

PRELUDE

〈ELEVATOR MUSIC〉

WHOAMI?

In brief. Im Tree, a big nerd

ESTABLISH CREDIBILITY

- Started breaking computers when I was about 12
- Somehow ended up with a job writing a software model checker at Victoria University
- Started working in cybersecurity (for Aura Information Security)
- And have been ever since for the last 5 years
- Moved from pentesting to infrastructure
- Built a lot of random crap during that time

ORDER OF BUSINESS

RSSH -> SSH -> WAG -> Wireguard & eBPF

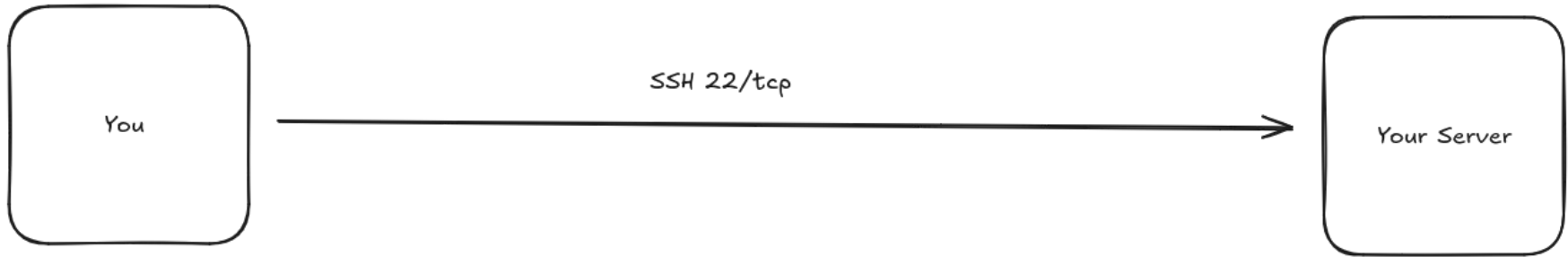
(many tangents)

REVERSE SSH (RSSH)

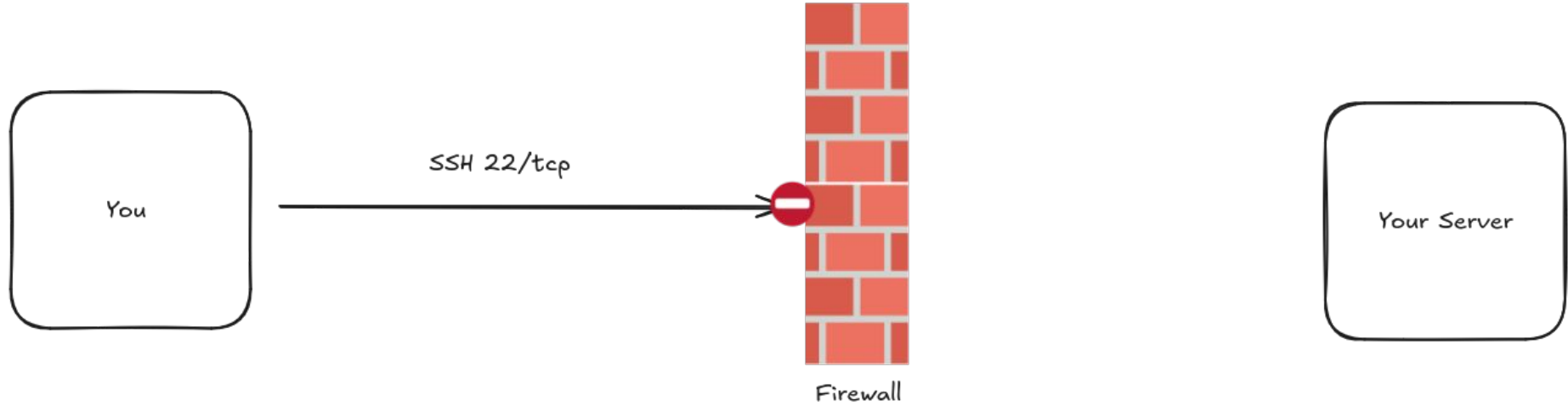
RSSH- REASON FOR EXISTING

I wanted to do SSH but in reverse 😎

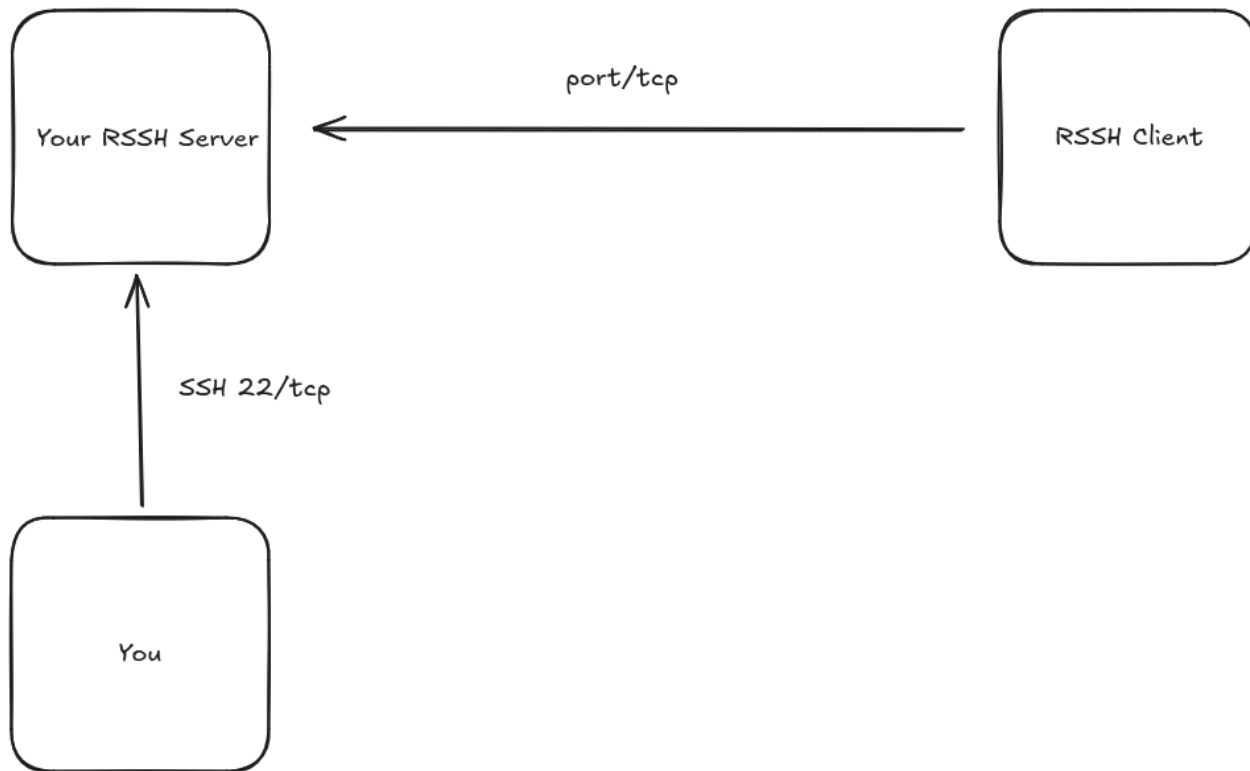
RSSH VS SSH



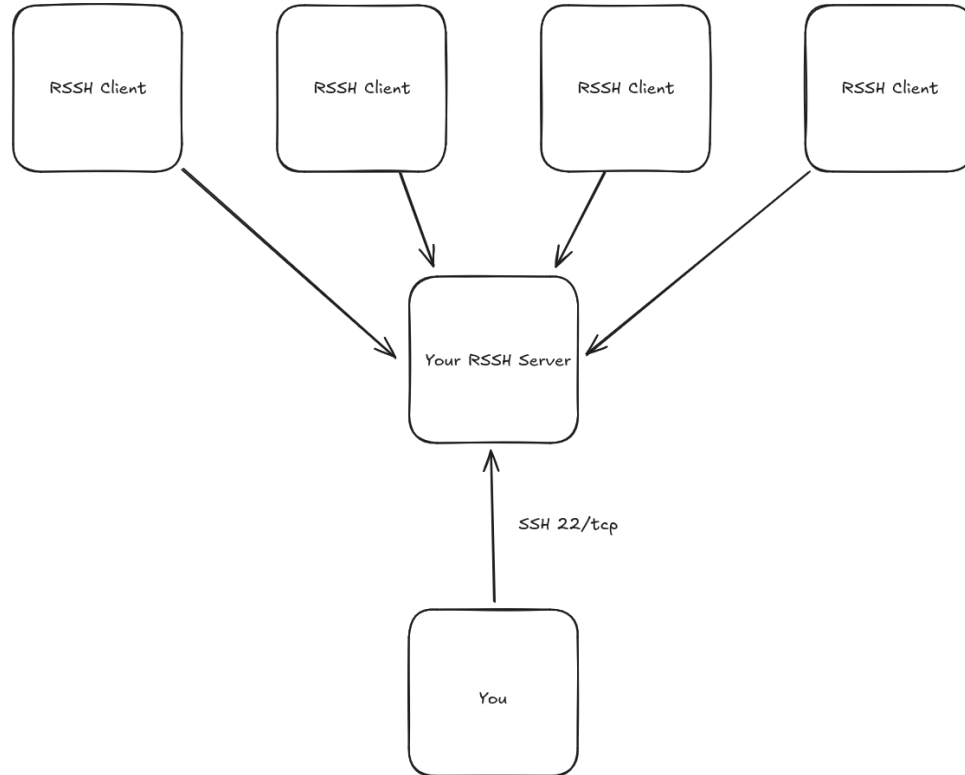
RSSH VS SSH



RSSH VS SSH



RSSH VS SSH



RSSH- FEATURES, FUNCTIONS AND FUN

- Single Golang Binary
- Implements the entire feature set of the OpenSSH toolset
 - TCP Forwarding
 - Tun (VPN)
 - File Copying
- No need to rely on SSH being installed on the target system

https://github.com/NHAS/reverse_ssh

RSSH- FEATURES, FUNCTIONS AND FUN

RSSH also:

- Supports multiple network transports
 - HTTP
 - WebSockets
 - TLS
- Windows? We do that too
- Multi-arch
- VT100 console emulation for fun
- Usefulness? Yes...

https://github.com/NHAS/reverse_ssh

RSSH- FEATURES, FUNCTIONS AND FUN

```
TERMINAL 0 -t 0
nhas@me ➔ sudo docker run -p3232:2222 -e EXTERNAL_ADDRESS=127.0.0.1:3232 -e SEED_AUTHORIZED_KEYS="ssh-ed25519 AAAAC3NzaC1lZD11NTE5AAAAAIikk/jazhGONHIRcJ/WXX+1DcWZYD0s6wLNNArqgruY nhas@me" -v data:/data reversessh/reverse_ssh

nhas@me ~/test

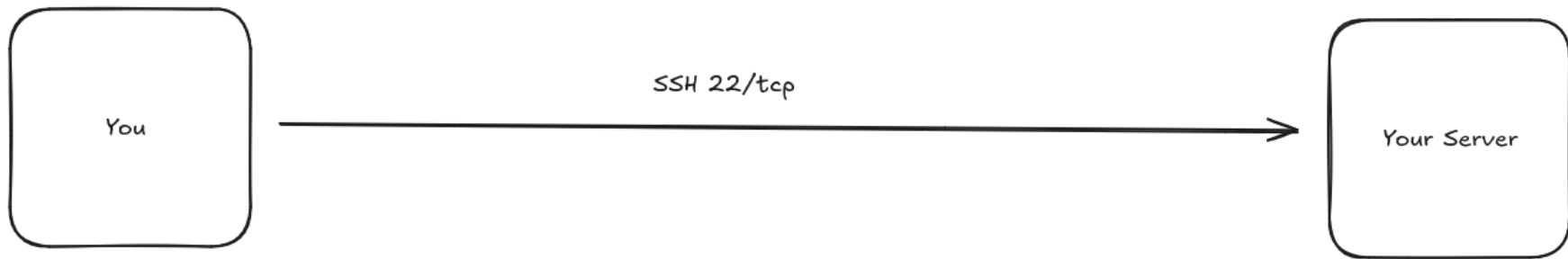
nhas@me ~/test

[0] 0:zsh* "root@me:~" 17:07 25-Sep-24
```

LEARNING TIME!

SSH CRASH COURSE - THE PROTOCOL

- Most people think of SSH just like this:
- Single connection, then magic happens

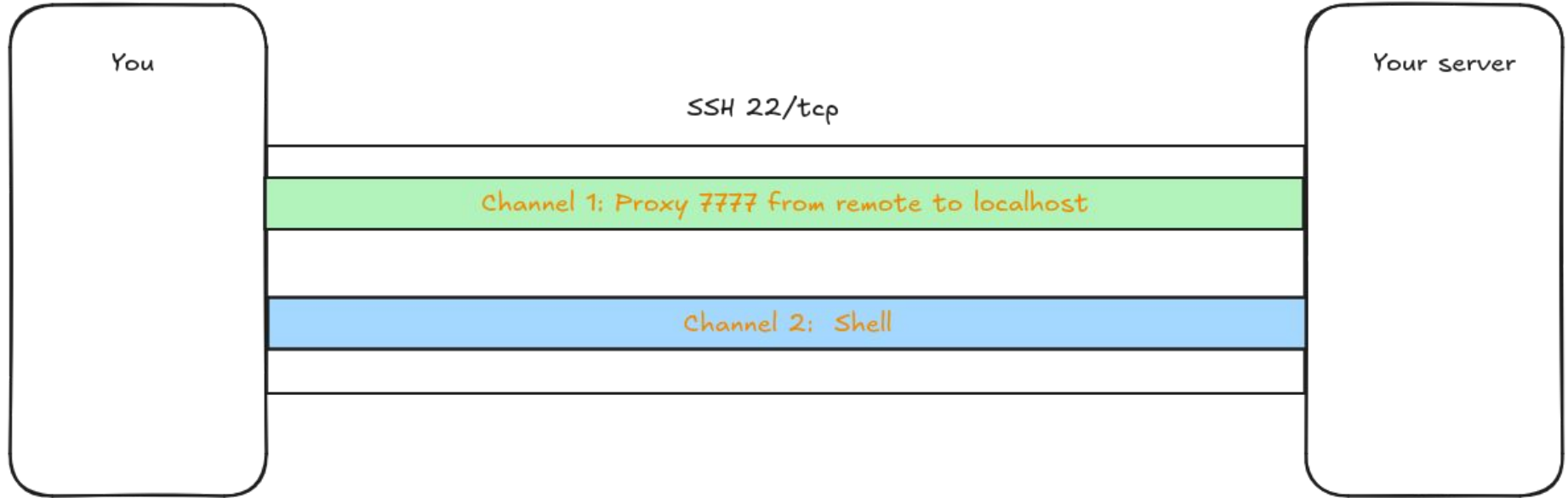


SSH CRASH COURSE - THE PROTOCOL

- But!
- SSH is actually quite interesting, it's multiplexed
- Meaning that while there is **one** connection, over that connection it has multiple logical connections called channels
- So if we do something like and zoom in:

```
ssh -L 7777:127.0.0.1:7777 your.cool.ssh.server
```


SSH CRASH COURSE - THE PROTOCOL



SSH CRASH COURSE - INTEROPERABILITY

- Very simple (as long as doing the crypto and underlying protocol is done for you)
- <https://datatracker.ietf.org/doc/html/rfc4254>
 - Defines the Channels that the SSH tool uses
 - "session"
 - "direct-tcpip"
 - "forwarded-tcpip"
 - "tcpip-forward"

From there it's all implementation details

SSH CRASH COURSE - A BIT ON GOLANG

- SSH library

```
// Before use, a handshake must be performed on the incoming net.Conn.
sshConn, chans, reqs, err := ssh.NewServerConn(tcpConn, sshConfig)
if err != nil {
    log.Printf("failed to handshake (%s)", err)
    continue
}

// Check remote address
log.Printf("new ssh connection from %s (%s)", sshConn.RemoteAddr(), sshConn.ClientVer)

// Print incoming out-of-band Requests
go handleRequests(reqs)
// Accept all channels
go handleChannels(chans)
```

SSH CRASH COURSE - A BIT ON GOLANG

```
func handleChannels(chans <-chan ssh.NewChannel) {  
    // Service the incoming Channel channel.  
    for newChannel := range chans {  
        // Channels have a type, depending on the application level  
        // protocol intended. In the case of a shell, the type is  
        // "session" and ServerShell may be used to present a simple  
        // terminal interface.  
        if t := newChannel.ChannelType(); t != "session" {  
            newChannel.Reject(ssh.UnknownChannelType, fmt.Sprintf("unknown channel type: %s", t))  
            continue  
        }  
    }  
}
```

SSH - TL:DL

- Multiplexed
- Extremely extensible
- Fun to write your own clients
- Secure

WAG

A VPN! KIND OF...

THEFT

- The best way to have fun is to commit crime
- And by crime I mean adding to a well known open source VPN technology and making it better

That's right, it's time to talk about WireGuard™

WIREGUARD - THE INITIAL THEFT

Wireguard:

- An extremely simple VPN
- Uses the Noise Protocol 😊
- Layer 3 only
- UDP only (<https://openvpn.net/faq/what-is-tcp-meltdown/>)
- Hard to setup wrong

WIREGUARD - THE INITIAL THEFT

[Interface]

PrivateKey=+DkFshdf4EnFtMyhx9FUR/n7/iZR1+a788TxIKfa71E=

Address=172.24.25.3

DNS=192.168.234.95

[Peer]

z20-demo

PublicKey=t1GmyuY/IKn7lOILNNraEShEDPkJJ4KC1mVo2cMMdmo=

Endpoint=10.111.56.95:51820

AllowedIPs=172.24.25.1,192.168.234.0/24

WIREGUARD- THE DOWNSIDES

- No MFA :(
- No Enrolment
- No DHCP
- No restrictions per service/port
- Difficult user management
- Why Jason why

WHY JASON WHY

- Jason Donenfeld, creator of Wireguard™
- Tailscale vectorized I/O (2022 -> 2023, 188)
 - Left on read for 2 months
 - 3Gb/s -> 5Gb/s
 - 52 lines of code changed
- Wg-quick, abandoned wg-dynamic
 - “WireGuard currently uses static addresses everywhere. This is because that is mostly a better way to design your network. But in some cases, insane people want dynamic IP addresses or other dynamic configuration.”
- No kernel netlink

Hi Jordan,

> Howdy,
> Sorry if this is terribly rude (Im not entirely sure of the etiquette
> around mailing kernel contributors).
I'm also not entirely sure about the etiquette, but I did not find it
rude at all :-).
>
> I saw your work on adding netfilter notifications on wireguard peer
> changes back in 13 Mar 2021
> (<https://lore.kernel.org/lkml/c97061f5-2d28-0323-c16a-aacacbd734f@lotz.l=i/>)
>
>
> Just wanted to say I absolutely love this idea and your work. Did that
> ever move forward at all? I cant find any other reference to it.
That is nice to hear!

I did update this patch twice:|

v2 <https://www.spinics.net/lists/netdev/msg714366.html> - fixed a
possible uninitialized use.

v3 <https://www.spinics.net/lists/kernel/msg4312979.html> - disables the
notifications per default and allows enabling the monitoring per device
via **netlink**.

Unfortunately I did not get any answer to the last patch and I need to
ask Jason what would be needed to get it added.

> Again, sorry if this is a pain. Just curious (and want it to be added
> to the kernel haha)

No worries :-)

Cheers,

Linus



- MFA 😊
- Enrolment 😊
- Magic eBPF firewall 😊
- User & Device management 😊
- Distributed VPN 😊

A screenshot of the WAG (Wireless Access Gateway) dashboard. The interface is dark-themed with a sidebar on the left containing navigation links: Dashboard, CLUSTER (Events, Members), POLICY (Rules, Groups), MANAGEMENT (Registration Tokens, Users, Devices), Diagnostics, and Settings. The main content area is titled 'Dashboard' and features four summary cards: 'MANAGE MFA' showing 177 users with incomplete MFA registration, 'MANAGE DEVICES' showing 176 devices, 'VIEW ACTIVE SESSIONS' showing 0 sessions, and 'REGISTER DEVICE'. To the right is an 'Instance Details' table. At the bottom is a 'Recent Log Messages' section showing two log entries from 2024/04/28.

Instance Details	
Node ID	d05dec4a11fc7921
Port	53230
Public Key	z2maXX7i3j1B88gjkMT6NBf40iZ7vo6Rfsw5v989zEB=
External Address	192.168.121.61
Subnet	192.168.122.0/24

Recent Log Messages	
2024/04/28 20:23:04	fronkrong 127.0.0.1:40804 admin logged in
2024/04/28 20:18:52	Started Managemnt UI: Listening: 127.0.0.1:4433

WAG - MFA

MFA: It's a bit silly

- Retrieve list of peers
- Store current real ip
- Check if it's changed
- Profit?

WAG - FIREWALL

The screenshot displays the WAG Firewall configuration interface. On the left is a dark sidebar with the WAG logo (a dog) and the text 'WAG'. Below the logo are menu items: Dashboard, POLICY (Rules, Groups), and MANAGEMENT (Registration Tokens, Users, Devices, Diagnostics, Settings). The main area shows a modal dialog titled 'Edit Rule' with a close button (X) in the top right corner. The dialog contains three sections: 'Effects' with a text input field containing 'tester'; 'MFA Routes (New line delimited)' with a text area containing '192.168.3.0/24 80/tcp' and '192.168.4.0/24 22/tcp'; and 'Public Routes (New line delimited)' with a text area containing '1.1.1.1 53/any' and 'google.com 80/tcp 443/any'. At the bottom right of the dialog are 'Cancel' and 'Save' buttons. In the background, a table with columns 'Routes (Number)' and 'Edit' is partially visible, showing two rows with the number '1'.

WAG

Dashboard

POLICY

- Rules
- Groups

MANAGEMENT

- Registration Tokens
- Users
- Devices
- Diagnostics
- Settings

Edit Rule

Effects

tester

MFA Routes (New line delimited)

192.168.3.0/24 80/tcp
192.168.4.0/24 22/tcp

Public Routes (New line delimited)

1.1.1.1 53/any
google.com 80/tcp 443/any

Cancel Save

Routes (Number)	Edit
1	
1	

EBPF - CRASH COURSE

> eBPF is a revolutionary technology with origins in the Linux kernel that can run sandboxed programs in a privileged context such as the operating system kernel. It is used to safely and efficiently extend the capabilities of the kernel without requiring to change kernel source code or load kernel modules.

- <https://ebpf.io/what-is-ebpf/>

EBPF - CRASH COURSE

- XDP
 - Specialised eBPF for fast networking
 - Used everywhere to filter packets

<https://engineering.fb.com/2018/05/22/open-source/open-sourcing-katran-a-scalable-network-load-balancer/>

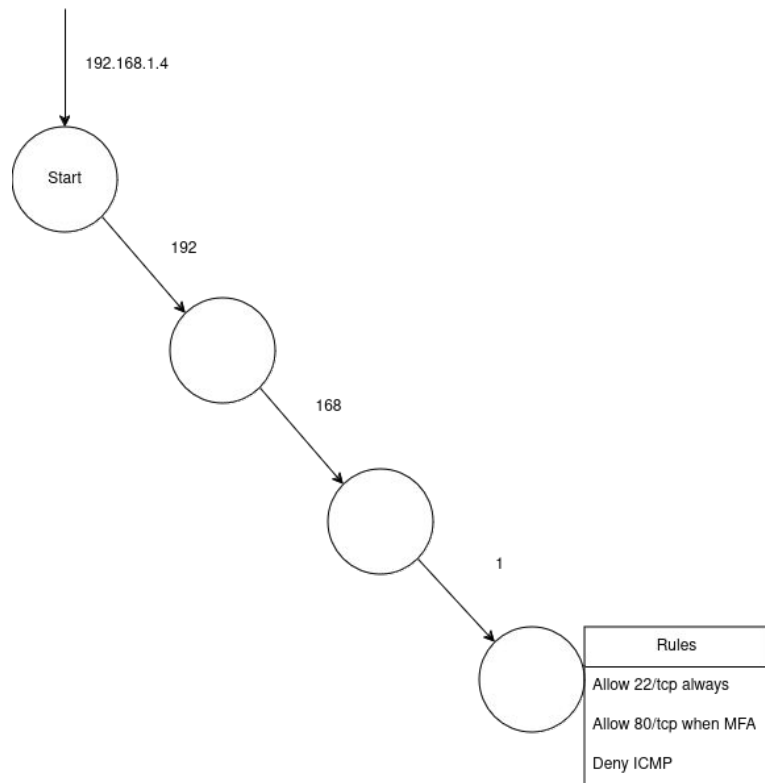
<https://github.com/gamemann/XDP-Firewall>

EBPF - CRASH COURSE

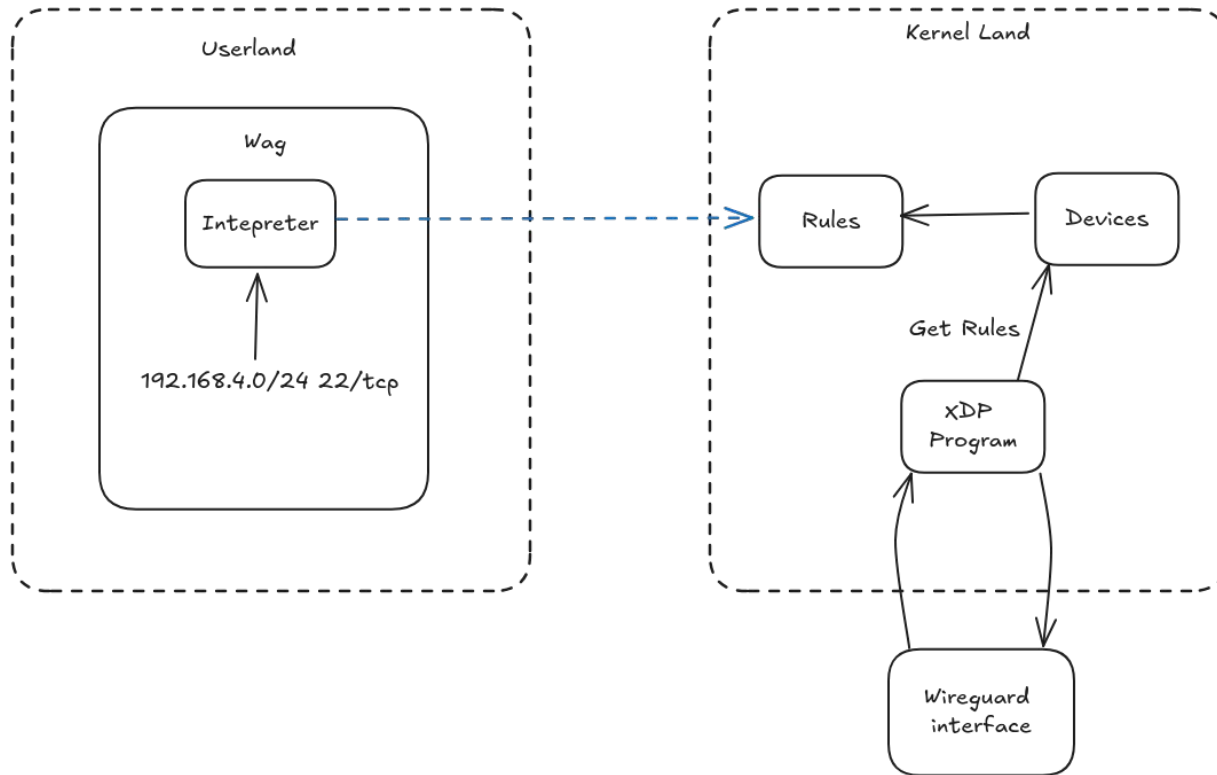
Userland -> Kernel land:

- HashMaps
- Longest prefix matching trie
- And more...

WAG - EBPf PUTTING IT TOGETHER



WAG - EBPf PUTTING IT TOGETHER

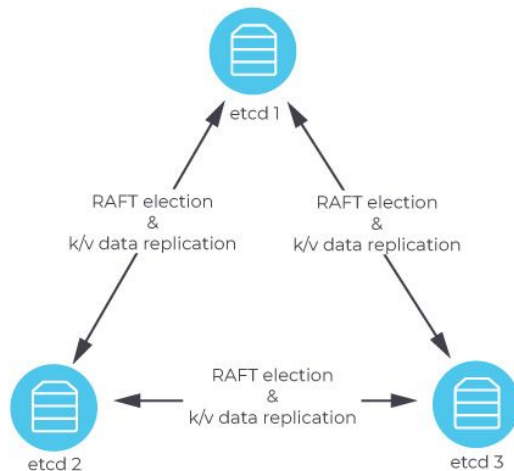


WAG - A BRIEF WORD ON ETCD

- Embeddable in golang
- Distributed Key Value store (Raft)
- Does events (through watching values)

simple etcd cluster:

CYBERTEC
DATA SCIENCE & POSTGRESOL



THE END

38 slides and some change.

Thanks for listening!