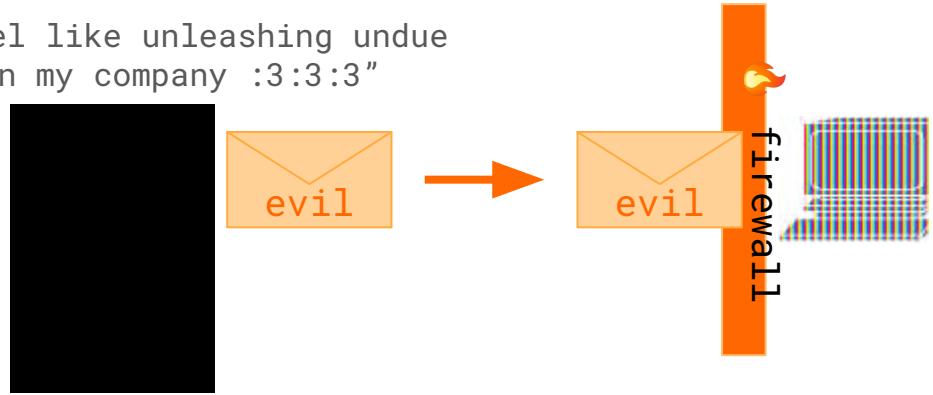
The iptables Firewall

Why Firewalls?

Lock the evil out

“today i feel like unleashing undue harm upon my company :3:3:3”



Why Firewalls (Pt.2)

We don’t want to block everything

- Goal: **drop** evil traffic, **accept** good traffic

hard to do perfectly → cyber attacks still happen

Basics of Networks

- Network: group of devices with *unique* identifiers (IPs)

run ipconfig (Win) or ifconfig (Mac/Linux)

```
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
        inet 192.168.0.239  netmask 255.255.255.0
```

(wifi card)

ip address identifies a device
subnet mask indicates the whole network's size

Subnets! (sub - network)

You have a unique IP address and a common subnet mask

pay attention to the 255's and 0's in subnet mask

ip: 192.168.000.239
mask: 255.255.255.000

note: the mask itself is the **same** for all IPs in a network

this part of IP is **constant** for all IPs in a network

this part of IP is **unique** for all IPs in a network

*this is a bit of an oversimplification, but that won't matter for today

Prefix Notation

255.255.255.0 → /24

255.255.0.0 → /16

255.0.0.0 → /8

255 == 11111111
1 2 3 4 5 6 7 8

The prefix number == the number of 1's in the mask

192.168.0.0/24 == 192.168.0.0 255.255.255.0

Subnet Example

Which IP address(es) are in 10.0.0.0/16 (255.255.0.0)?

- 10.0.0.1 — starts with 10.0 → ✅
- 10.1.7.7 — doesn't start with 10.0 → ❌
- 10.0.30.30 — starts with 10.0 → ✅
- 13.0.0.10 — doesn't start with 10.0 → ❌
- 10.13.40.9 — doesn't start with 10.0 → ❌

Internet Data

Sending data on the Internet is like sending mail

Internet Data	Mail
contains Source IP, Destination IP	sender address, recipient address
protocol → TCP, UDP, etc.	postal service → USPS, FedEx, etc.
source & destination Port Numbers	apartment unit numbers

🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥

actual iptables time

🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥🔥

follow along using the simulator!

algolg.github.io/iptables-sim

Firewall Rules

A rule **matches** specific traffic and specifies whether to **accept** or **drop** it

Random Examples:

- is source ip in 172.0.0.0/24? → DROP
- is the traffic using HTTP (TCP, port 80)? → ACCEPT
- does traffic not meet any of my other rules? → DROP

algolg.github.io/iptables-sim

Basic iptables Rules

iptables uses flags

flag	purpose
-A / --append	Append rule to the end of a Chain (set of rules)
-s / --source	Match data by source IP address
-d / --destination	Match data by destination IP address
-j / --jump	Set the action if data matches (ACCEPT/DROP)

Basic rules pt. 2

default iptables chains:

- INPUT: handles incoming traffic
- OUTPUT: handles outgoing traffic

- there are some others but we won’t be focusing on those today

Basic rules (examples)


```
iptables -A INPUT -s 10.0.0.10 -d 192.168.0.10 -j DROP
```

"if incoming data has a source IP of 10.0.0.10 and a destination IP of 192.168.0.10, then drop it"

```
iptables -A INPUT -s 130.10.10.0/24 -j DROP
```

"if incoming data has a source network of 130.10.10.0 255.255.255.0, then drop it"

flag	purpose
-A / --append	Append rule to the end of a Chain (set of rules)
-s / --source	Match data by source IP address
-d / --destination	Match data by destination IP address
-j / --jump	Set the action if data matches (ACCEPT/DROP)

Try them out!

algolg.github.io/iptables-sim

Practice #1

1. Write a rule to block all traffic destined to the IP 192.168.0.10
2. Write a rule to block traffic from 10.0.0.0 255.255.255.0

flag	purpose
-A / --append	Append rule to the end of a Chain (set of rules)
-s / --source	Match data by source IP address
-d / --destination	Match data by destination IP address
-j / --jump	Set the action if data matches (ACCEPT/DROP)

algolg.github.io/iptables-sim

Practice #1 (Possible Answers)

1. Write a rule to block all traffic destined to the IP 192.168.0.10

   `iptables -A INPUT -d 192.168.0.10 -j DROP`

2. Write a rule to block traffic from 10.0.0.0 255.255.255.0

   `iptables -A INPUT -s 10.0.0.0/24 -j DROP`

flag	purpose
-A / --append	Append rule to the end of a Chain (set of rules)
-s / --source	Match data by source IP address
-d / --destination	Match data by destination IP address
-j / --jump	Set the action if data matches (ACCEPT/DROP)

algolg.github.io/iptables-sim

Multiple rules in a chain

Rules are processed one after another, in the order you entered them -- **order matters**

Ex:

```
iptables -A INPUT -s 45.45.45.45 -j ACCEPT
iptables -A INPUT -j DROP
```

If we flipped these two rules, what would happen?

Data from 45.45.45.45 matches rule #1 → ACCEPT'd

Data from anywhere else matches rule #2 → DROP'd

algolg.github.io/iptables-sim

iptables -S INPUT

Lists all rules from the INPUT chain

(we'll explain iptables -P later)

iptables: Port Filtering

flag	purpose
-p / --protocol	Match traffic from a protocol (tcp,udp,icmp,any)
--sport / --source-port	Match traffic from specific source port(s)
--dport / --destination-port	Match traffic from specific destination port(s)

In order to specify a port, you must first specify the protocol (tcp/udp/any)

Common protocols/ports

protocol	protocol/port
SSH	TCP 22
Telnet	TCP 23
HTTP	TCP 80
HTTPS	TCP 443
DNS	UDP 53

there's an overwhelmingly-long list on wikipedia

Port Filtering Examples


```
iptables -A INPUT -p tcp --sport 23 -j DROP
```

"if incoming TCP data has a source port of 23, then drop it"
(aka "drop incoming Telnet data")

```
iptables -A OUTPUT -p tcp --dport 80,443 -j ACCEPT
```

"if outgoing TCP data has a destination port of 80 or 443, then accept it"
(aka "accept outgoing HTTP and HTTPS data")

flag	purpose
-p / --protocol	Match traffic from a protocol (tcp,udp,icmp,any)
--sport / --source-port	Match traffic from specific source port(s)
--dport / --destination-port	Match traffic from specific destination port(s)

protocol	protocol/port
SSH	TCP 22
Telnet	TCP 23
HTTP	TCP 80
HTTPS	TCP 443
DNS	UDP 53

algolg.github.io/iptables-sim

Practice #2

1. Write a rule to block outgoing DNS requests
    a. test this with curl
    b. **`curl example.com`** should stop working but **`curl 1.1.1.10`** should still work
2. Write a single rule to block incoming remote access connections

flag	purpose
-p / --protocol	Match traffic from a protocol (tcp,udp,icmp,any)
--sport / --source-port	Match traffic from specific source port(s)
--dport / --destination-port	Match traffic from specific destination port(s)

protocol	protocol/port
SSH	TCP 22
Telnet	TCP 23
HTTP	TCP 80
HTTPS	TCP 443
DNS	UDP 53

algolg.github.io/iptables-sim

Practice #2 (Possible Solutions)

1. Write a rule to block outgoing DNS requests

   `iptables -A OUTPUT -p udp --dport 53 -j DROP`

2. Write a single rule to block incoming remote access connections

   `iptables -A INPUT -p tcp --dport 22,23 -j DROP`

   (can also write 22:23 == range from port 22 through 23)

flag	purpose
-p / --protocol	Match traffic from a protocol (tcp,udp,icmp,any)
--sport / --source-port	Match traffic from specific source port(s)
--dport / --destination-port	Match traffic from specific destination port(s)

protocol	protocol/port
SSH	TCP 22
Telnet	TCP 23
HTTP	TCP 80
HTTPS	TCP 443
DNS	UDP 53

algolg.github.io/iptables-sim

Other important flags

iptables -P <Chain> <Default-Policy>

Sets the default policy for a chain

Ex. **iptables -P INPUT DROP**

if incoming data doesn't match any rules in INPUT, then DROP it

Other important flags (Pt. 2)

`iptables -D <Chain> ...`

Deletes a given rule

Ex. **`iptables -D INPUT -d 192.168.0.0/24 -j DROP`**

> deletes a rule that blocked incoming traffic to the 192.168.0.0/24 network

(You can also use a rule number as it appears in `iptables -S <Chain>`)

Other important flags (Pt. 3)

`iptables -F [Chain]`

Flushes a given chain (removes all rules)

Ex. **`iptables -F INPUT`**

removes all rules from INPUT chain

Ex. **`iptables -F`**

removes all rules from all chains

Stateful Connections

Suppose you have the rule: **iptables -A INPUT -j DROP**

(drops all incoming traffic)

If you try to access the PC on any port, what happens?

If you try to `curl` a webpage (example.com), what happens?

Stateful Connections (Pt. 2)

Solution: firewall should track outgoing connections

- if an outgoing connection is expecting a response, then the firewall should dynamically allow a response

State Tables keep track of connections
Once a response is received, the table entry is removed

Implementing in iptables?

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	192.0.2.71	80	Initiated
192.168.1.102	1031	10.12.18.74	80	Established
192.168.1.101	1033	10.66.32.122	25	Established
192.168.1.106	1035	10.231.32.12	79	Established

Stateful Connections (Pt. 3)

iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

- **-m conntrack**: tells iptables that this rule tracks connections
- **--ctstate ESTABLISHED,RELATED**: if the incoming data is from an established connection (or related to an established connection), then match it

Try in combination with **iptables -A INPUT -j DROP** (recall that rule order matters)

Does accessing from the outside host still work?
Does curl example.com still get blocked?

algolg.github.io/iptables-sim

thank you!

-akhil