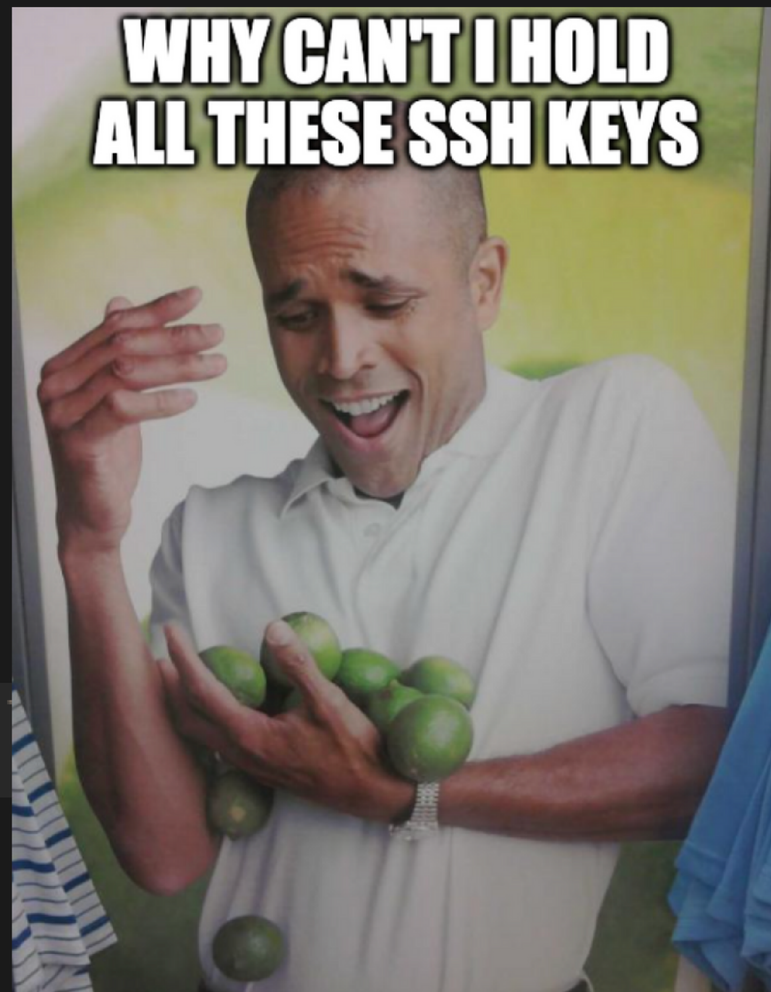
```
root@vuln-server:~# ssh-add -l
ssh-add -l
Could not open a connection to your authentication agent.
root@vuln-server:~# lsof -U | grep agent
lsof -U | grep agent
sshd      4145            admin   11u  unix 0x000000007f0fda18      0t0 46998936 /tmp/ssh-ZzrtT2ZwVr/agent.4145 type=STREAM
root@vuln-server:~# export | grep SSH_AUTH_SOCK
export | grep SSH_AUTH_SOCK
root@vuln-server:~# export SSH_AUTH_SOCK=/tmp/ssh-ZzrtT2ZwVr/agent.4145
export SSH_AUTH_SOCK=/tmp/ssh-ZzrtT2ZwVr/agent.4145
root@vuln-server:~# ssh-add -l
ssh-add -l
4096 SHA256:llb6AquVVMuenl0tZeOfnCk9GilnVA59YjGMfIzCvhM smores@campfire (RSA)
4096 SHA256:eUzEJijonWZJBnioyLCxpUTfIwjjzn5oC9VaKdovUKI smores@campfire (RSA)
```



What locks do the keys unlock?

1. Crack known host hashes
   a. SSH keeps an log of host you've connected to but it's hashed...
   b. Luckily hashes can be cracked
2. Check history file
   a. Where have they SSH'd to before?
3. Brute force
   a. Just try to connect to everything in the network with a dumb bash script (and don't tell your SOC for added "fun"... Oh also let it run overnight)

```
PS C:\Users\Graham\Documents\tools\hashcat-6.2.6> .\hashcat.exe -m 160 --hex-salt .\converted_known_hosts
a35cae4cf8852f18d698339759502da7df71a541:6653fb288689e102fcb3cf80c3a589de267b8b60:192.168.1.3
0ee1f40c134d1d6bb3c137091b6dbd7a672a15cf:f1aaabdc8234c0b330c1f12f6dcf91c554c15fa9:192.168.1.2
```



Agent Forwarding TLDR;

- Helpful for admins, introduces some risk (probably low or MAYBE medium) depending on your “tHrEaT mOdEl”
- Keys are not “exportable” so you’re not really “compromising” the key itself.
- If you’re looking to do this please read:
  - https://grahamhelton.com/blog/ssh_agent/

File: **ssh_agent.md**

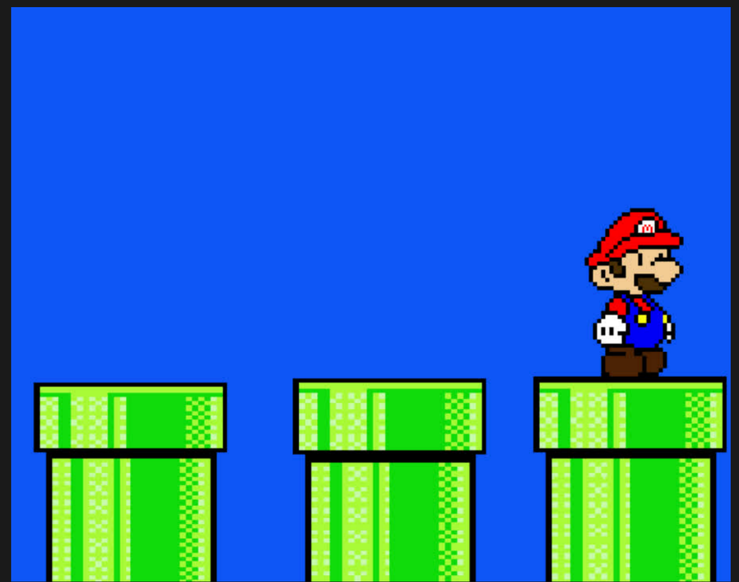
```
---
title: "Zero Effort Private Key Compromise: Abusing
date: 2023-08-18
tags: ["redteam","security","linux"]
images: https://grahamhelton.com/
draft: false
description: "A walkthrough of compromising private
---
# Intro
The other day I was looking through some videos I ha
by Hal Pomeranz. About halfway though the video I ha
t I hadn't seen before. Using that training as a jum
a walk through of what I learned going down that rab
# What is SSH Agent?
For starters, we need to understand a little about w
ivate keys. It's similar to the concept of single si
sh-add <private_key_file>`. These keys can then be l
to re-enter the password. This is useful for both hu
se your private keys from the machine you're connect

1. *Admin* is a server administrator who is responsi
ns do.
2. *Admin* Utilizes the `ssh-agent` utility to help
e with the `ssh-add <privatekey>` command. (Note: yo
    ![](/Pasted-image-20230816005938.png)
3. Now, if *admin* needs to connect to say, a dns se
ey file, she can simply type `ssh root@pihole` and t
```

moving on...

Local Port Forwarding (-L)

- Use cases:
  - Connect to an internal service (listening on localhost)
  - Reach a port blocked by a firewall

1. Establishes an SSH tunnel
2. Traffic sent to a port on your local machine is tunneled through SSH
3. Traffic arrives at destination port on the other side of the tunnel



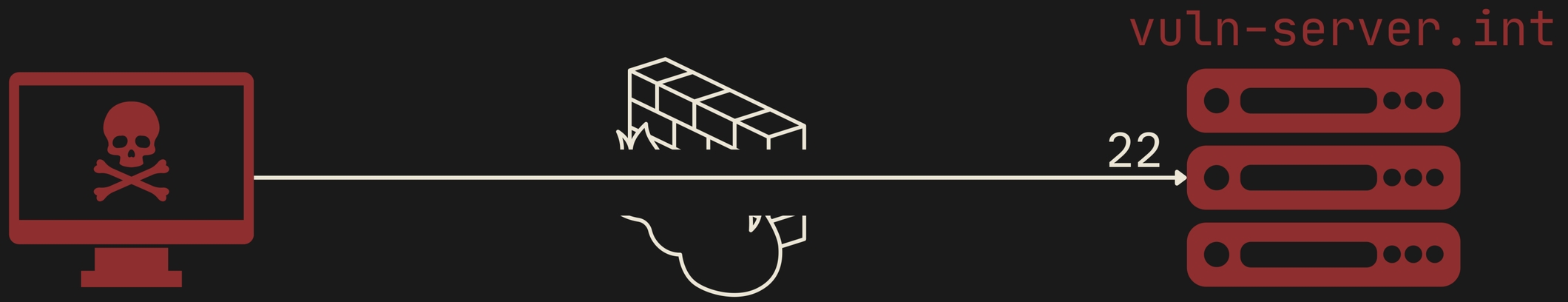
Problem #1

- Your target is a homegrown credential manager service hosted on port 1337
- Service is listening on vuln-server.int port 1337
- Firewall only allowing traffic on port 22

vuln-server.int

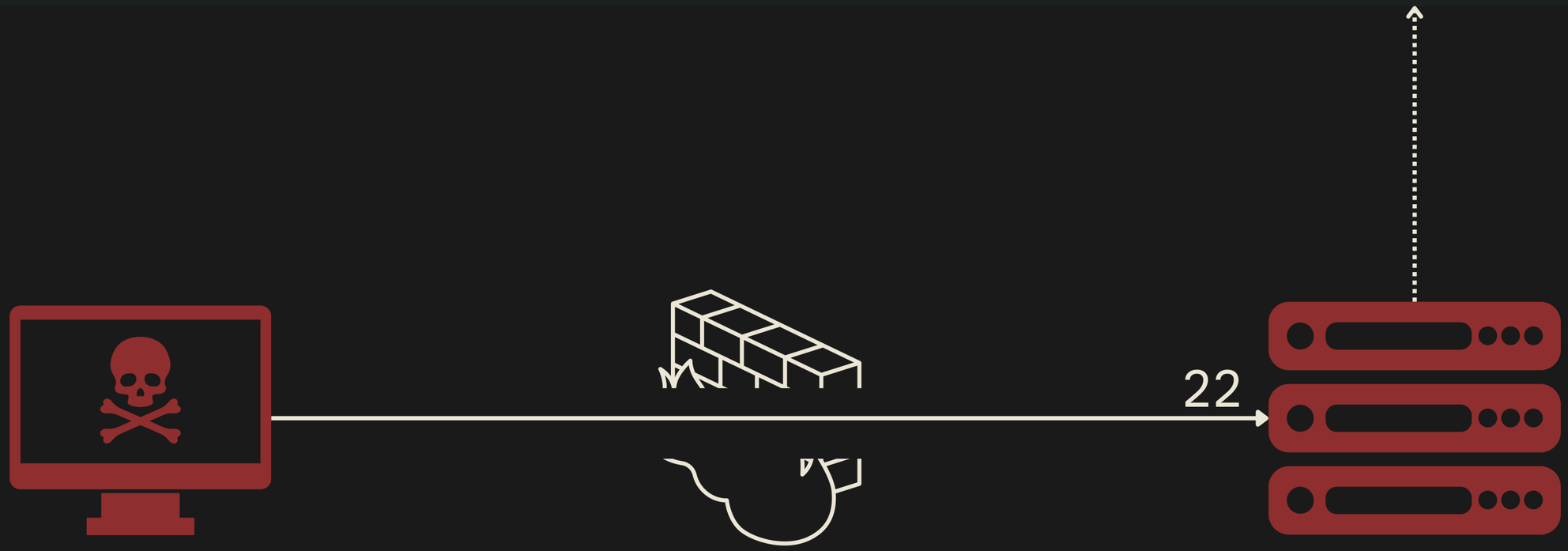
Problem #1

- You compromise the machine via ✨ sick hax ✨ and have SSH access
- Firewall allows SSH traffic



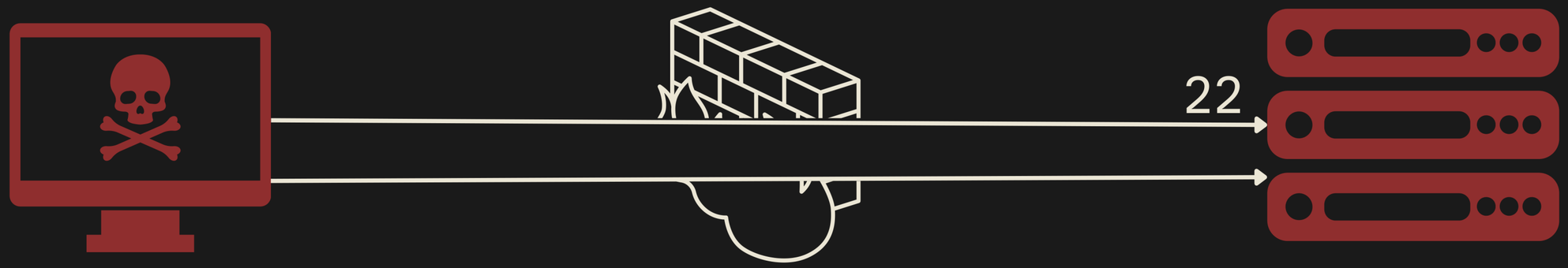
Problem #1

```
bsides@nashville$ lsof -i | grep LISTEN
vuln_serv 5435 ubuntu    3u  IPv6  25777      0t0  TCP *:1337 (LISTEN)
bsides@nashville$ curl -X POST http://127.0.0.1:1337/ -H "Content-Type: application/json" -d '{"text": "hello world!"}'
Received message: hello world!
bsides@nashville$
```



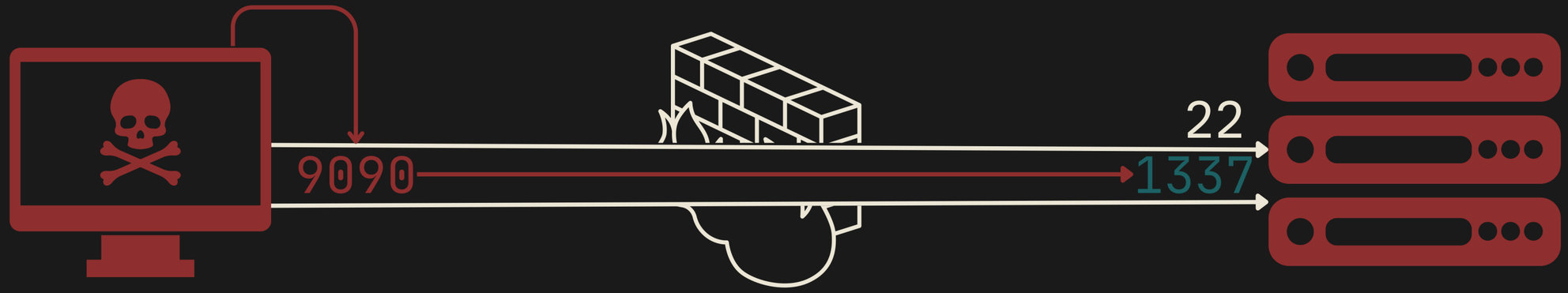
Solution Local Port Forwarding (-L)

```
ssh  \ # Run SSH
-i ~/.ssh/nashville.pem \ # Specify the key to connect to bsides@nashville
-N \ # Let SSH know you're not sending any commands
-f \ # Send SSH to the background
-L 9090:127.0.0.1:1337 \ # Tell SSH to forward a local port
ubuntu@54.226.94.73 # The server to connect to
```

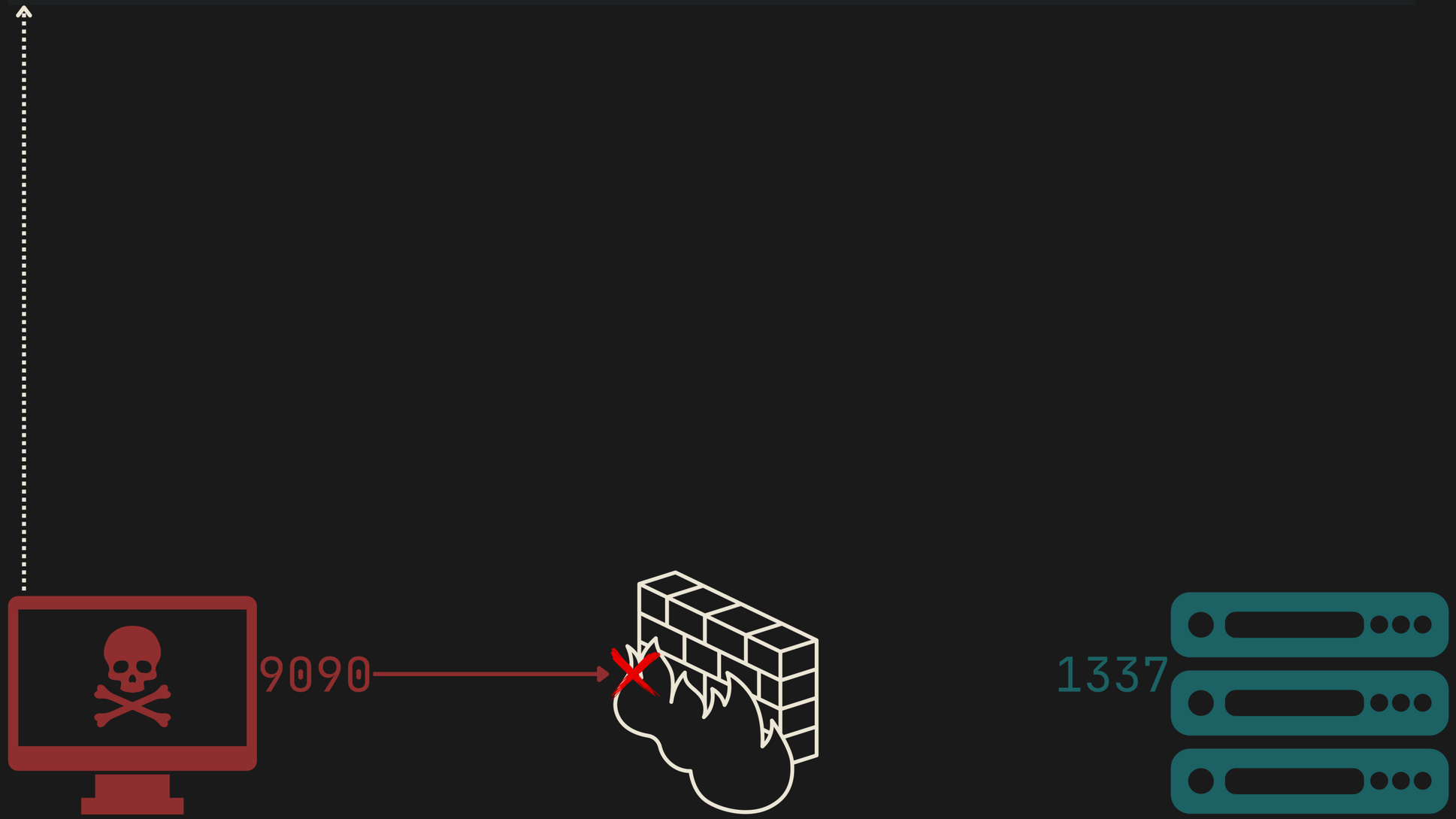


Solution Local Port Forwarding (-L)

```
ssh  \ # Run SSH
-i ~/.ssh/nashville.pem \ # Specify the key to connect to bsides@nashville
-N \ # Let SSH know you're not sending any commands
-f \ # Send SSH to the background
-L 9090:127.0.0.1:1337 \ # Tell SSH to forward a local port
ubuntu@54.226.94.73 # The server to connect to
```

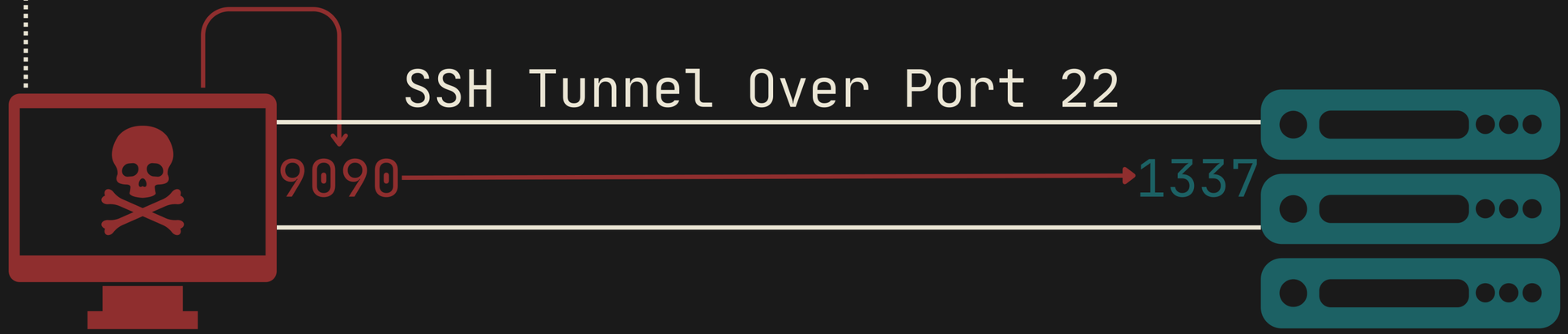


```
attacker@crashville$ curl -X POST http://54.226.94.73:9090/ \
> -H "Content-Type: application/json" \
> -d '{"text": "hello nashville"}' \
> -m 3
curl: (28) Connection timed out after 3002 milliseconds
```



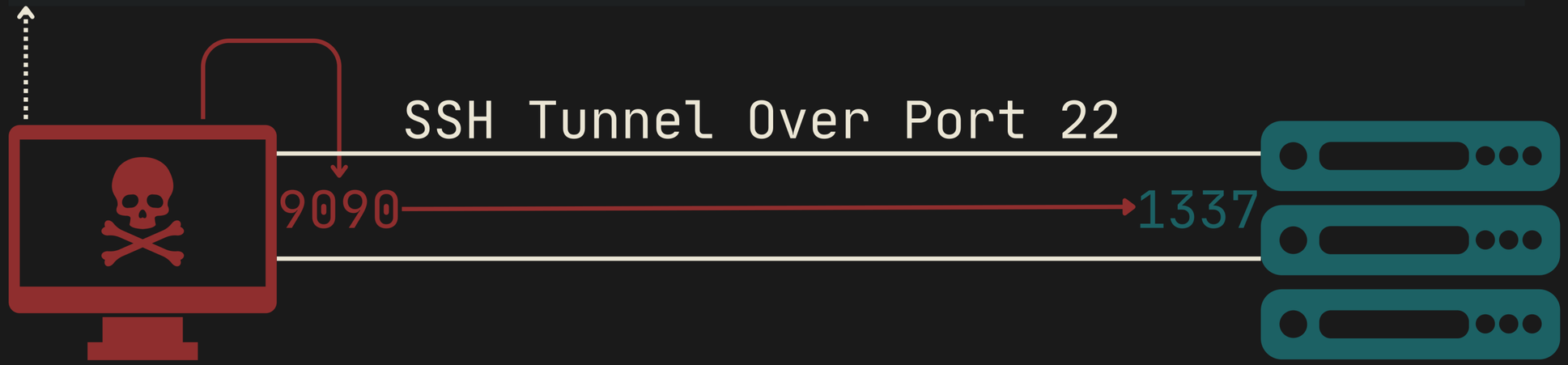
Source: https://www.docker.com/resources/what-container/

```
attacker@crashville$ curl -X POST http://54.226.94.73:9090/ \
> -H "Content-Type: application/json" \
> -d '{"text": "hello nashville"}' \
> -m 3
curl: (28) Connection timed out after 3002 milliseconds
attacker@crashville$ ssh -i ~/.ssh/nashville.pem \
> -N \
> -f \
> -L 9090:127.0.0.1:1337 \
> ubuntu@54.226.94.73
```



Source: https://www.docker.com/resources/what-container/

```
attacker@crashville$ curl -X POST http://54.226.94.73:9090/ \
> -H "Content-Type: application/json" \
> -d '{"text": "hello nashville"}' \
> -m 3
curl: (28) Connection timed out after 3002 milliseconds
attacker@crashville$ ssh -i ~/.ssh/nashville.pem \
> -N \
> -f \
> -L 9090:127.0.0.1:1337 \
> ubuntu@54.226.94.73
attacker@crashville$
attacker@crashville$ curl -X POST http://localhost:9090/ \
> -H "Content-Type: application/json" \
> -d '{"text": "hello nashville"}'
Received message: hello nashville
attacker@crashville$
```



Source: https://www.docker.com/resources/what-container/

Remote Port Forwarding (-R)

- Use cases:
  - Have traffic originate from target machine
  - Move deeper into a network
- Think of this as using the target as a proxy

Problem

- Target: internal-web.int internal HR website
- Problem: Blocked by firewall

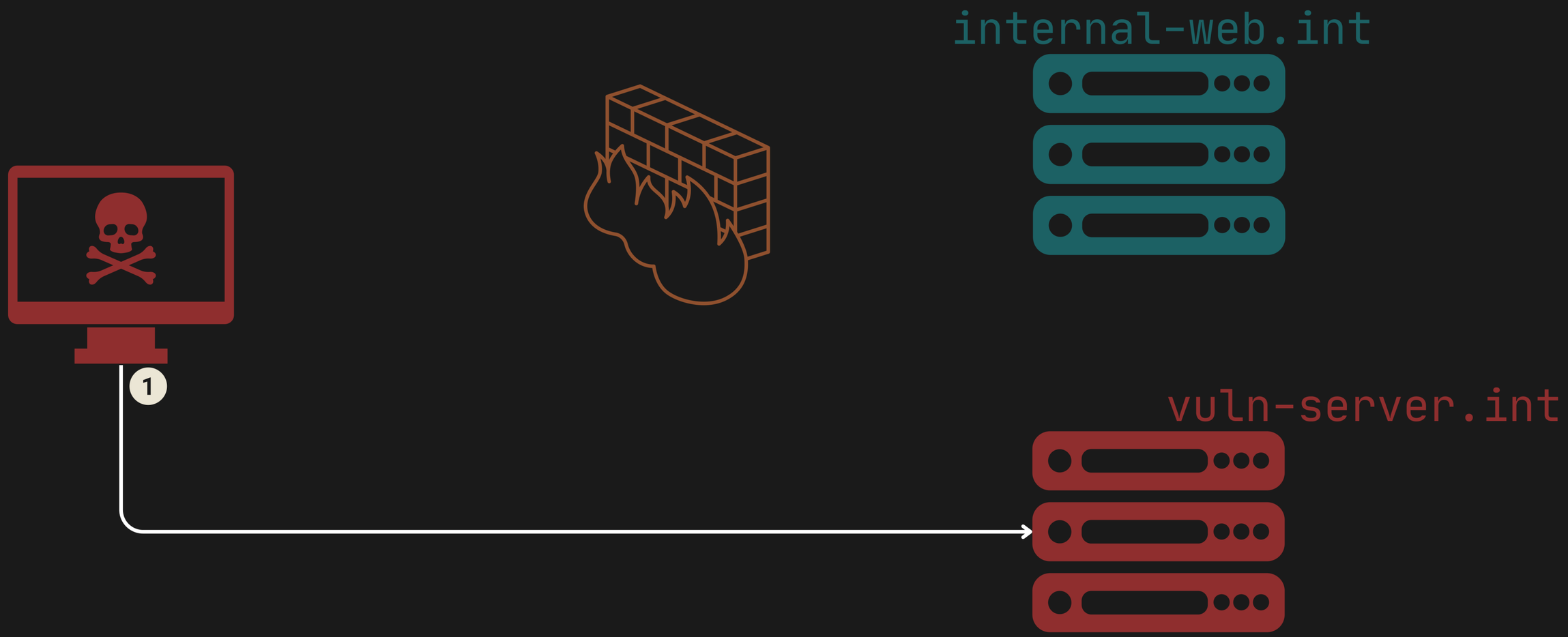
internal-web.int

vuln-server.int

Source: https://www.docker.com/resources/what-container/

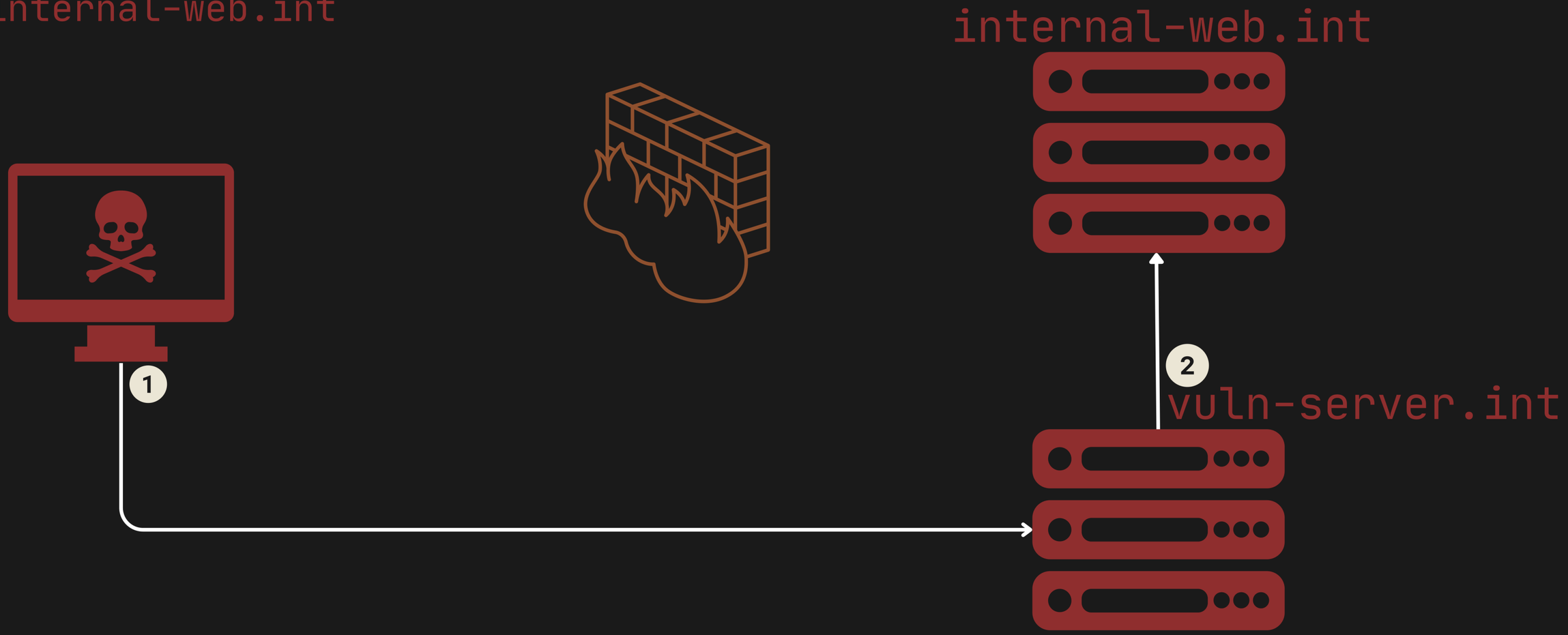
Solution

1. Compromise vuln-server.int (DMZ)



Solution

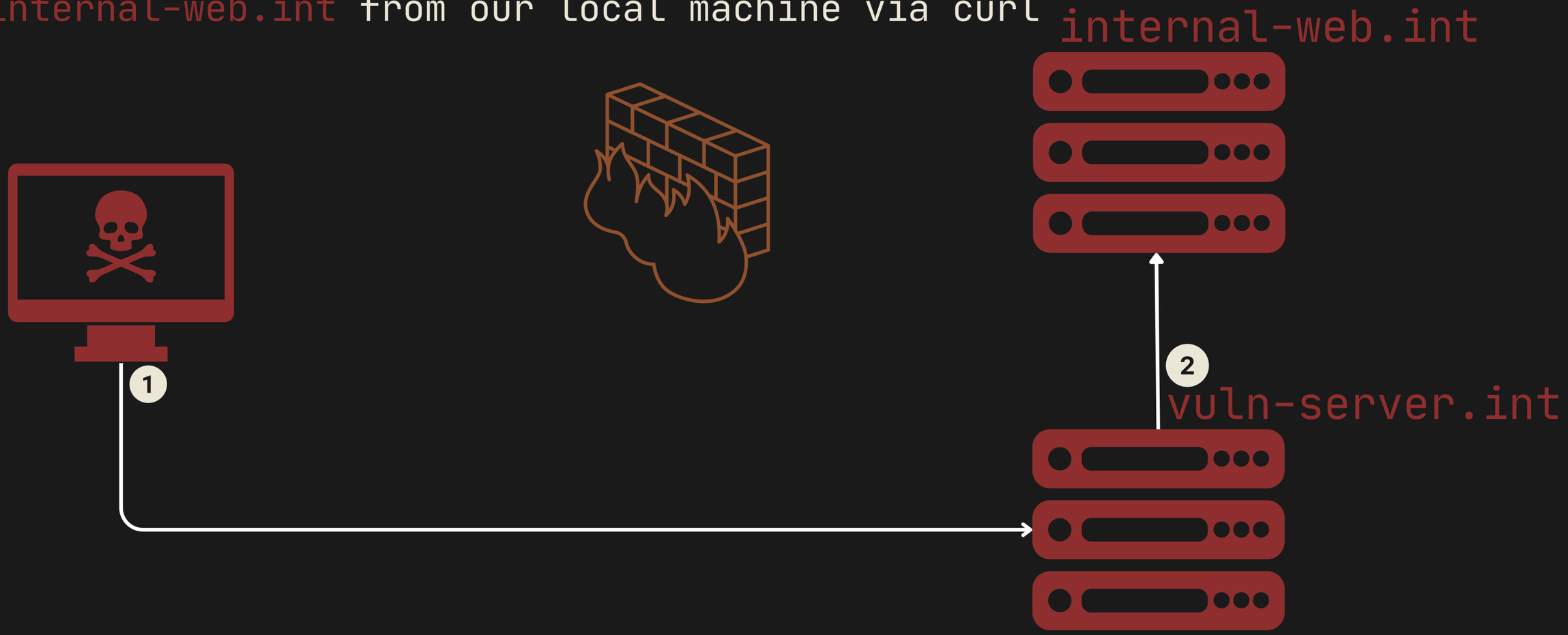
1. Compromise vuln-server.int (DMZ)
2. Utilize trusted network access to compromise the server internal-web.int



Source: https://www.docker.com/resources/what-container/

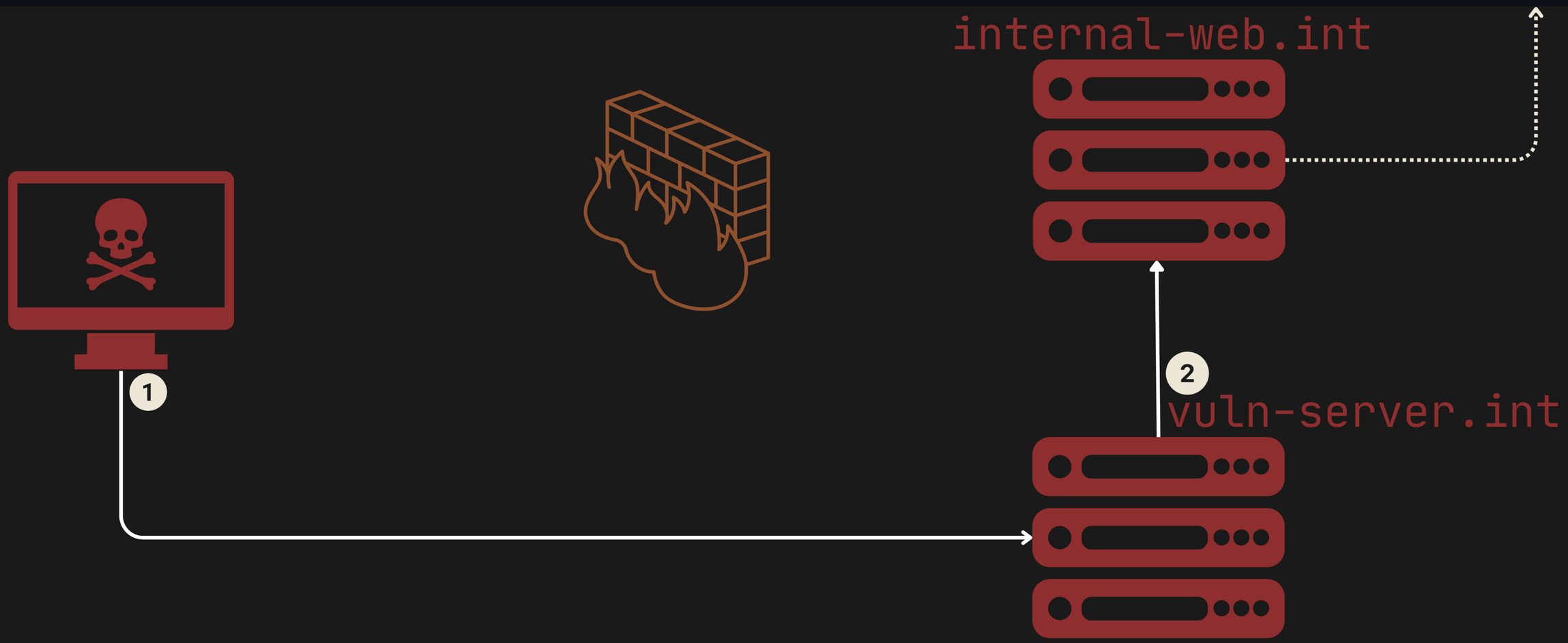
Objective

- We've compromised internal-web.int
- We want to interact with the HR website's API on internal-web.int from our local machine via curl

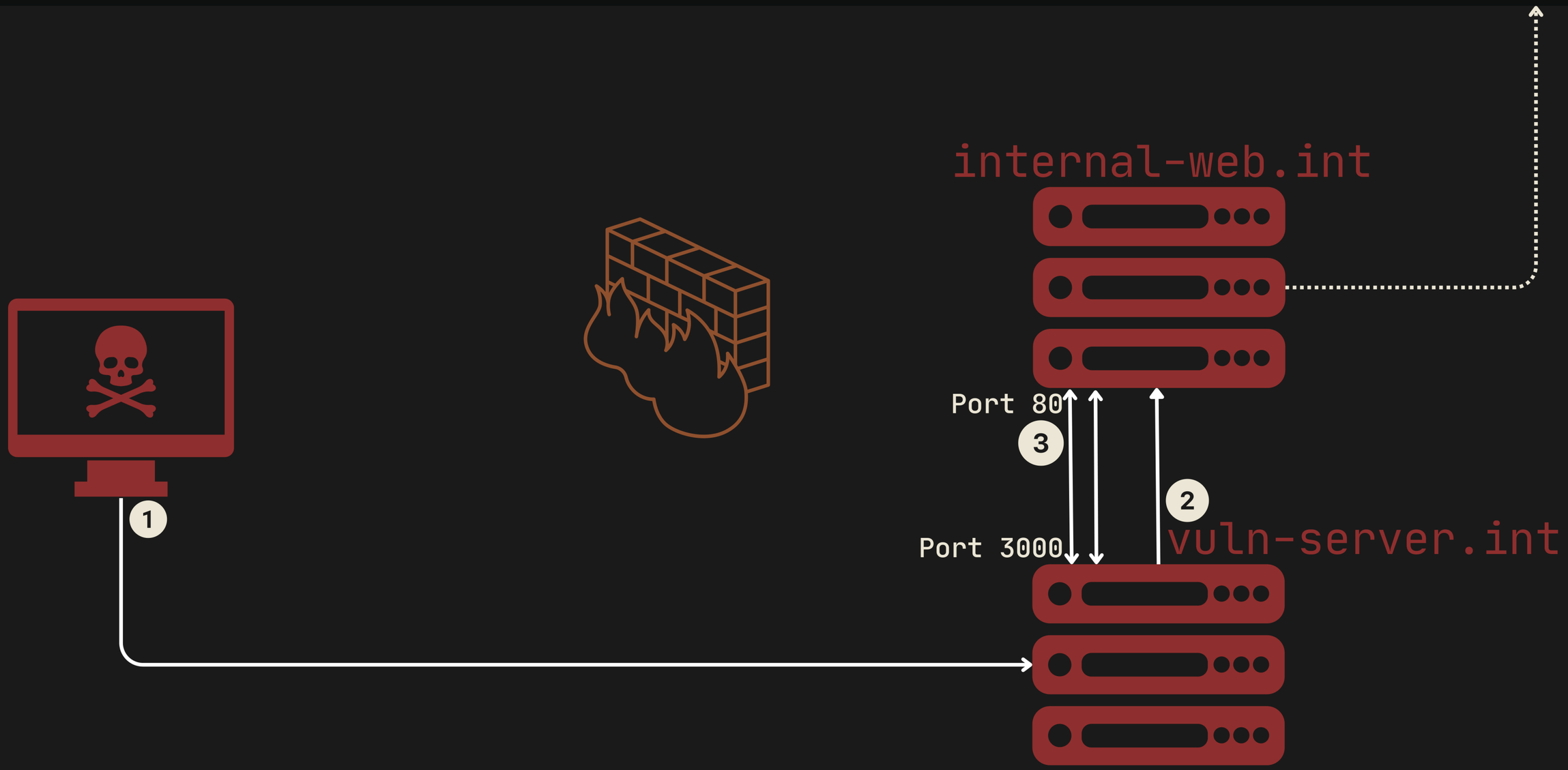


Source: https://www.docker.com/resources/what-container/

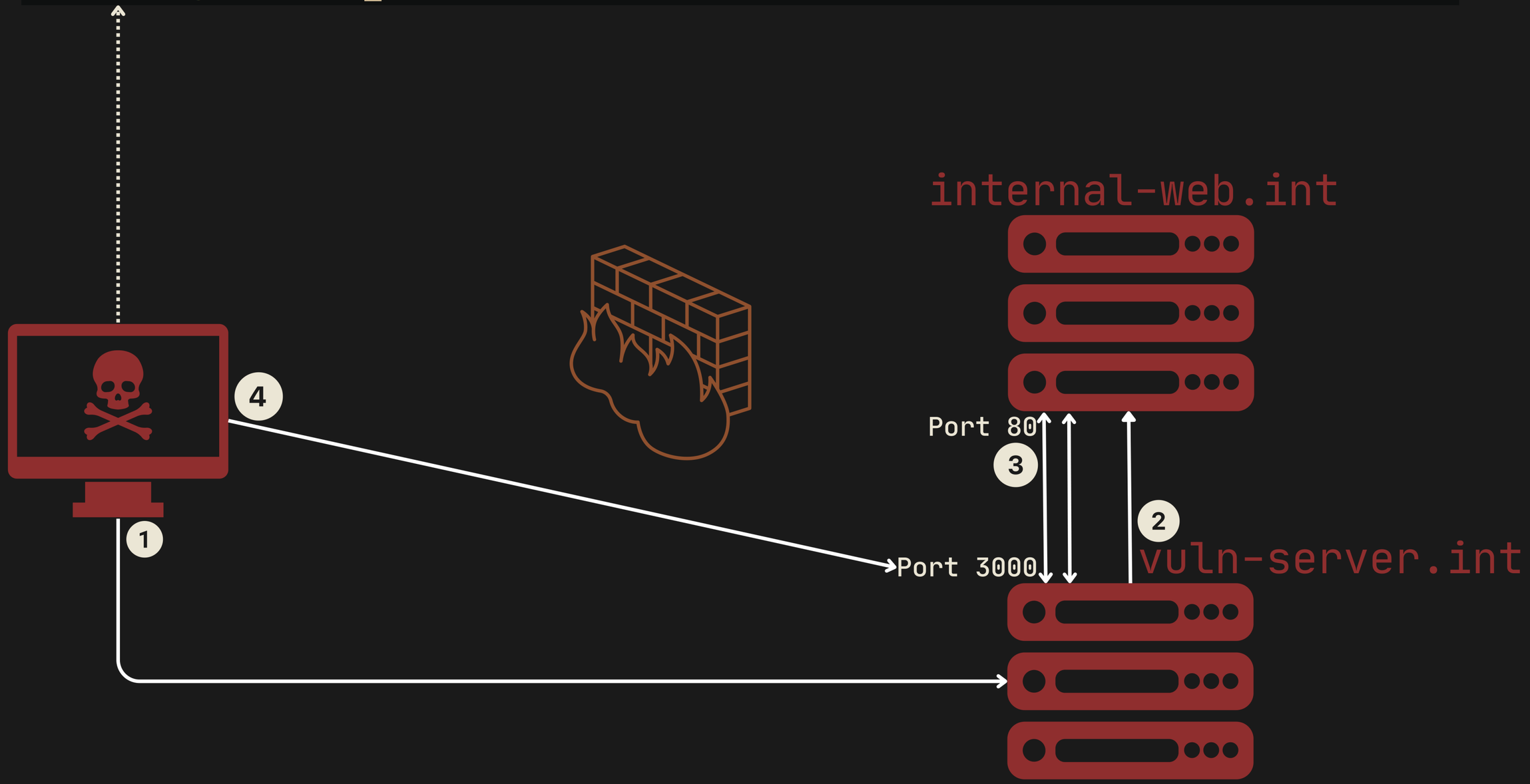
```
ssh \ # Run SSH
-i ~/.ssh/vun-server-key \ # Specifiy the key to connect to vulnserver.int
-N \ # Let SSH know you're not sending any commands
-f \ # Send SSH into the background
-R 3000:127.0.0.1:80 \ # Tell SSH to bind the remote port 3000 to the local port of 80
root@vuln-server.int # Tell SSH to connect to vuln-server.int
```



```
root@internal-web(192.168.1.185)# ssh -N -f -R 3000:127.0.0.1:80 root@vuln-server.int
root@vuln-server.int's password:
root@internal-web(192.168.1.185)#
```

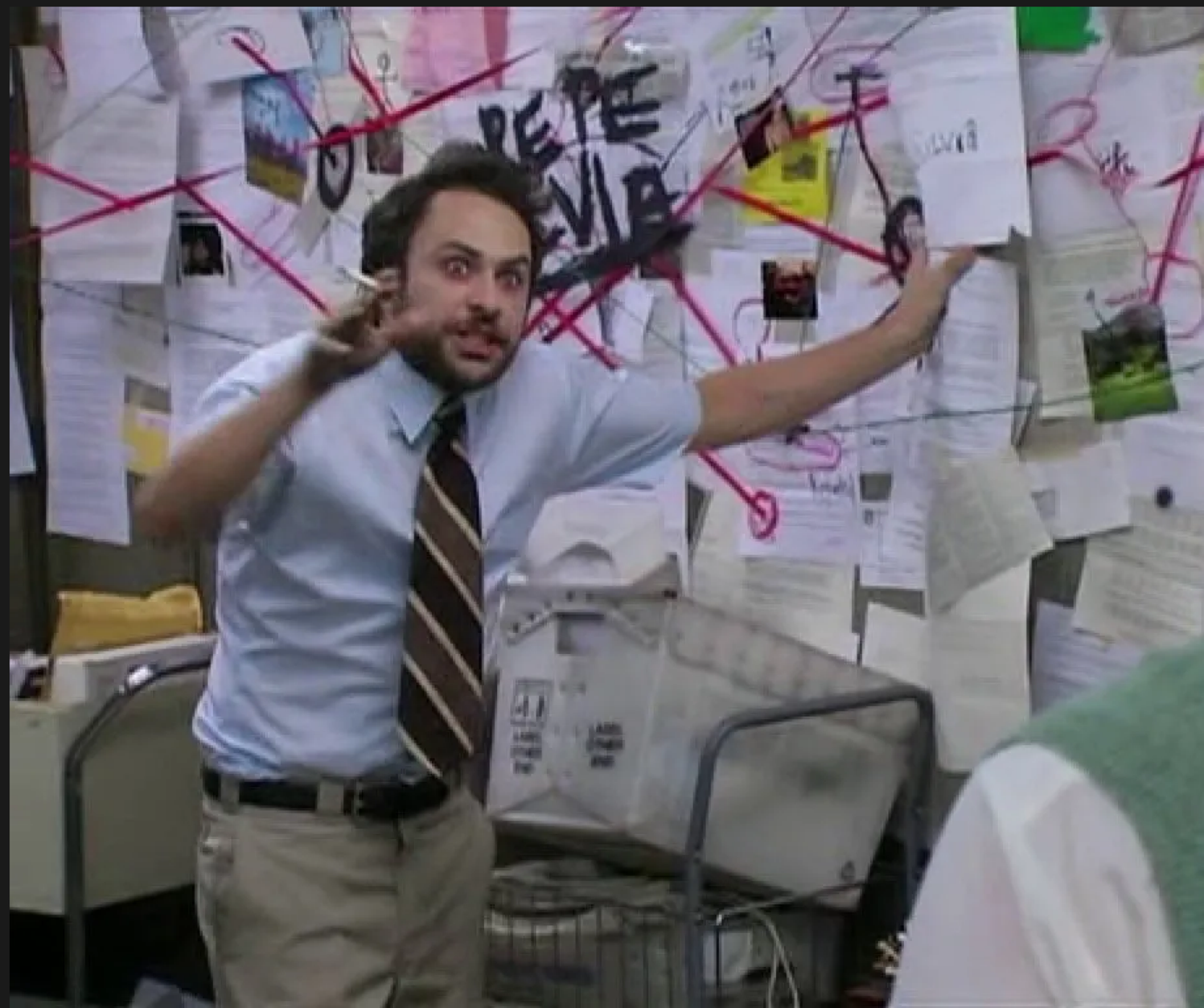


```
smores@campfire:~ $ curl -s vuln-server.int:3000 | grep Welcome
<title>Welcome to nginx!</title>
<h1>Welcome to nginx! This server is hosted on internal-web.int (192.168.1.185)</h1>
smores@campfire:~ $
```



I’m no longer feeling port 22

- In fact, I’m scared and want to go home
- You still have to contend with firewalls



moving on...

Dynamic Port Forwarding (-D)

- Use case: We want **all** of our traffic to be sent to a remote host first
  - Make traffic look like it’s coming from a remote host
  - Bypass firewalls
  - Blend in to “normal”
- Requires some pre-setup using proxychains

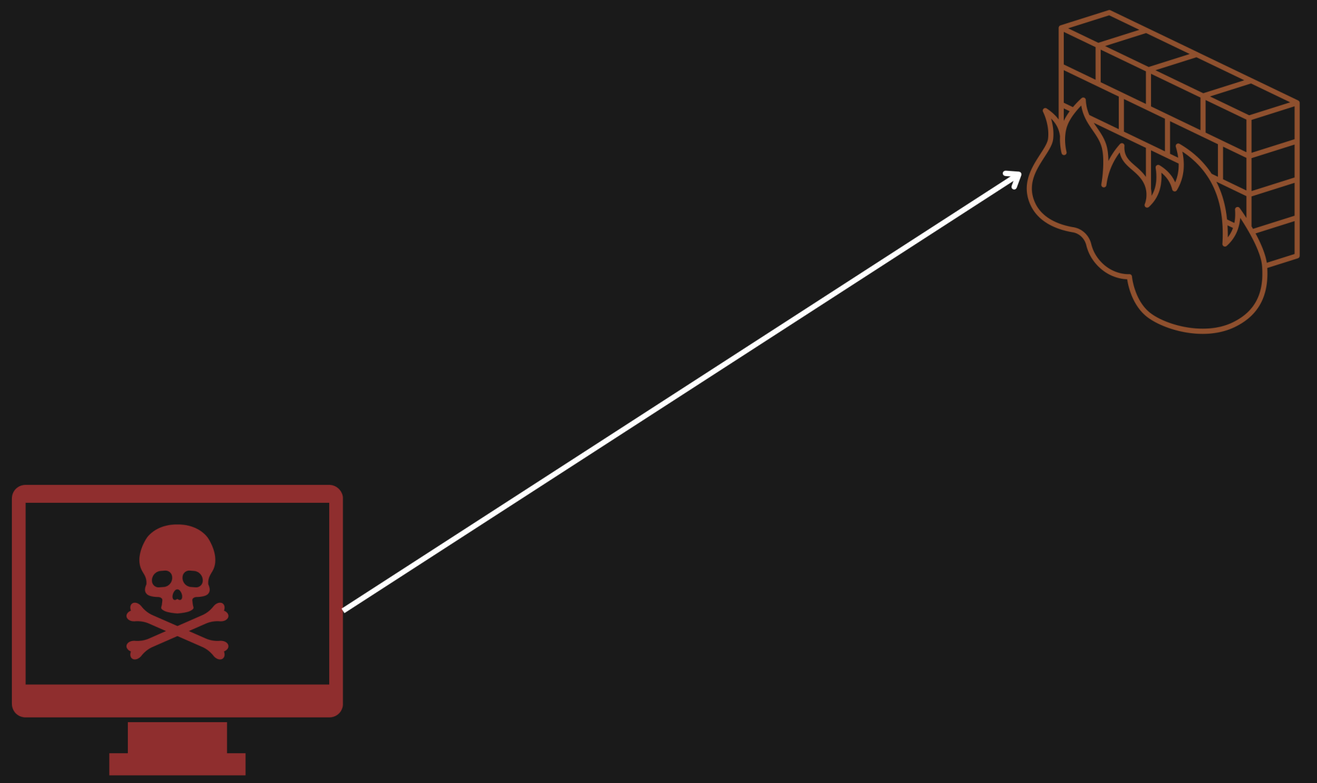
Dynamic Port Forwarding (-D)

1. Edit `/etc/proxychains.conf`
2. Add the following line:
   `socks5 127.0.0.1 8080`

- **Socks5**: Tells proxychains to use socks version 5
- **127.0.0.1**: Tells proxychains to use our localhost
- **8080**: Is the port we will use for our dynamic forward. This must match the port you specify with -D in your SSH command.

Dynamic Port Forwarding (-D)

- Target: internal-web.int internal HR website
- Problem: Blocked by firewall



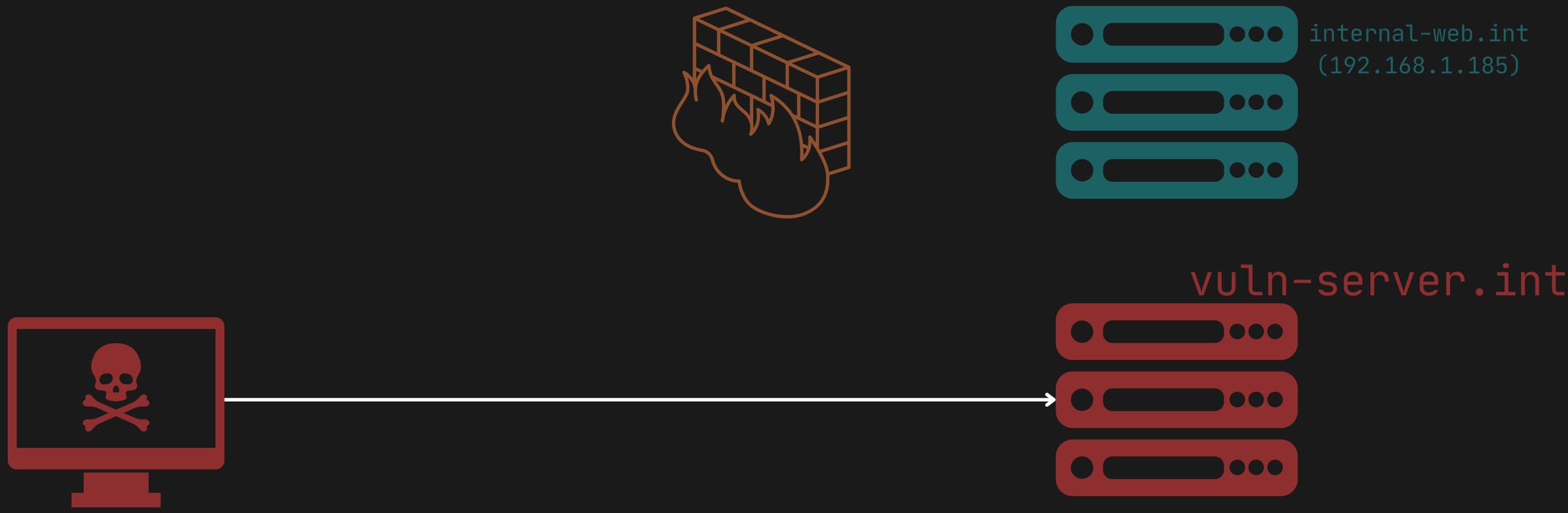
internal-web.int
(192.168.1.185)

vuln-server.int

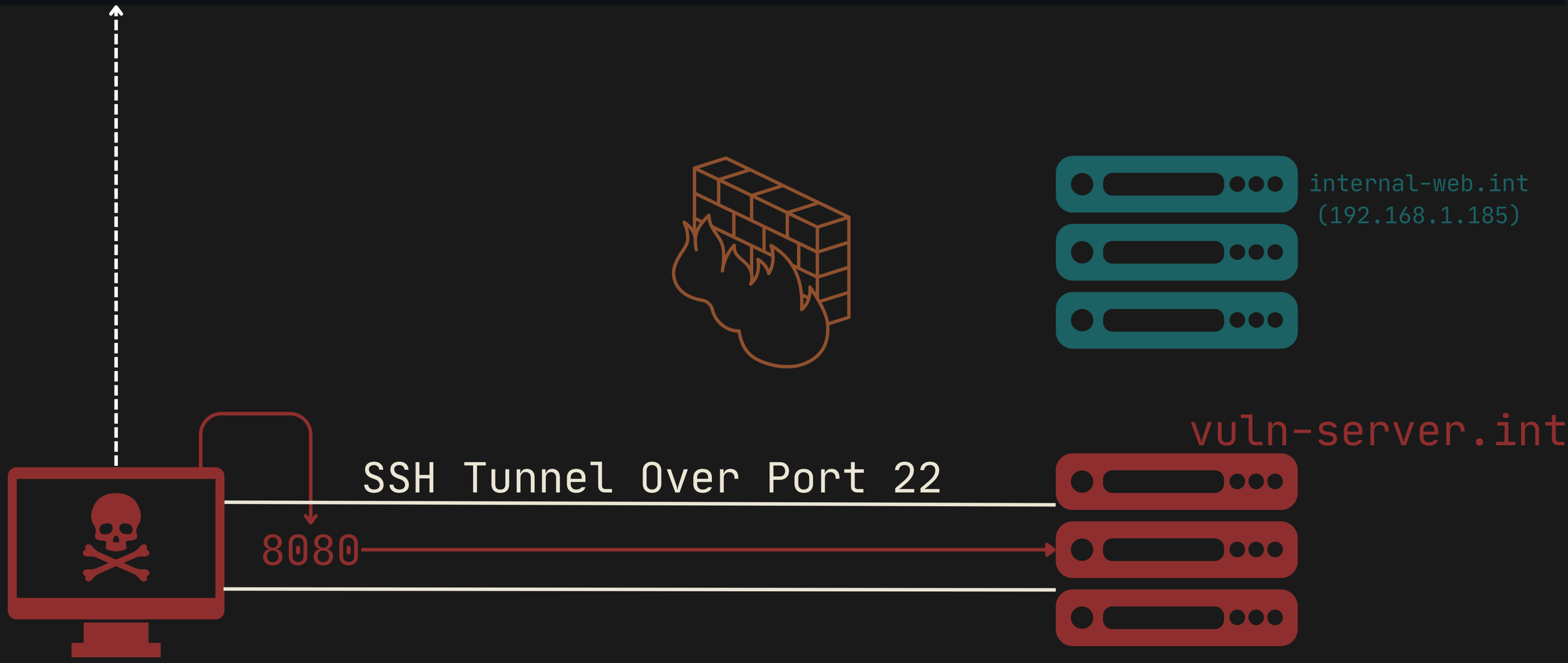
Source: https://www.docker.com/resources/what-container/

Solution

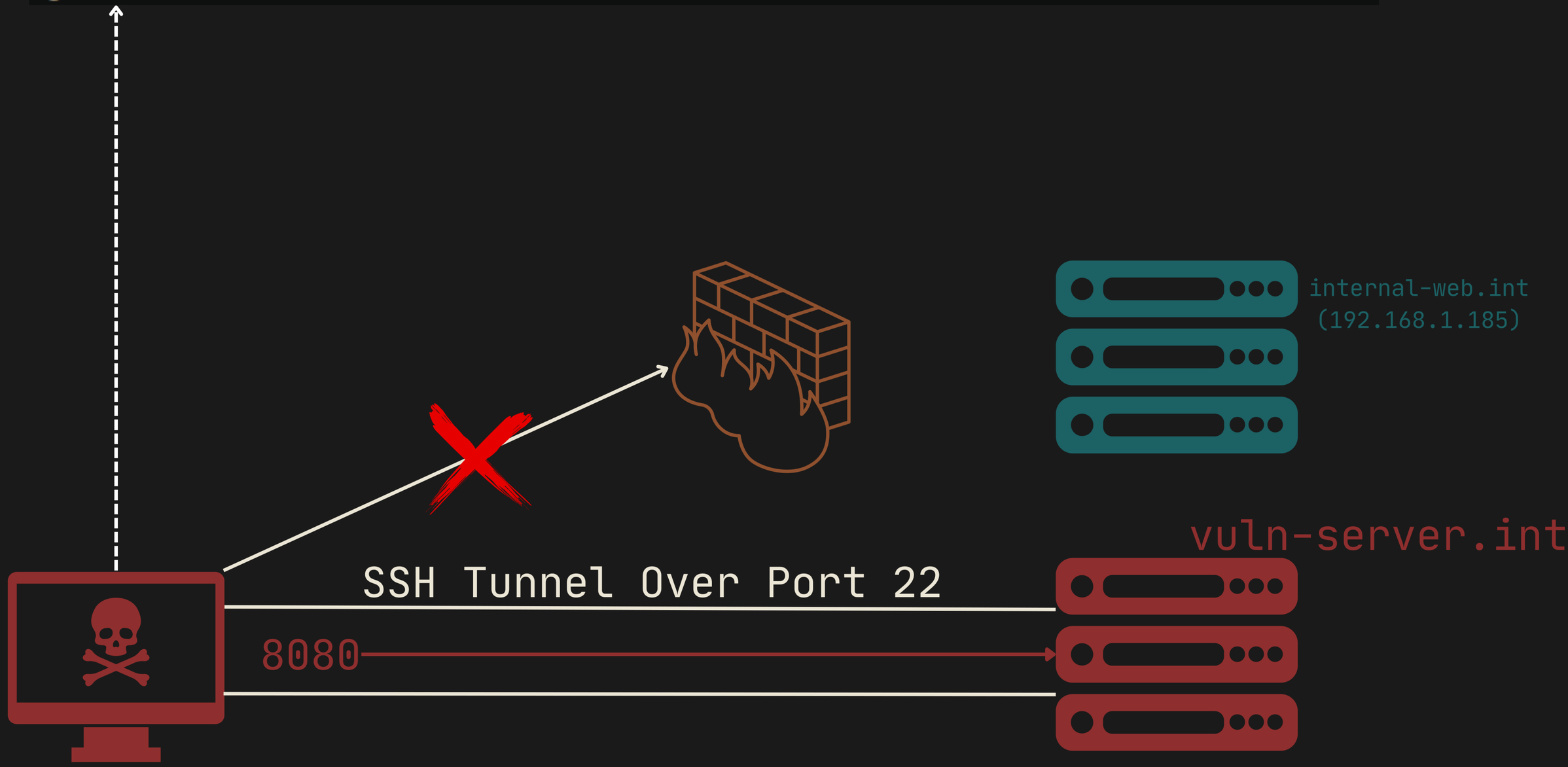
1. Compromise vuln-server.int (DMZ)



```
ssh \ # Run SSH
-N \ # Let SSH know we're not sending any commands
-f \ # Send SSH to the background
-D 8080 \ # Tell SSH to create a dynamic local port to send our traffic through
root@vuln-server.int # The server we wish for our traffic to be sent to
```

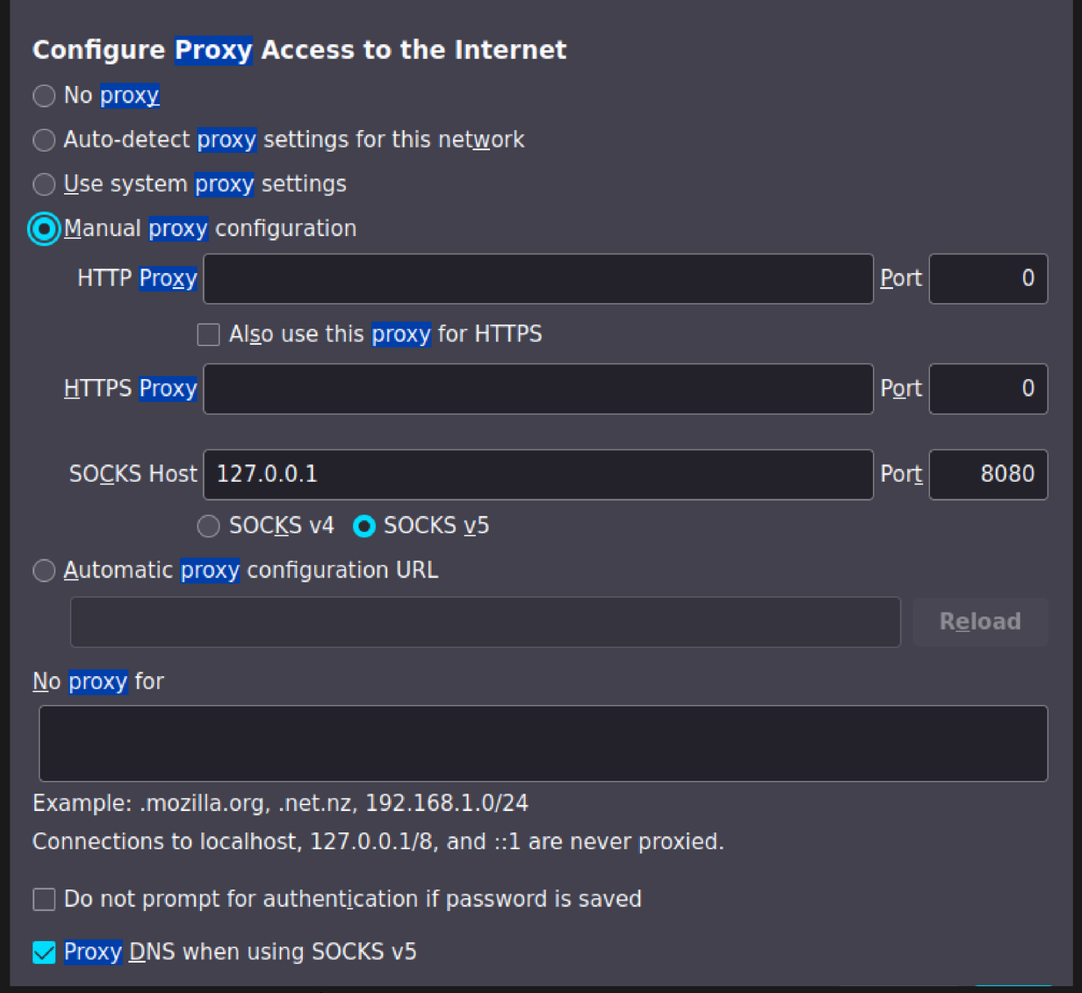


```
smores@campfire:~ $ ssh -N -f -D 8080 root@vuln-server.int
root@vuln-server.int's password:
smores@campfire:~ $ curl internal-web.int
^C
```

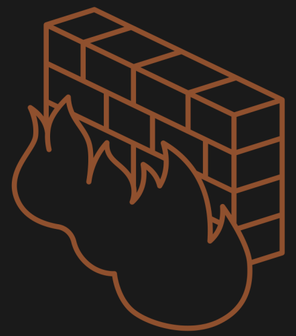


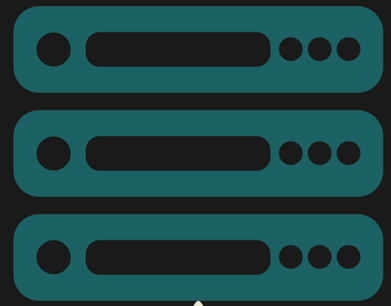
```
smores@campfire:~ $ ssh -N -f -D 8080 root@vuln-server.int
root@vuln-server.int's password:
smores@campfire:~ $ curl internal-web.int
^C
smores@campfire:~ $ proxychains curl 192.168.1.185
ProxyChains-3.1 (http://proxychains.sf.net)
|D-chain|-<>-127.0.0.1:8080-<><>-192.168.1.185:80-<><>-OK
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

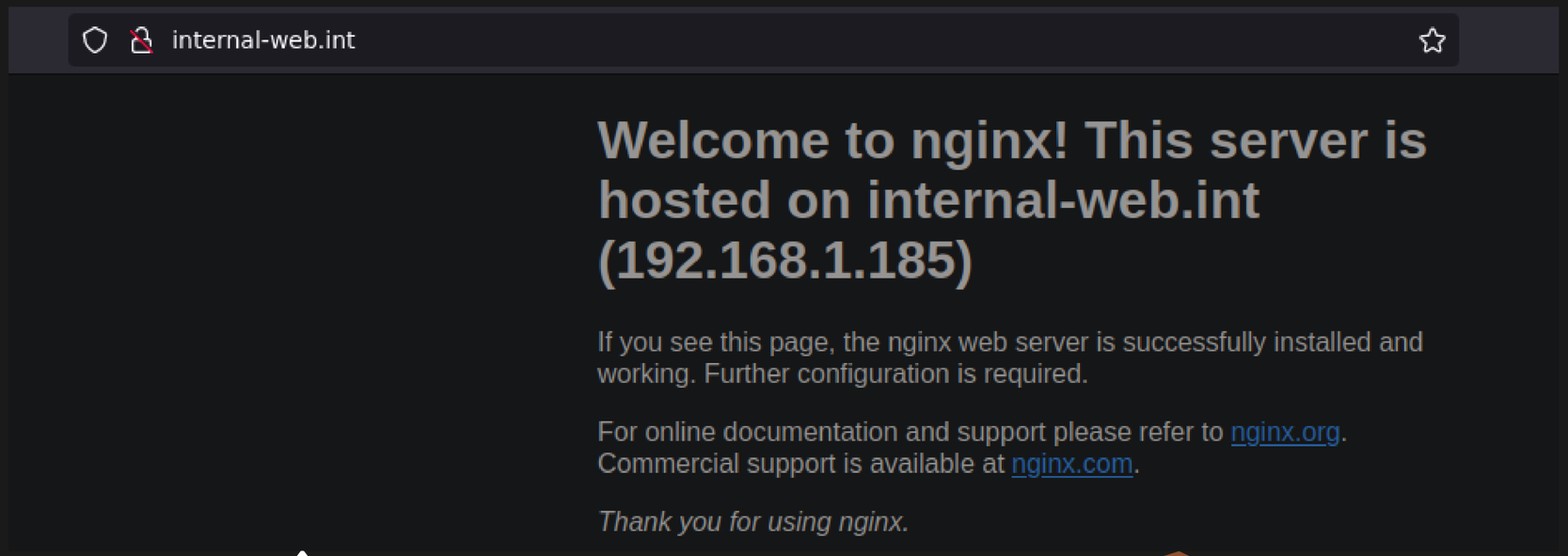
What if we want to browse the website like a normal human being and not a massive nerd using curl...



- Firefox: Network Settings
  - Proxy Settings
    - Manual Proxy Configuration ✓
    - SOCKS Host: 127.0.0.1
    - Port: 8080
    - Proxy DNS ✓






internal-web.int
Welcome to nginx! This server is hosted on internal-web.int (192.168.1.185)
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.
Thank you for using nginx.


internal-web.int
(192.168.1.185)
vuln-server.int
SSH Tunnel Over Port 22
8080

Dynamic Port Forwarding (-D) TLDR:

- DNS can get even more funky depending on the network, start here for troubleshooting
- Useful for proxying through burpsuite

Port bending

- What if (for whatever VERY LEGAL reason) you wanted to redirect traffic on port 1337 to port 8080
  - Interesting MiTM opportunities, especially if unencrypted communications
  - Inject traffic
  - Scapy
  - ~~Take a hammer to a multi billion dollar company's new product~~
- iptables is your friend

Port bending

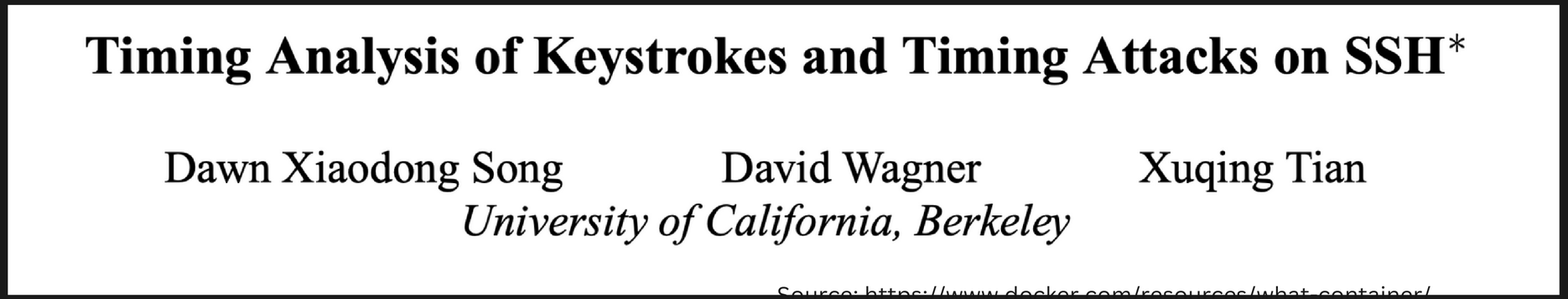
- `iptables -t nat -I PREROUTING -p tcp --dport 1337 -j REDIRECT --to-ports 8080`
- `iptables -t nat -I OUTPUT -p tcp -o lo --dport 1337 -j REDIRECT --to-ports 8080`
- Only nerds memorize iptables switches
  - And tweet about them...

Port bending

- Netcat connection to 1337 gets redirected to 8080

Ghosts aren’t real so they can’t hurt me, right? RIGHT?

- SSH is encrypted
- But every keypress sends a packet of data
- Using keystroke timing, you can glean data from encrypted traffic
- Purely theoretical and not seen in the wild 🤔🤔🤔
- Mitigation added for this send phantom keystrokes at random intervals. Very chatty on a network: https://man.openbsd.org/ssh_config.5#ObscureKeystrokeTiming



Timing Analysis of Keystrokes and Timing Attacks on SSH*

Dawn Xiaodong Song David Wagner Xuqing Tian

University of California, Berkeley

Source: https://www.docker.com/resources/what-container/

exit(0)

- Q/A (Time permitting)
- GrahamHelton.com/links
- https://grahamhelton.com/blog/ssh-cheatsheet
- https://grahamhelton.com/blog/ssh-agent