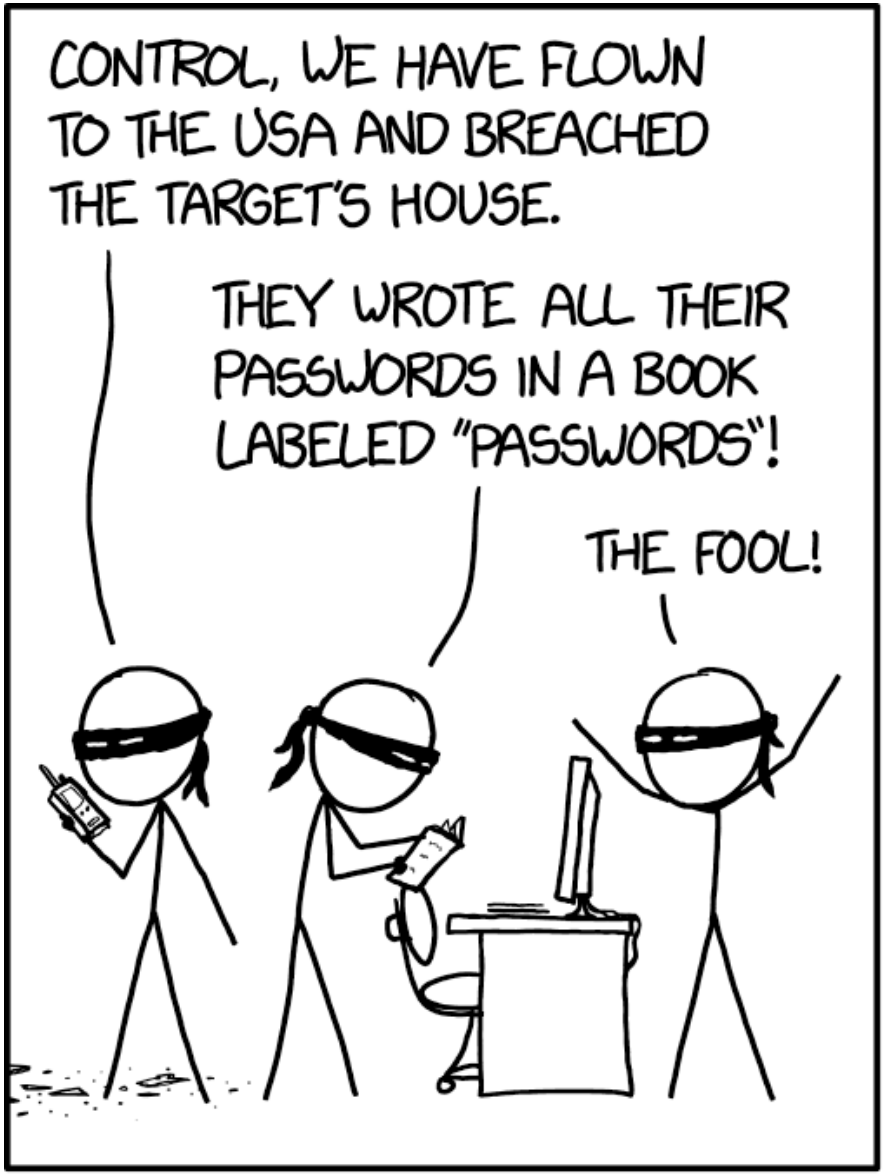
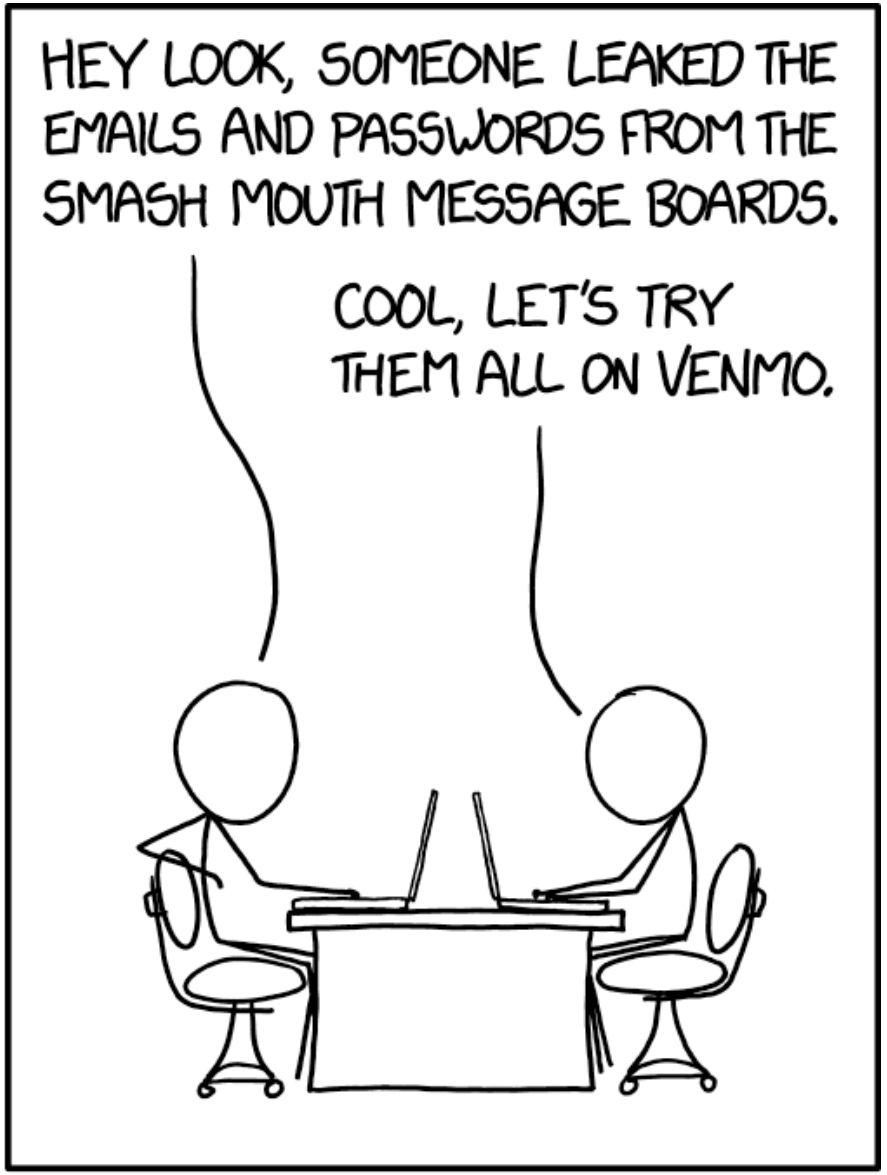


HOW PEOPLE THINK HACKING WORKS



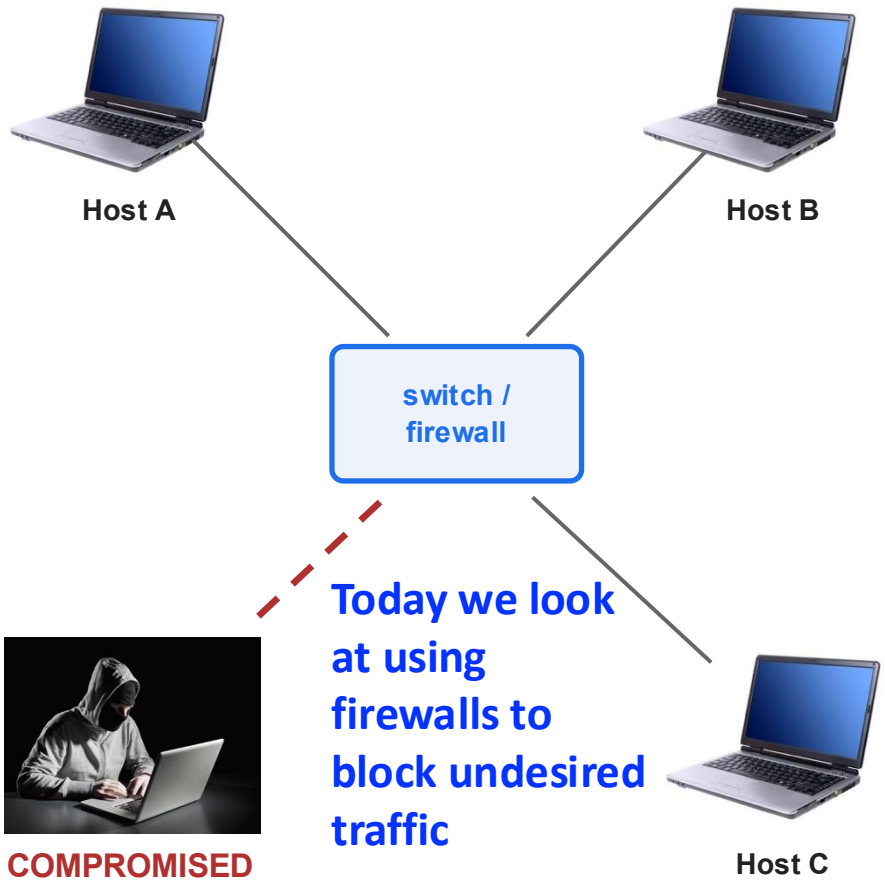
HOW IT ACTUALLY WORKS

Source: https://xkcd.com/2176/

CS 60:
Computer Networks

Threats and firewalls

Discussion: you have a compromised host on your network. Now what?



Things to consider:

- Isolate or watch? Pull it off the network or observe to learn what the attacker is doing?
- Pull the plug to power down? Powering off destroys memory evidence (keys, running malware).
- Preserve evidence — logs, memory image, disk — before you change anything.
- Scope compromise — is it just this host, or has the attacker moved laterally to others?
- Contain spread — block its traffic at the firewall (quarantine rule), rotate credentials.
- Who do you tell? Incident response plan, management, maybe law enforcement / customers.
- Recover — rebuild from known-good, patch the hole that let them in.

Key tension: speed of containment vs. preserving evidence and understanding the full breach

Review from last class

We discussed cryptography

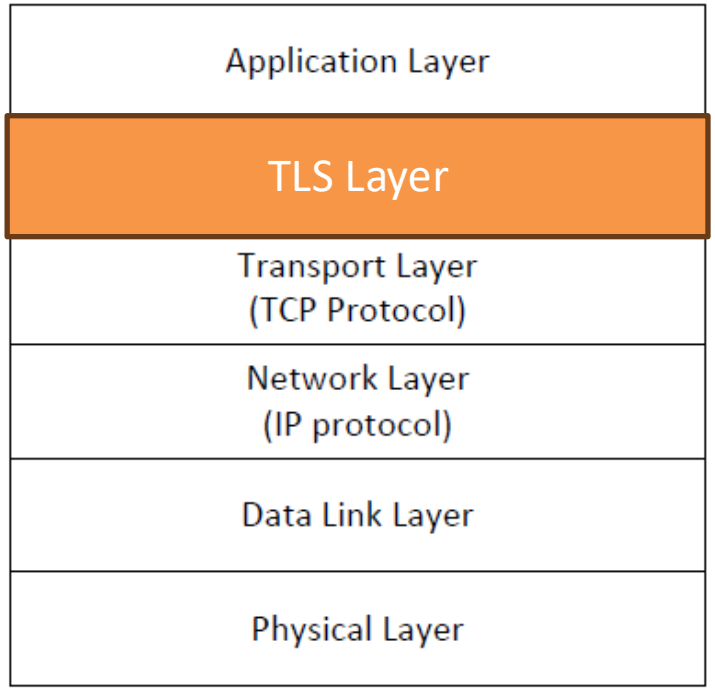
- Symmetric keys (AES) – sender and receiver have the *same* key
- Hashing (SHA-2) – one way function, given input compute digest
- Asymmetric keys (RSA) – public and private keys
  - Slower than symmetric encryption
  - Can solve problem of distributing symmetric keys using a CA

We use cryptography for:

- Transport Layer Security (TLS) – a layer between Layer 7 and Layer 4
  - Authenticates server (sometimes both sides authenticate)
  - Encrypts network traffic (asymmetric to develop common key during handshake, then symmetric for speed)
- Virtual Private Networks (VPN) – creates a secure tunnel between an outside host and a private network like a home or office
- The Onion Router (TOR) – triple encrypt traffic so observers can not see source, destination or message; no node knows all three

TLS sits between the Transport and Application layers to provide encryption

- Unprotected data is given to TLS by Application layer
- TLS handles encryption, decryption and integrity checks
- Client identifies server using CA's signed copy of server's public key during handshake
- TLS negotiates a common key using asymmetric encryption, then uses symmetric encryption for speed
- TLS gives protected data to Transport layer

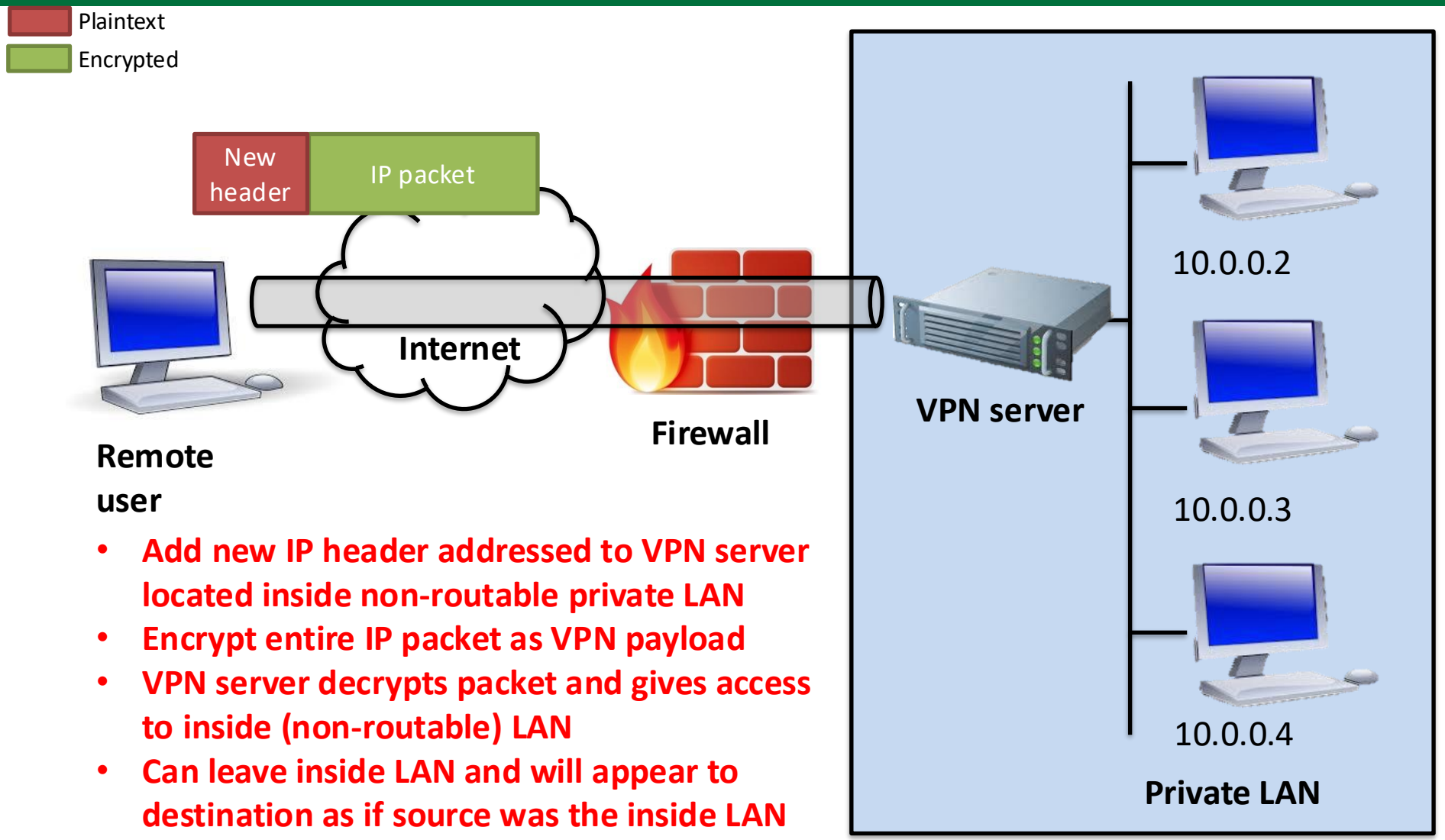


Are IP and MAC addresses hidden?
No!
Why?
Needed for routing and switching

Is Layer 7 message hidden?
Yes!
Can an adversary see which hosts are communicating with each other?
Yes!

Adapted from *Computer & Internet Security* by Du

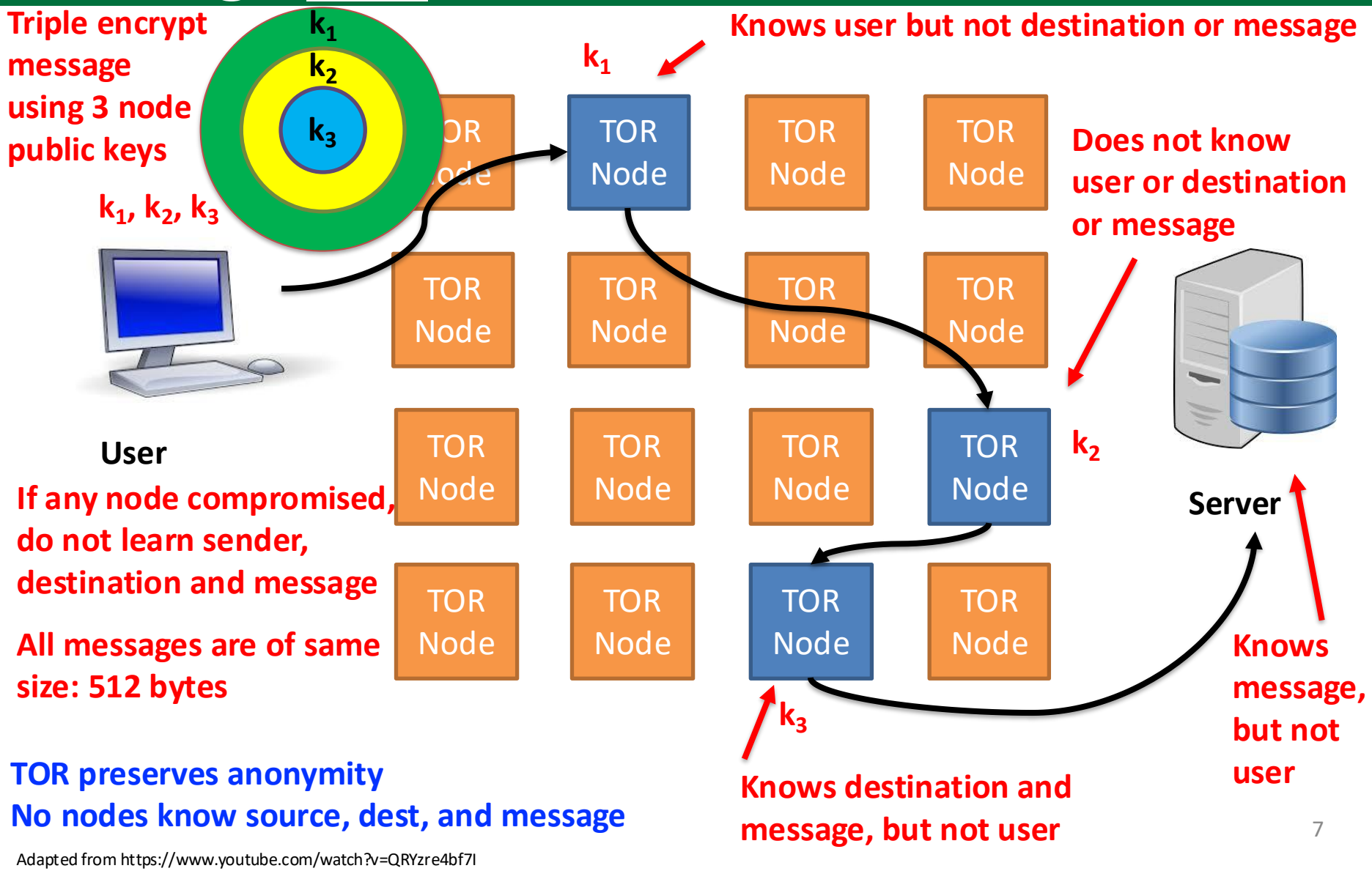
A VPN creates an encrypted tunnel into a network



- Add new IP header addressed to VPN server located inside non-routable private LAN
- Encrypt entire IP packet as VPN payload
- VPN server decrypts packet and gives access to inside (non-routable) LAN
- Can leave inside LAN and will appear to destination as if source was the inside LAN
- Hides destination and message to outsiders

Adapted from *Computer & Internet Security* by Du

The Onion Router (TOR) obscures a message *and* the source and destination



Adapted from https://www.youtube.com/watch?v=QRYzre4bf7I

Agenda



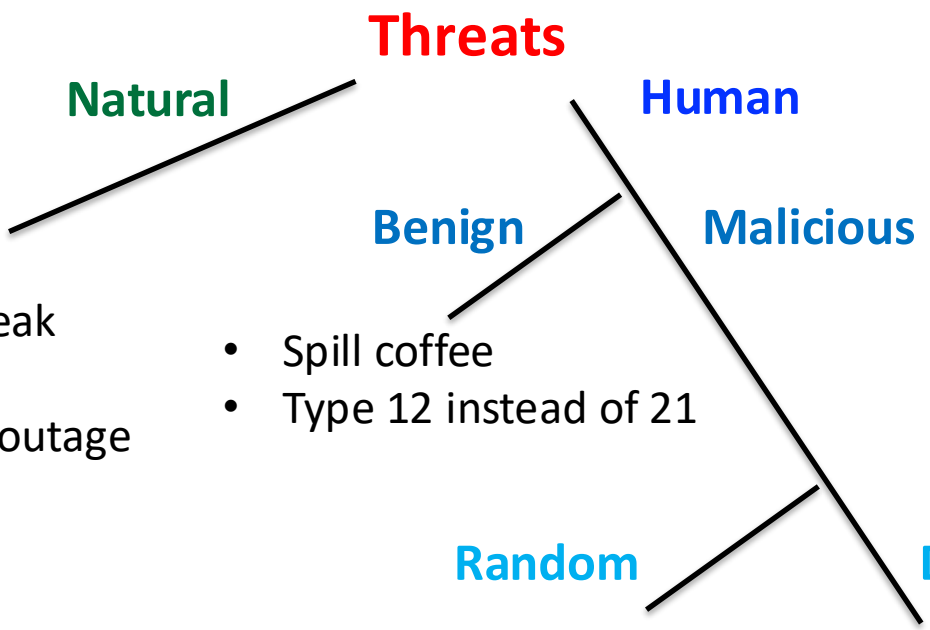
1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises

Information is one of the few things that can be in multiple places at the same time

-- Pierson?

Threats can be natural or human, benign or malicious, random or directed

Threat taxonomy



Just you
- Component failure
- Fire/water pipe break

Localized
- Block-wide power outage

Regional
- Hurricane/flood

- Spill coffee
- Type 12 instead of 21

- Malicious code on a general web site
- Scam phone calls to all numbers
- Phishing

- Social engineering
- Spear phishing
- APTs

Not always clear when attack is underway (e.g., system failure could be natural or directed attack)

NYC power outage

Disaster Recovery (DR)
- **Organization's plan to recovery different types of threats**
- **My view: large tomes are not useful!**

Adapted from *Security in Computing* by Pfleeger, Pfleeger, and Margulies

Discussion

How do you know if something is an attack or an ordinary human or technological breakdown?

Have you ever experienced harm as a result of a failure of computer security? Was it malicious or not? Did the attack target you specifically or were you unlucky?

There are many different types of threat actors

Script kiddie

- Runs pre-made scripts
- May not know what's really happening
- Not necessarily young
- No funding
- Often motivated by the hunt



Hacktivist

- Hacker and activist
- Have an agenda
- Can be sophisticated
- Historically funding was limited, but that is changing



Adapted from https://www.professormesser.com/security-plus/sy0-501/threat-actors/

There are many different types of threat actors



Organized crime

- Motivated by money
- Sophisticated
- Well funded
- May sell data hacked



Governments

- Experts working for government agency
- Can be extremely sophisticated
- Advanced Persistent Threat (APT)

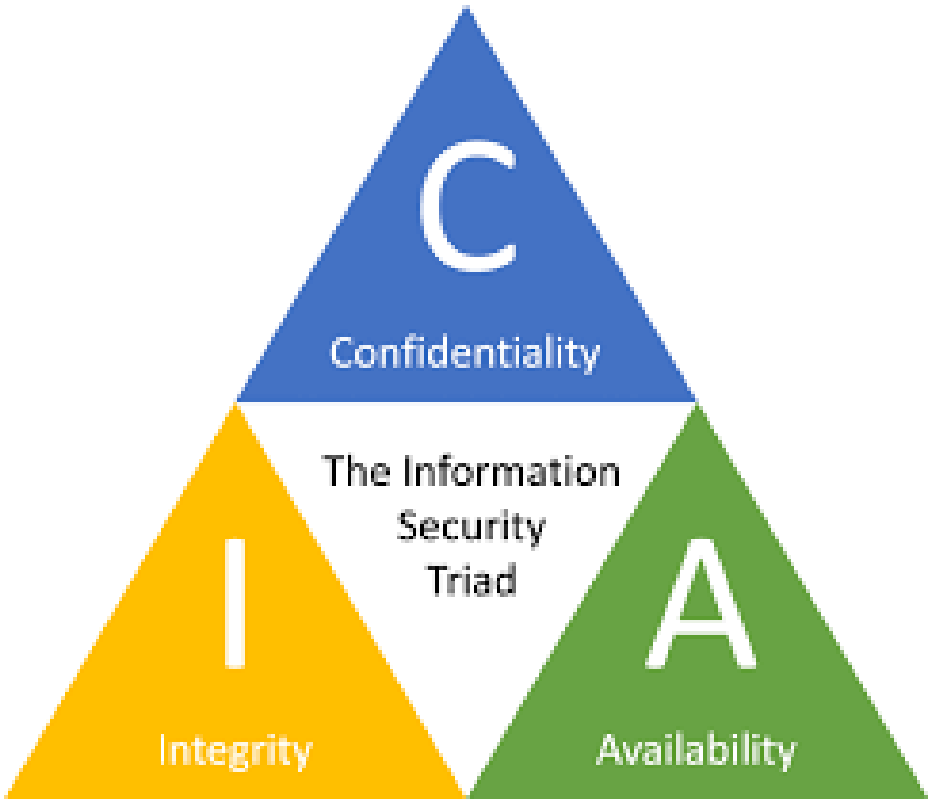


Insiders

- Understand organization
- Have a grudge
- Know/have access to the sensitive data

Adapted from https://www.professormesser.com/security-plus/sy0-501/threat-actors/

What are the adversary's goals?



Goals: compromise CIA triad

- Financial gain
- Espionage
  - Government
  - Corporate
- Sabotage
- Political or ideological motives
- Reputational damage
- Loss of business (DoS)

https://www.kelacyber.com/academy/cti/understanding-the-different-types-of-cyber-adversaries/

Cyber kill chain and MITRE ATT&CK framework describe what adversaries do

Goal: understand attack process and mitigate it



PRE-ATT&CK

- Priority definition (planning, direction)
- Target selection
- Information gathering (technical, people, organizational)
- Weakness identification (technical, people, organizational)
- Adversary OpSec
- Establish & maintain infrastructure
- Persona development
- Build capabilities
- Test capabilities
- Stage capabilities

ATT&CK FOR ENTERPRISE

- Initial access
- Execution
- Persistence
- Privilege escalation
- Defense evasion
- Credential access
- Discovery
- Lateral movement
- Collection
- Exfiltration
- Command and control
- Impact

OpSec: Operational Security
Pizza orders example

https://www.cyberbit.com/cybersecurity-training/introducing-the-mitre-attck-enterprise-framework-collection-2/

In the Recon phase, adversaries try to learn as much as possible about the network

Network scanning methodology

1. **Check for live systems**
   - Gives a list of computers are on the network
   - Ping can do this
2. **Check for open ports**
   - Different machines will run different services
   - Services typically run using standard ports (e.g., ssh on port 22)
   - See ports on which computers are listening
3. **Scan around the IDS**
   - Don't be too noisy, avoid IDS detection (do not anger the IDS!)
4. **Perform banner grabbing**
   - Finds OS and versions services are running
5. **Scan for vulnerabilities**
   - Given OS and service versions, check for CVE's (Common Vulnerabilities and Exposures)
6. **Draw network diagrams**

Nmap is an amazingly powerful tool that can help with these steps

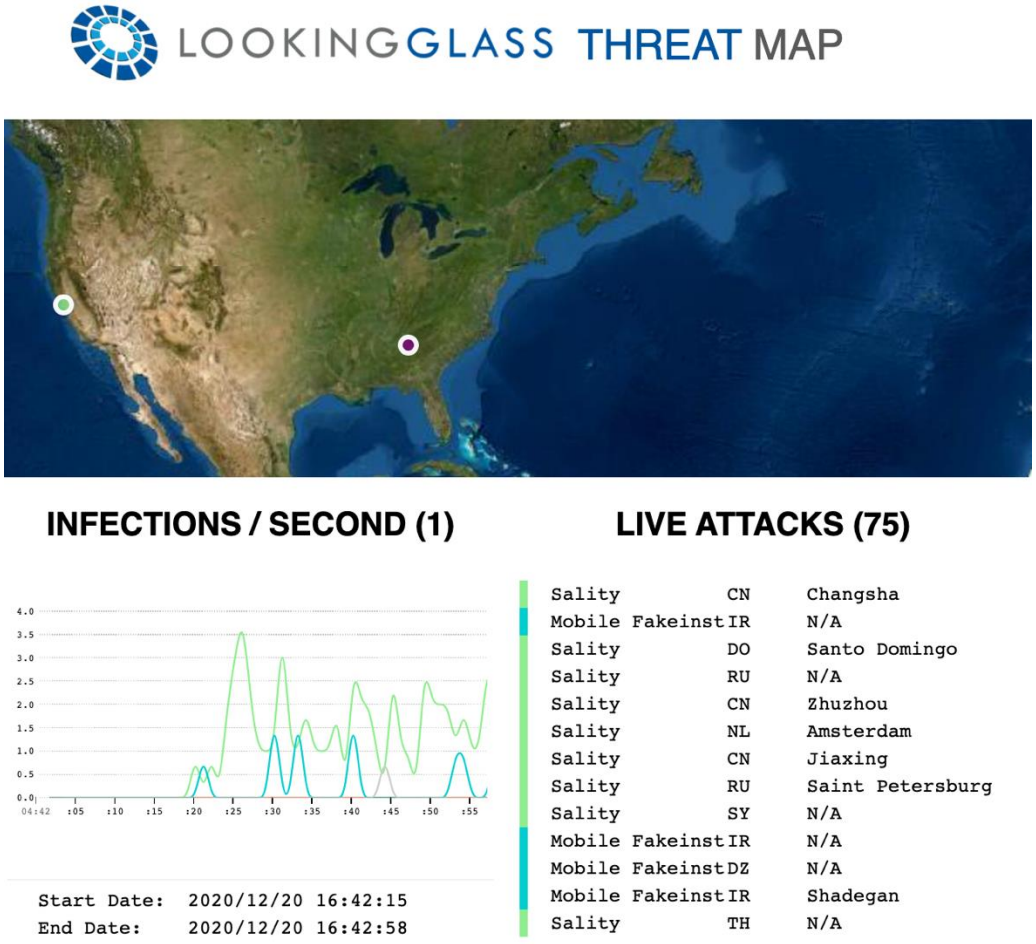
You will write a similar tool in Lab 5

Walker, Matt. *CEH Certified Ethical Hacker All-In-One Study Guide, Fourth Edition*. McGraw-Hill, 2019.

Botnets command large numbers of devices via Command-and-Control server

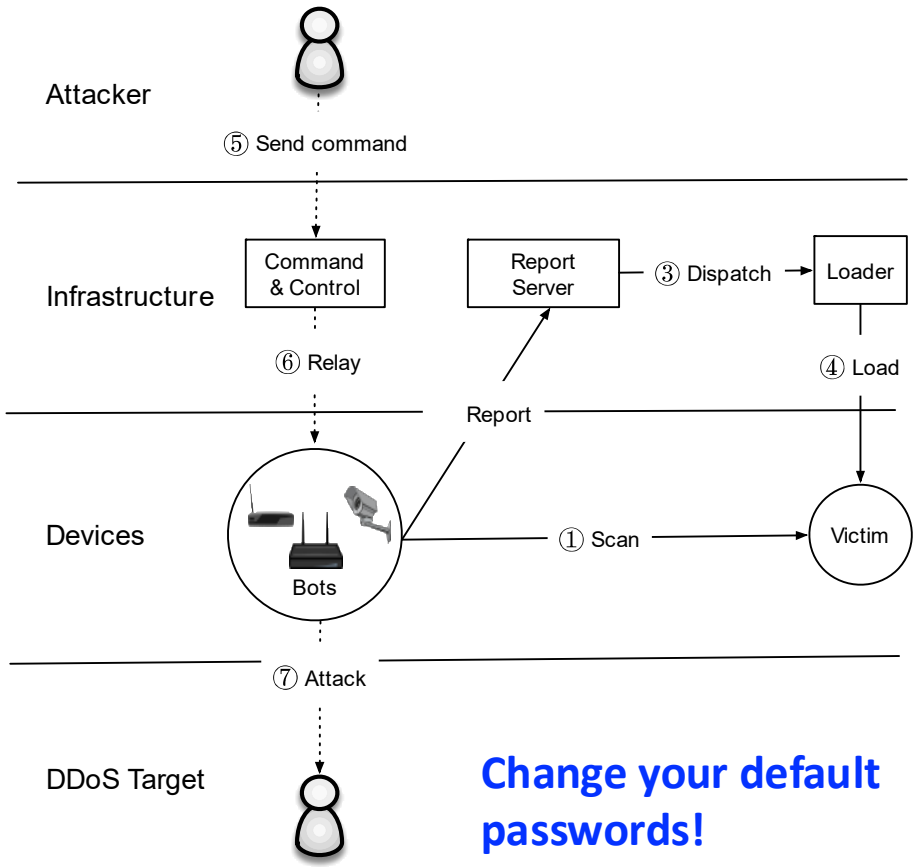
Botnet

- "Robot network"
- Sometimes installed via trojan or virus
- Sometimes install with default passwords
- Wait for command from Command-and-Control server
- Can rent time on botnet for DOS attack
- Example: Mirai botnet
- Real time view: https://threatmap.checkpoint.com/



Devices in a botnet can act together to cause major disruption

Mirai botnet



Change your default passwords!

Many devices on the Internet used default credentials

Password	Device Type
123456	ACTi IP Camera
anko	ANKO Products DVR
pass	Axis IP Camera
888888	Dahua DVR
666666	Dahua DVR
vizxv	Dahua IP Camera
7ujMko0vizxv	Dahua IP Camera
7ujMko0admin	Dahua IP Camera
666666	Dahua IP Camera
dreambox	Dreambox TV Receiver
juantech	Guangzhou Juan Optical
xc3511	H.264 Chinese DVR
OxhlwSG8	HiSilicon IP Camera
cat1029	HiSilicon IP Camera
hi3518	HiSilicon IP Camera
klv123	HiSilicon IP Camera

Mirai quickly compromised hundreds of thousands of devices

Antonakakis, Manos, et al. "Understanding the Mirai botnet." *USENIX Security Symposium*. 2017.

Devices in a botnet can act together to cause major disruption

Mirai botnet



Distributed Denial of Service (DDOS) attack

- DNS provider Dyn was flooded with traffic in 2016
- Dyn servers overwhelmed
- Users unable to access web sites

Mirai and variants are still alive today

- Researchers discovered Mirai with updated functionality
- Malware authors used TOR Network for anonymity

https://gbhackers.com/most-dangerous-iot-malware-mirai-now-using-cc-server-in-the-tor-network-for-anonymity/

The fortress mentality doesn't work anymore; need P > D + R

Protection > Detection + Reaction



>



+



Protection

- Secure systems development
- Cryptography
- Secure comms
- Firewalls/VPNs
- Physical security

Detection

- IDS/IPS
- Pen testing/ forensics

Lots of focus on protection, don't forget detection and reaction!

Note: this framework is not particularly popular, but I like it!

Reaction

- Automatic
- Manual
- Incident management
- Team leadership
- Disaster recovery

Schwartau, Winn. "Time-based security explained: Provable security models and formulas for the practitioner and vendor." *Computers & Security* 17.8 (1998): 693-714.

Pierson's imperfect method of patching an organization

Modified "patch and pray"

- New patches come out frequently
- Patch Tuesday (second Tues in month) -> exploit Wednesday
- Could apply them across your organization and hope updates don't break anything (called "patch and pray")

Instead roll out over a couple of days

- Choose the least time-dependent part of the organization
- Roll out patch to half of that group on day 1
- See if anything breaks
- Next day
  - Complete patch of first part of organization
  - Roll out patches to half of another of another group in organization
- Continue until whole organization covered
- This approach is imperfect to be sure!

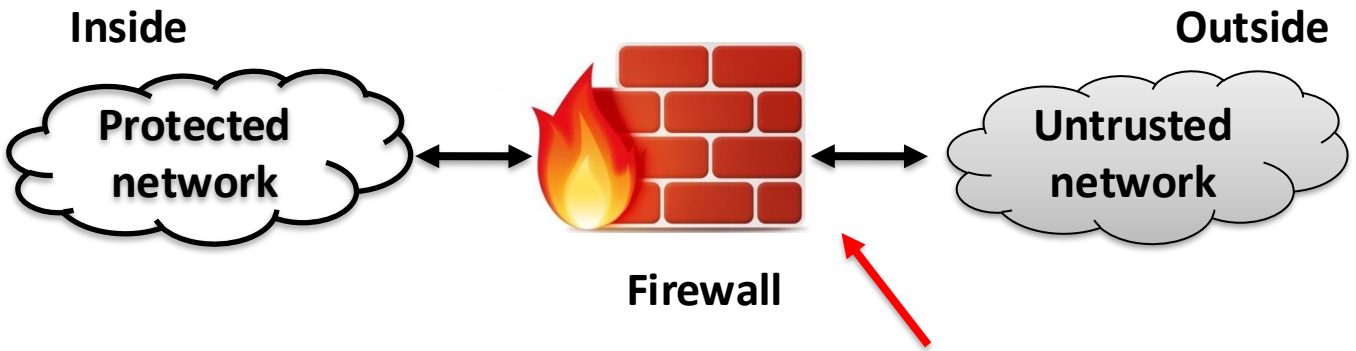
Significant problems:

- **Parts of organization exposed for a few days**
- **Problems for one group may not affect another**
- **If patch causes problem, taking down half a group may still be a big issue!**

Agenda

1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises

Firewalls are designed to stop unauthorized network traffic



Firewall separates trusted and untrusted network components

- Often sold as a hardware appliance, but the appliance is just a computer (usually runs Linux)
- Often part of routers in homes

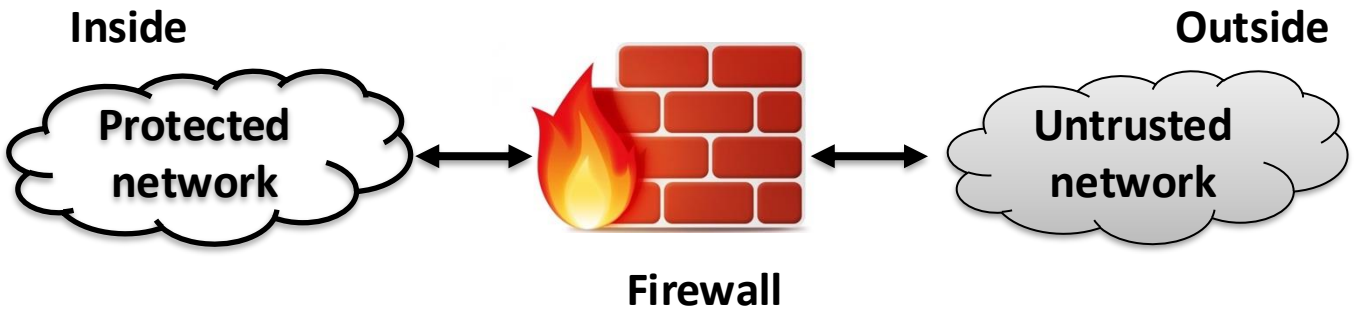
Main functions:

- Filtering packets
- Redirecting traffic
- Protecting against network attacks

Requirements

- Traffic between trust zones should flow through firewall
- Only authorized traffic as defined by policy passes
- Firewall itself must be hardened against attack

Firewalls are designed to stop unauthorized network traffic

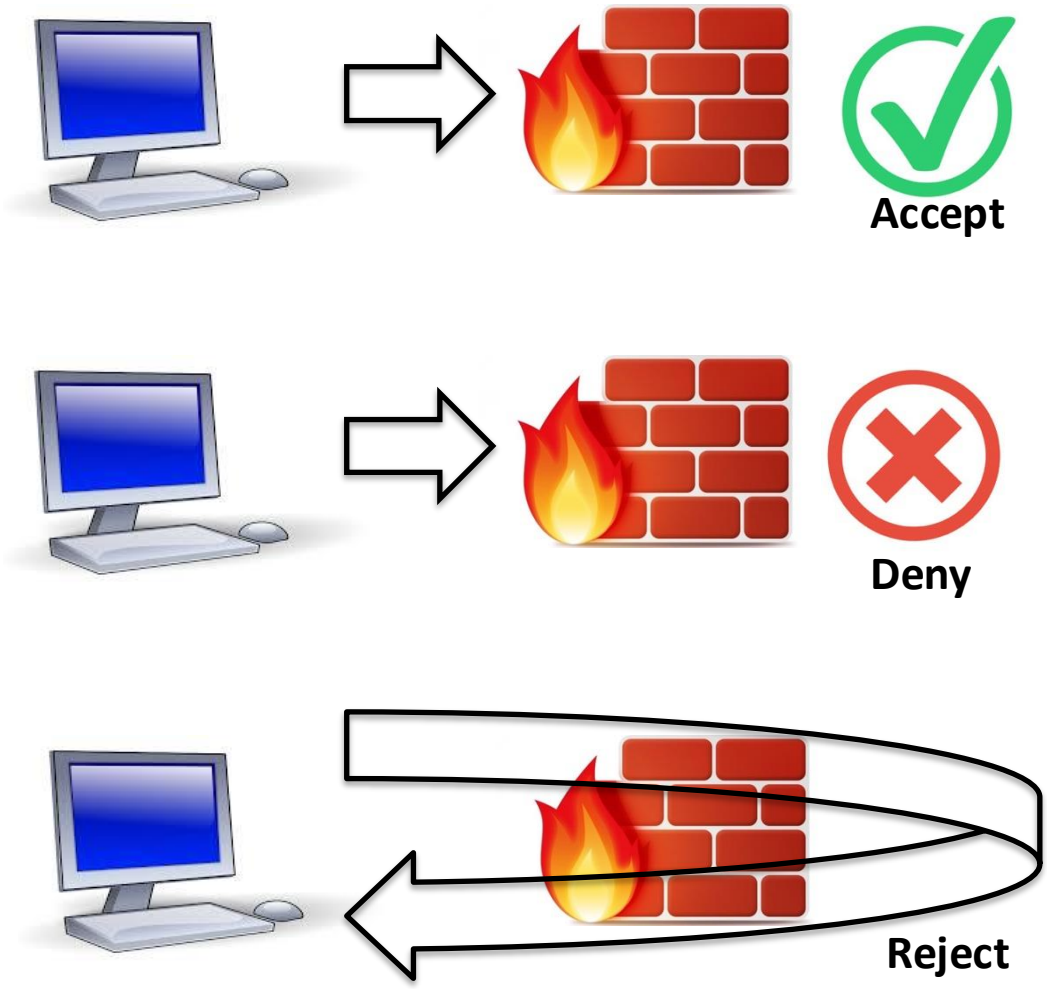


Firewall separates trusted and untrusted network components

Firewall policy

- User control: Controls access based on the role of the user. Applied to users inside the firewall perimeter
- Service control: Controls access by the type of service offered by the host. Applied on the basis of network address, protocol of connection and port numbers
- Direction control: Determines the direction in which requests may be initiated and are allowed to flow through the firewall. It tells whether the traffic is "inbound" (from the outside to firewall) or "outbound" (from the inside to the firewall)

Firewalls normally take one of three actions on a packet: accept, deny, or reject



Accept
Traffic allowed through firewall

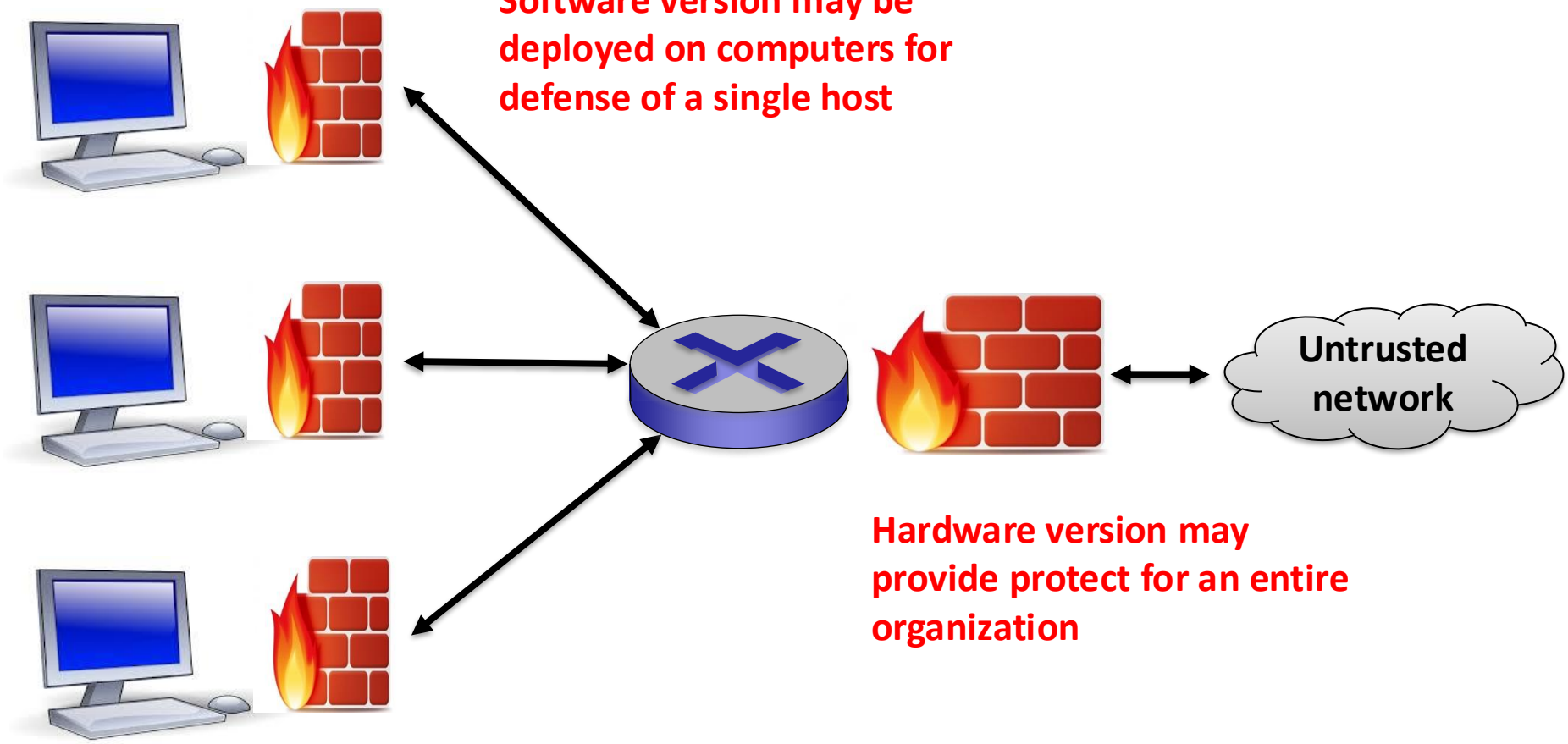
Deny
Traffic stopped at firewall

Reject
Traffic stopped at firewall and sender notified

Filtering can be done on ingress or egress

Why filter on egress?
Prevent access to outside (porn, gambling, …)
Corporate secrets exfil

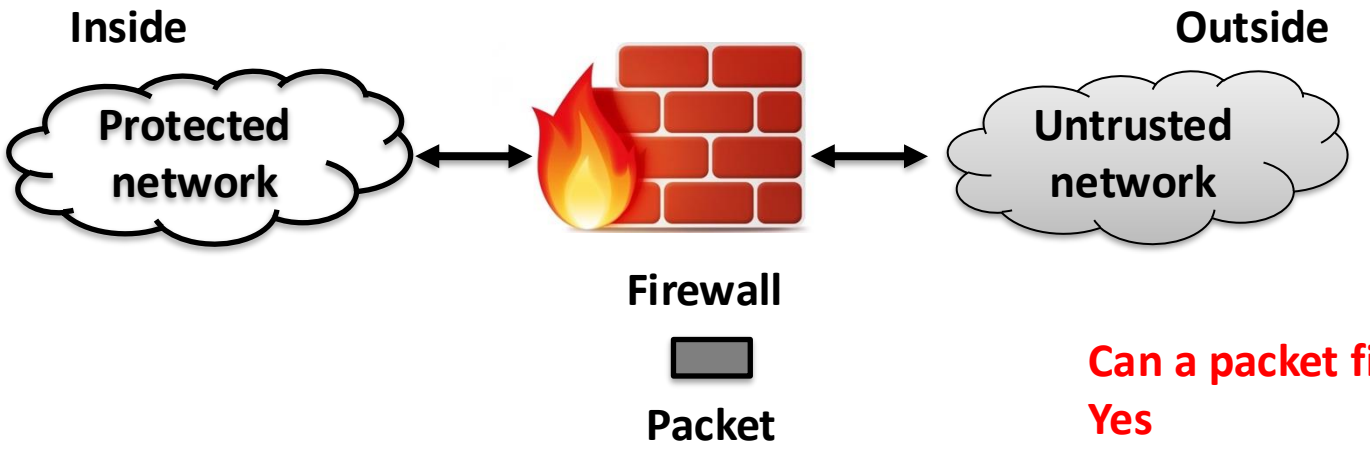
Firewalls come in software and hardware versions to protects hosts or organizations



There are four main types of firewalls

1. Packet filters
2. Stateful inspection
3. Application proxy
4. Next Generation FireWalls (NGFW)

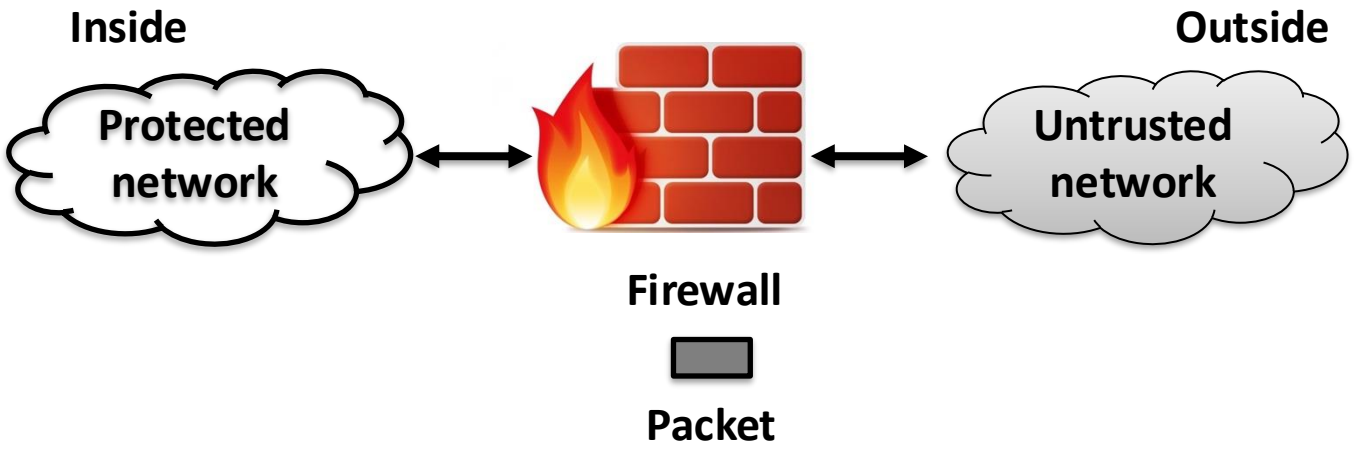
1) Packet filter firewalls make decisions based on a packet-by-packet basis



Can a packet filter work with TLS?
Yes
Only Layer 7 encrypted

- Decisions made based on a *single* packet
- Controls traffic based on the information in packet headers up to Layer 4, without looking into the payload that contains application data
- Does not consider if the packet is a part of existing stream of traffic
- Does not maintain info on connection state

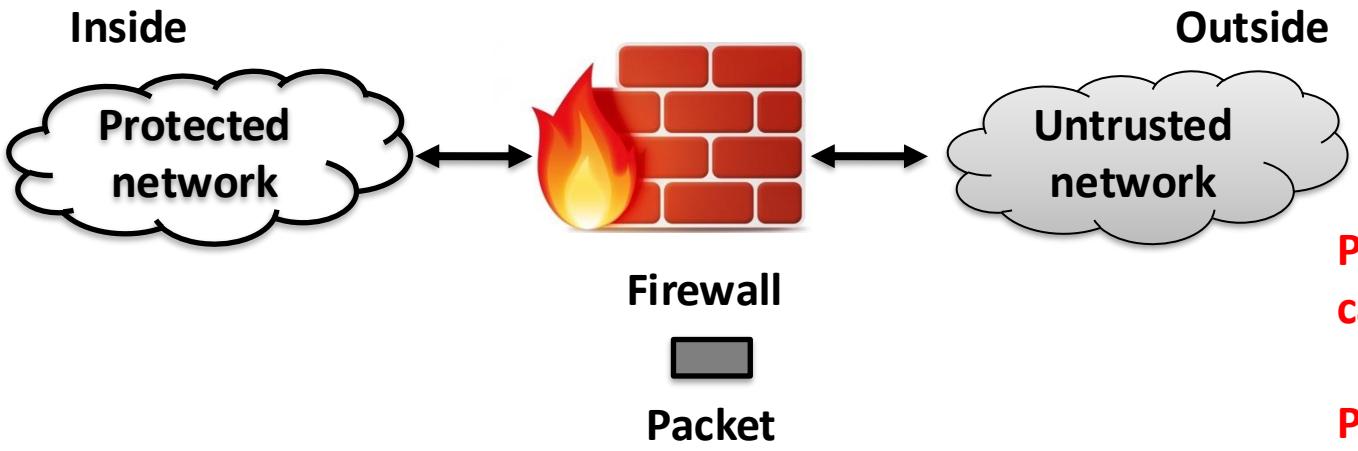
1) Packet filter firewalls make decisions based on a packet-by-packet basis



Decision to forward/reject/drop packet based on Layer 4 and lower:

- Source destination IP address
- Source, destination port numbers
- ICMP message type
- TCP SYN, ACK bits

1) Packet filter firewalls make decisions based on a packet-by-packet basis



Packet filter firewalls also called Stateless Firewalls

Primary advantages: speed and cost

Think of a packet filter as generalized forwarding with a smaller action set

Same idea as the OpenFlow flow tables from the data-plane lecture: match on header fields (IP, port, protocol, flags), apply an action, first match wins

- Forwarding action: *send out port X*
- Filtering action: *accept / drop / reject*
  Both are stateless, both run in TCAM at line speed — which is exactly why packet filtering survives in routers, switches, and cloud ACLs

Examples

- Block UDP and telnet: block incoming and outgoing datagrams with IP protocol field = 17 and with either source or dest port = 23
  - Result: all incoming and outgoing telnet connections are blocked

- Block inbound TCP:
  - Result: prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside

Common on home routers with built in firewall
Must allow connection if you want to play games with your online friend
- **Often use port forwarding to allow one port through the firewall**

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

More examples

Policy	Firewall Setting
No outside Web access	drop all outgoing packets to any IP address, port 80

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

More examples

Policy	Firewall Setting
No outside Web access	drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only	drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

More examples

Policy	Firewall Setting
No outside Web access	drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only	drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth	drop all incoming UDP packets - except DNS and router broadcasts

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

More examples

Policy	Firewall Setting
No outside Web access	drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only	drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth	drop all incoming UDP packets - except DNS and router broadcasts
Prevent your network from being used for a DoS attack	drop all ICMP packets going to a broadcast address (stops Smurf attack)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

More examples

Policy	Firewall Setting
No outside Web access	drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only	drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth	drop all incoming UDP packets - except DNS and router broadcasts
Prevent your network from being used for a DoS attack	drop all ICMP packets going to a broadcast address (stops Smurf attack)
Prevent your network from being tracerouted	drop all outgoing ICMP TTL expired traffic

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Access Control Lists are used to apply rules

ACL: Rules applied top to bottom to incoming packets: (action, condition) pairs: looks like OpenFlow forwarding

Allow Ack'd inbound web connections

Users can surf the web

Allow outbound connections to web

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Access Control Lists are used to apply rules

ACL: Rules applied top to bottom to incoming packets: (action, condition) pairs: looks like OpenFlow forwarding

Allow DNS response in

Users can resolve domain names

Allow DNS requests out

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Access Control Lists are used to apply rules

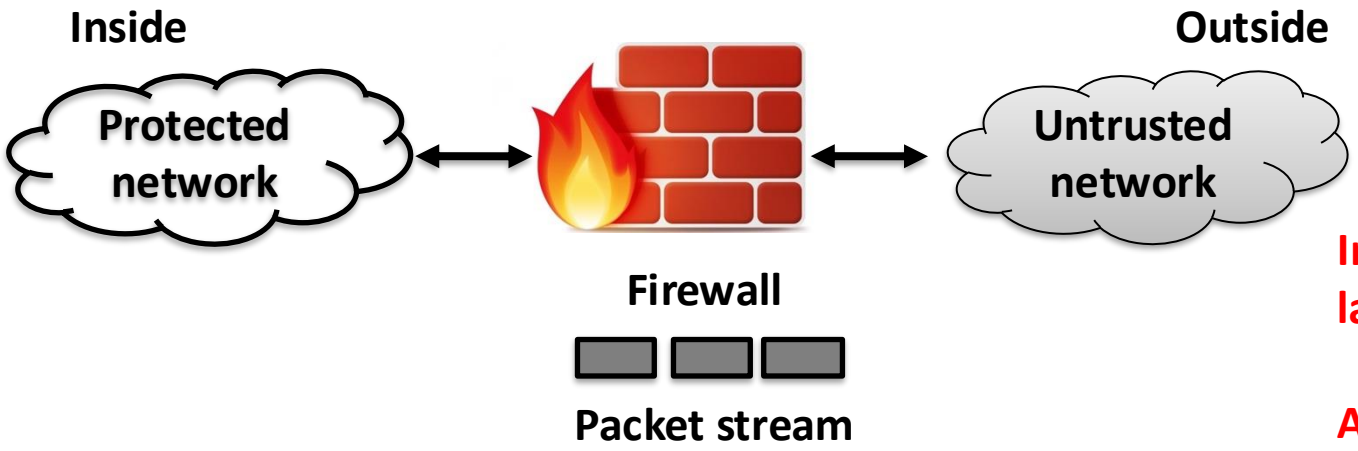
ACL: Rules applied top to bottom to incoming packets: (action, condition) pairs: looks like OpenFlow forwarding

Everything else is blocked!

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

2) Stateful firewalls make decisions based on a stream of packets



Inspects up to transport layer (layer 4)

Able to prevent many kinds of spoofing attacks

- Decisions based on a *stream* of packets
- Tracks the state of traffic by monitoring all the connection interactions until is closed
- Connection state table is maintained to understand the context of packets
- Example: connection are only allowed through the ports that hold open connections

Usually more expensive than packet filer firewalls

2) Stateful firewalls make decisions based on a stream of packets

- *Stateless packet filter:* heavy handed tool
  - Allows packets that "make no sense," e.g., dest port = 80, ACK bit set, even if no TCP connection established from a host on the inside

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

2) Stateful firewalls make decisions based on a stream of packets

- *Stateful packet filter:* track status of every TCP connection
  - Track connection setup (SYN), teardown (FIN): determine if incoming, outgoing packets "makes sense"
  - Timeout inactive connections at firewall

Non-requested packets dropped

action	source address	dest address	proto	source port	dest port	flag bit	check connection
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X

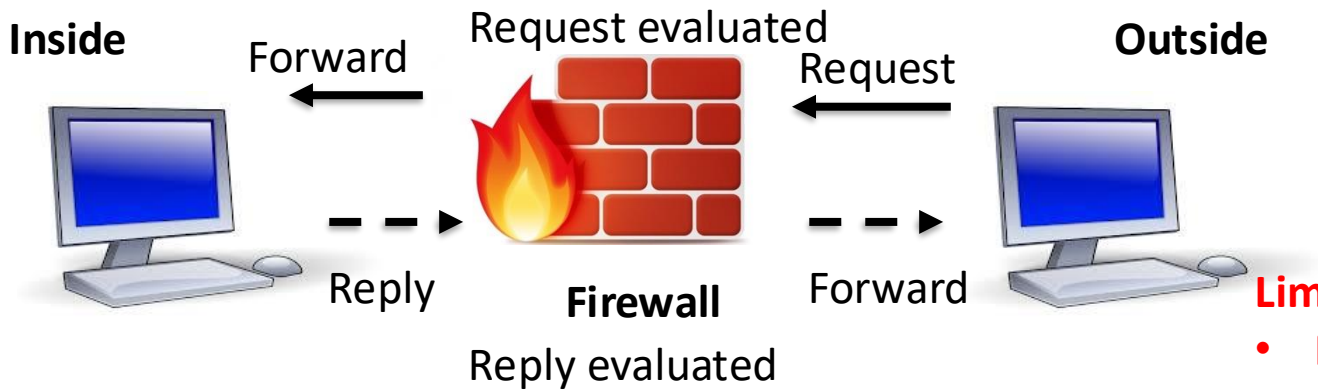
Check inbound is reply to inside

Source	Destination	sport	dport
222.22.1.7	37.96.87.123	12699	80
22.22.93.2	199.1.205.23	37654	80
222.22.65.143	203.77.240.43	48712	80

Open connections Each initiated from the inside

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

3) Application proxy firewalls control access to/from a service



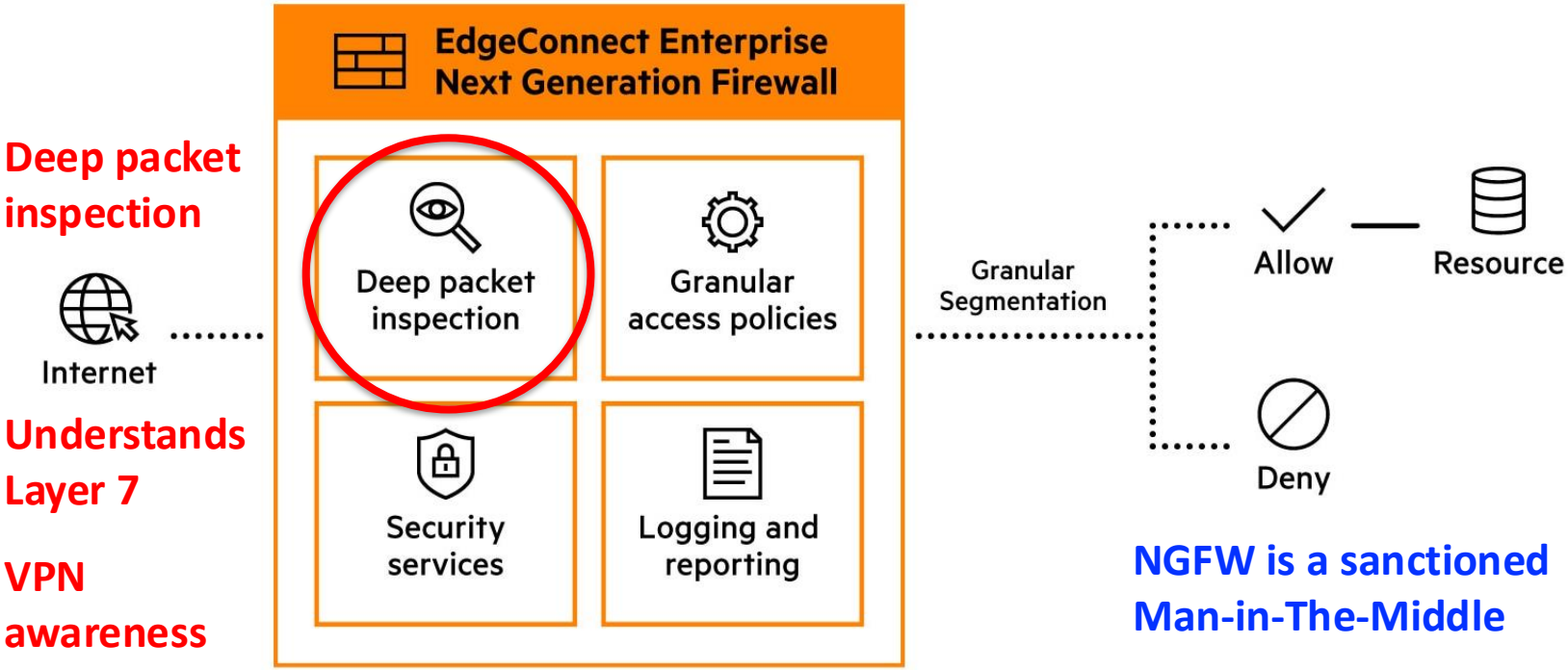
Proxy acts like a Man-in-the-Middle
SQUID was a popular implementation

Limitations:
- **Must implement proxy rules for each app**
- **Slower than other firewalls**

Proxy firewalls mostly replaced by Next Generation Firewalls

- Firewall controls I/O to/from application or service
- Acts as intermediary (no direct contact to app/service)
- Client connection terminates at firewall, separate connection initiated to application/service
- Data on connection analyzed up to application layer to determine if packet allow or denied/rejected
- Can prevent sensitive information leaks

4) Next Generation Firewalls (NGFW) add capabilities up to Layer 7

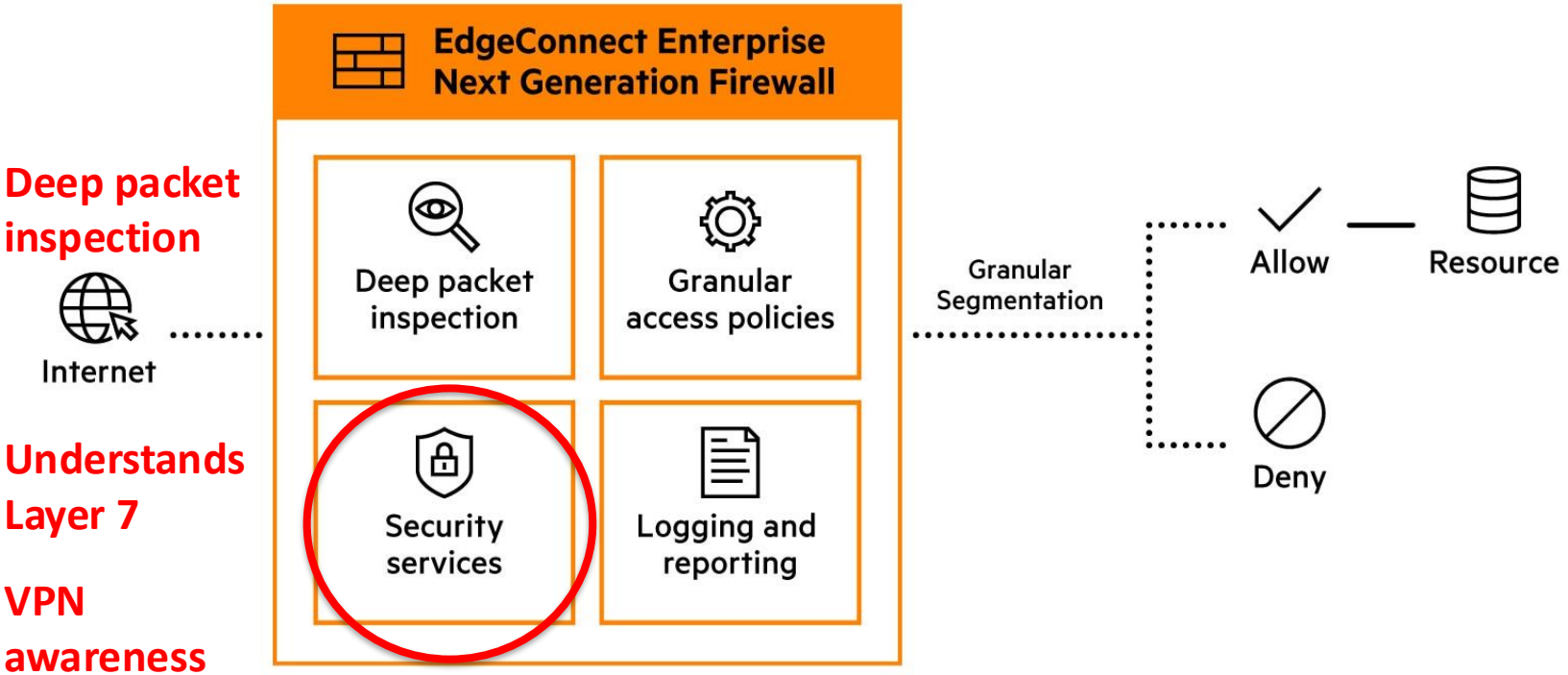


How can the NGFW inspect Layer 7 if traffic is TLS-encrypted?

- NGFW acts as a sanctioned man-in-the-middle, running two TLS connections
- Intercepts the internal host's **ClientHello**, opens its own TLS session to the outside server
- Validates the outside server's certificate normally (against public CAs)
- Creates a certificate for that site on the fly, signed by the **organization's internal CA**
- Internal host accepts it — IT pre-installed that CA in every device's trust store
- NGFW now decrypts, inspects Layer 7, and re-encrypts to each side

https://www.hpe.com/us/en/what-is/next-gen-firewall.html

4) Next Generation Firewalls (NGFW) add capabilities up to Layer 7



Deep packet inspection

Understands Layer 7

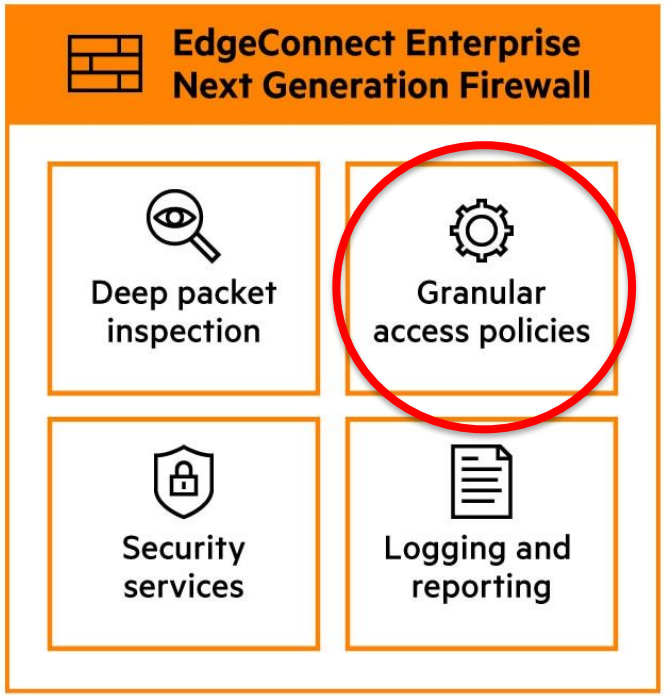
VPN awareness

Intrusion detection/ prevention

Anti-malware
Web filtering
Anti-spam
Anti-DoS

https://www.hpe.com/us/en/what-is/next-gen-firewall.html

4) Next Generation Firewalls (NGFW) add capabilities up to Layer 7



Deep packet inspection

Internet

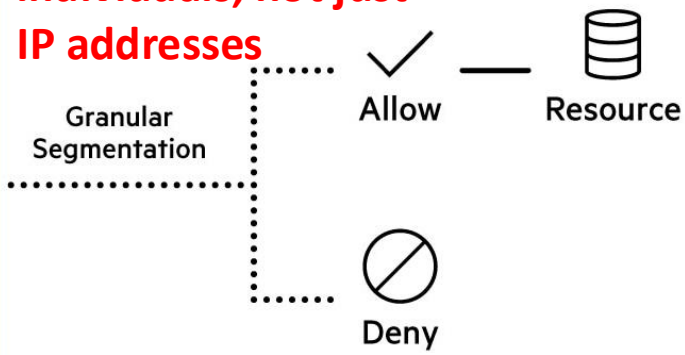
Understands Layer 7

VPN awareness

Intrusion detection/ prevention

Anti-malware
Web filtering
Anti-spam
Anti-DoS

Can map traffic to individuals, not just IP addresses



https://www.hpe.com/us/en/what-is/next-gen-firewall.html

4) Next Generation Firewalls (NGFW) add capabilities up to Layer 7



Deep packet inspection

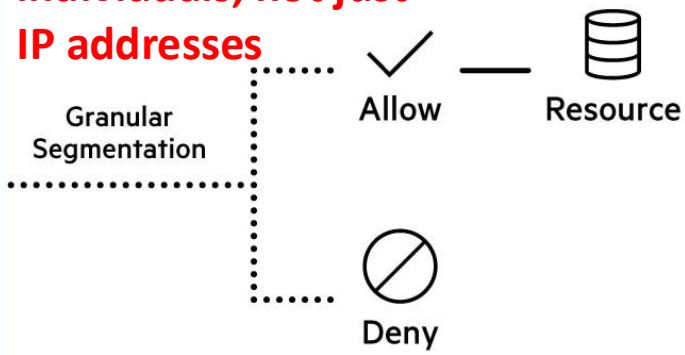
Understands Layer 7

VPN awareness

Intrusion detection/ prevention

Anti-malware
Web filtering
Anti-spam
Anti-DoS

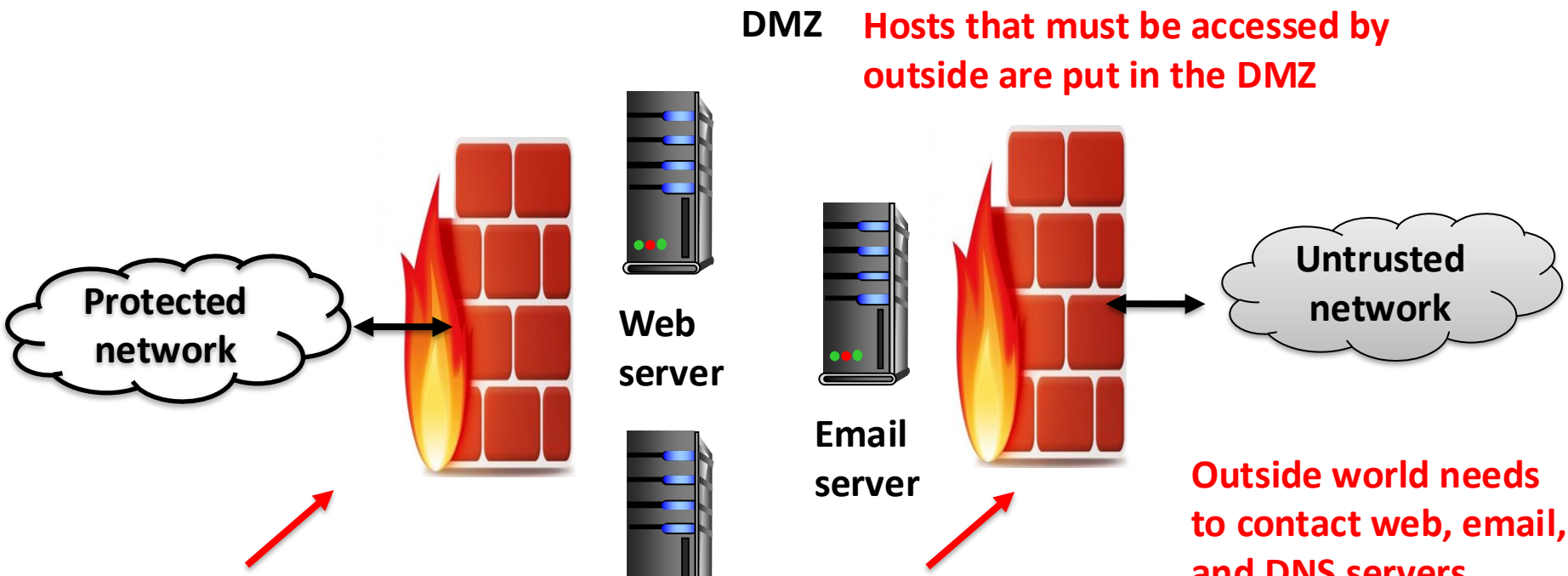
Can map traffic to individuals, not just IP addresses



Send logs to Security Information and Event Management systems (SIEM)

https://www.hpe.com/us/en/what-is/next-gen-firewall.html

Firewalls often configured to provide a Demilitarized Zone (DMZ)



Port forwarding allows access to internal hosts from outside



Port forwarding most commonly used in homes without sophisticated firewalls



Be careful opening ports on your firewall!

Most routers/firewalls block unsolicited inbound network traffic

Port forward allows traffic to a specific IP address and port to pass through firewall

Traffic forwarded to a specific host on the inside

Gives access to inside hosts to outside

Example: Set up a rule for port 25565 to forward to your Minecraft server

Agenda

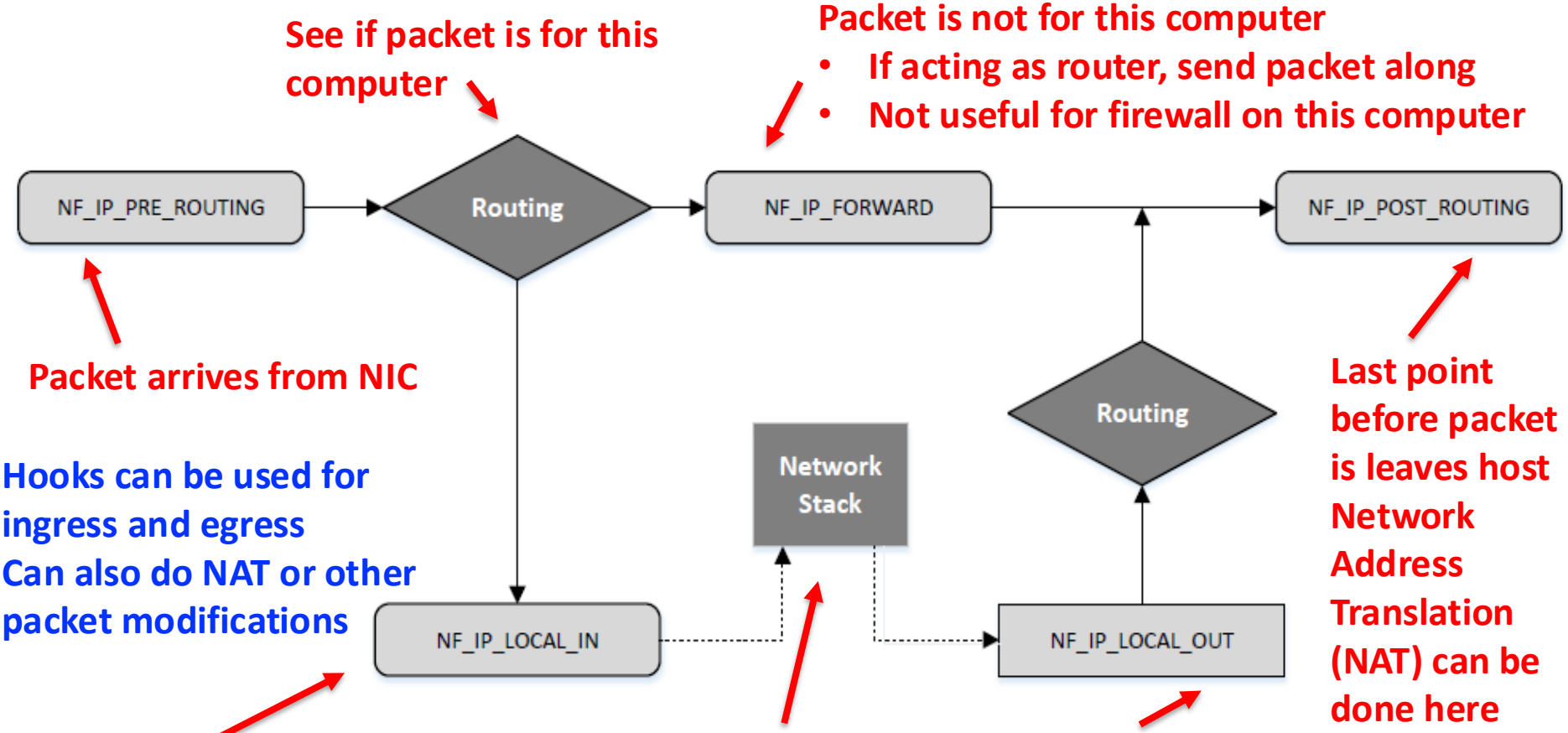
1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises

Linux provides two things useful for building a simple packet filter firewall

Packet filter firewall implementation in Linux

- Packet filtering must be done inside the kernel, user space will not be able to control packet flow
- Need changes in the kernel, two way to do this:
  - Netfilter: Provides hooks at critical points on the packet traversal path inside Linux Kernel
  - Loadable Kernel Modules: Allow privileged users to dynamically add/remove modules to the kernel, so there is no need to recompile the entire kernel

Netfilter hooks can call our code when a packet arrives at the hook



Adapted from: Internet Security: A hands on approach by Du

Code connected to a hook can render one of five decisions on a packet

1. `NF_ACCEPT`: Let the packet continue
2. `NF_DROP`: Discard the packet
3. `NF_QUEUE`: Pass the packet to the user space via nf_queue facility (async)
4. `NF_STOLEN`: typically used to store fragmented packets so related packets can be analyzed together
5. `NF_REPEAT`: Request that netfilter calls this module again

Our code must run in the kernel; use Loadable Kernel Modules (LKMs)

- Loadable Kernel Modules allow us to change the kernel without the need to recompile the entire kernel
- Developers can use LKMs to register callback functions to these hooks
- When a packet arrives at a hook
  - Netfilter checks if any kernel module has registered a callback function at this hook
  - Registered modules will be called
  - Modules are free to analyze or manipulate the packet and return the verdict on the packet
- Modules call in priority order (NF_IP_PRI_FIRST highest)

Adapted from: Internet Security: A hands-on approach by Du

LKMs have *init* and *exit* functions that fire when the modules is installed or removed

kMod.c

```
static int kmodule_init(void) {
        printk(KERN_INFO "Initializing this module\n");
        return 0;
}

static void kmodule_exit(void) {
        printk(KERN_INFO "Module cleanup\n");
}

module_init(kmodule_init);
module_exit(kmodule_exit);

MODULE_LICENSE("GPL");
```

Invoked when kernel module is loaded (sudo insmod kMod.ko)

Invoked when kernel module is removed (sudo rmmod kMod)

Note: printk, not printf!

Set init and exit modules

```
//insert our kernel module
$ make
$ sudo insmod kMod.ko

//list installed kernel modules
$ lsmod | grep kMod
kMod            16384     0

//remove our kernel module
$ sudo rmmod kMod

//check dmesg
$ dmesg | tail
[ 2969.190306] Initializing this module
[ 3005.730520] Module cleanup
```

Makefile

```
obj-m += kMod.o
all:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

-C specifies directory of library files for kernel

M indicates kernel modules

We can use netfilter and LKMs to block outgoing telnet traffic

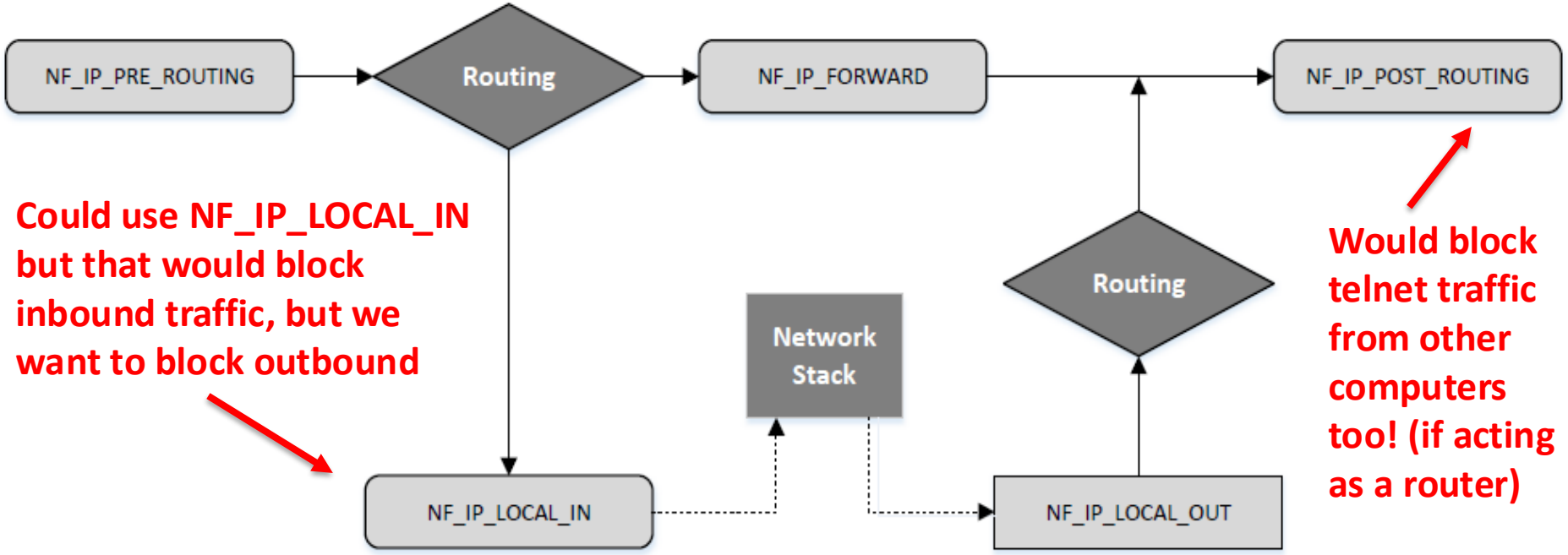
Goal: block outgoing *telnet* traffic

- TCP
- Uses port 23

We can use hooks with our LKMs to block outbound telnet traffic, but which hook?

Goal: block outgoing telnet traffic (TCP on port 23)

Which hook should we use?



Packets traverse stages

Kernel looks for hooks registered at each stage

Make callback if registered

Callback returns NF_ACCEPT, NF_DROP (or other)

Adapted from: Internet Security: A hands-on approach by Du

Implement a telnet filter with LKM

telnetFilter.c

```
static struct nf_hook_ops telnetFilterHook;

int setUpFilter(void) {
        printk(KERN_INFO "Registering a Telnet filter.\n");
        telnetFilterHook.hook = telnetFilter;
        telnetFilterHook.hooknum = NF_INET_POST_ROUTING;
        telnetFilterHook.pf = PF_INET;
        telnetFilterHook.priority = NF_IP_PRI_FIRST;

        // Register the hook.
        nf_register_net_hook(&telnetFilterHook);
        return 0;
}

void removeFilter(void) {
        printk(KERN_INFO "Telnet filter is being removed.\n");
        nf_unregister_net_hook(&telnetFilterHook);
}

module_init(setUpFilter);
module_exit(removeFilter);
```

We will fill this struct with needed data

Identify callback function (shown on next slide)

Identify hook to use (use NF_INET_LOCAL_IN if blocking inbound traffic)

IPv4

Set hook priority (first here)

Register callback

Remove callback

Set init and exit modules

Implement a telnet filter with LKM

telnetFilter.c

Hook from previous slide

```
unsigned int telnetFilter(void *priv, struct sk_buff *skb,
                          const struct nf_hook_state *state)
{
  struct iphdr *iph;
  struct tcphdr *tcph;

  iph = ip_hdr(skb);
  tcph = (void *)iph+iph->ihl*4;

  if (iph->protocol == IPPROTO_TCP && tcph->dest == htons(23)) {
    printk(KERN_INFO "Dropping telnet packet to %d.%d.%d.%d\n",
        ((unsigned char *)&iph->daddr)[0],
        ((unsigned char *)&iph->daddr)[1],
        ((unsigned char *)&iph->daddr)[2],
        ((unsigned char *)&iph->daddr)[3]);
    return NF_DROP;
  } else {
    return NF_ACCEPT;
  }
}
```

Entire packet is here

Get IP and TCP headers

TCP header begins at IP header address + header length * 32 bits (=4 bytes)

Drop if TCP and port 23, otherwise accept

Netfilter handles destroying packet (we don't have to do it ourselves)

DEMO

Primary computer (10.0.2.15)		Secondary computer (10.0.2.4)
		$ telnet 10.0.2.15 Log in with seed and dees (works) **$ exit**
# make and install firewall rule **$ cd ~/src/firewall/packet_filter/** **$ make** <builds> **$ sudo insmod telnetFilter.ko**	**Note: hook set to NF_INET_POST_ROUTING (blocking outbound traffic)**	
		$ telnet 10.0.2.15 Blocked
$ dmesg | tail Initializing this module Module cleanup Registering a Telnet filter. Dropping telnet packet to 10.0.2.15 # remove firewall rule **$ sudo rmmod telnetFilter**		
		$ telnet 10.0.2.15 Log in with seed and dees (works) **$ exit**

Agenda

1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises

Linux has a built-in firewall called *iptables* built on top of Netfilter

Netfilter

Users enter commands into iptables, iptables inserts kernel code via xtables

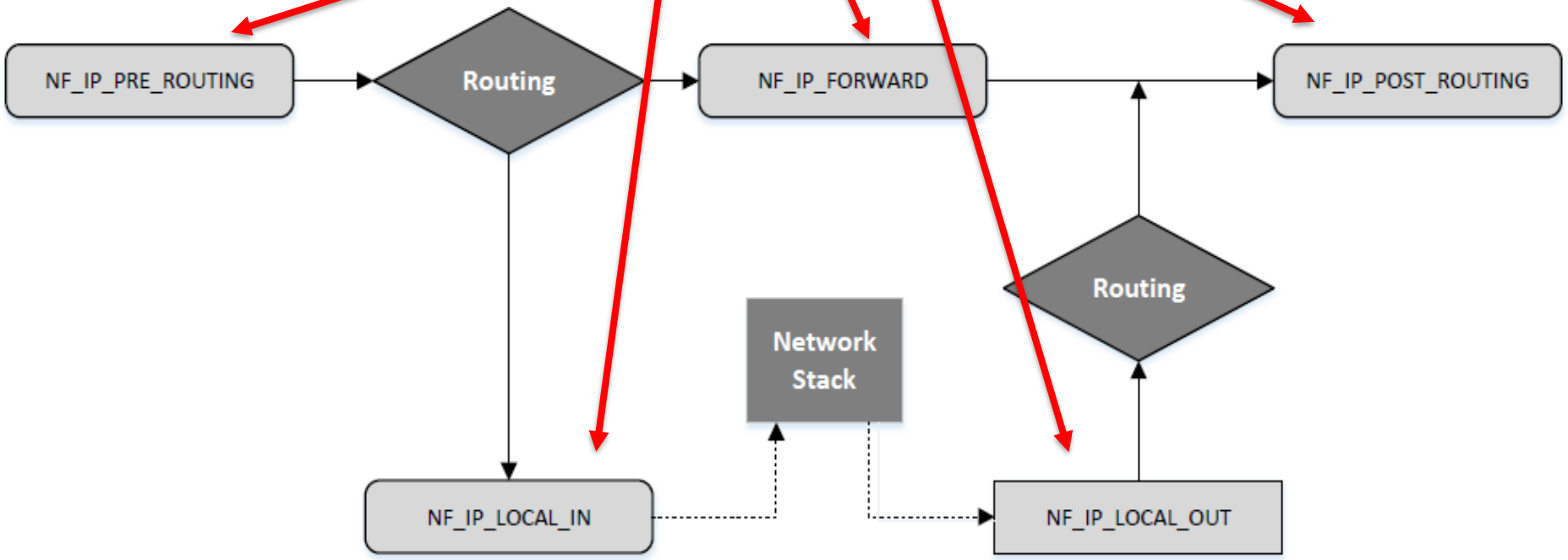
No need to write and compile kernel modules



User space: iptables

Kernel space: xtables

xtables has access to netfilter's 5 hooks



Adapted from: Internet Security: A hands-on approach by Du

iptables organizes functionality into tables and chains based on needed functionality

iptables organized into three tables

Chains correspond to netfilter hooks

INPUT = NF_IP_LOCAL_IN
FORWARD = NF_IP_FORWARD
OUTPUT = NF_IP_LOCAL_OUT

Table	Chain	Functionality
filter	INPUT FORWARD OUTPUT	Packet filtering
nat	PREROUTING INPUT OUTPUT POSTROUTING	Modifying source or destination network addresses
mangle	PREROUTING INPUT FORWARD OUTPUT POSTROUTING	Packet content modification

To do port forwarding (send packets to a different port) add hook to INPUT

There are a total of 5 hooks
But only 3 are meaningful for filtering applications
All 5 are available for changing (mangling) packets

If you want to implement a packet filter, put your rules in the filter table

If you want to change a packet, put your rules in the mangle table

If you want to do NAT….

Adapted from: Internet Security: A hands-on approach by Du

Filtering applications only need 3 hooks, other applications may need more

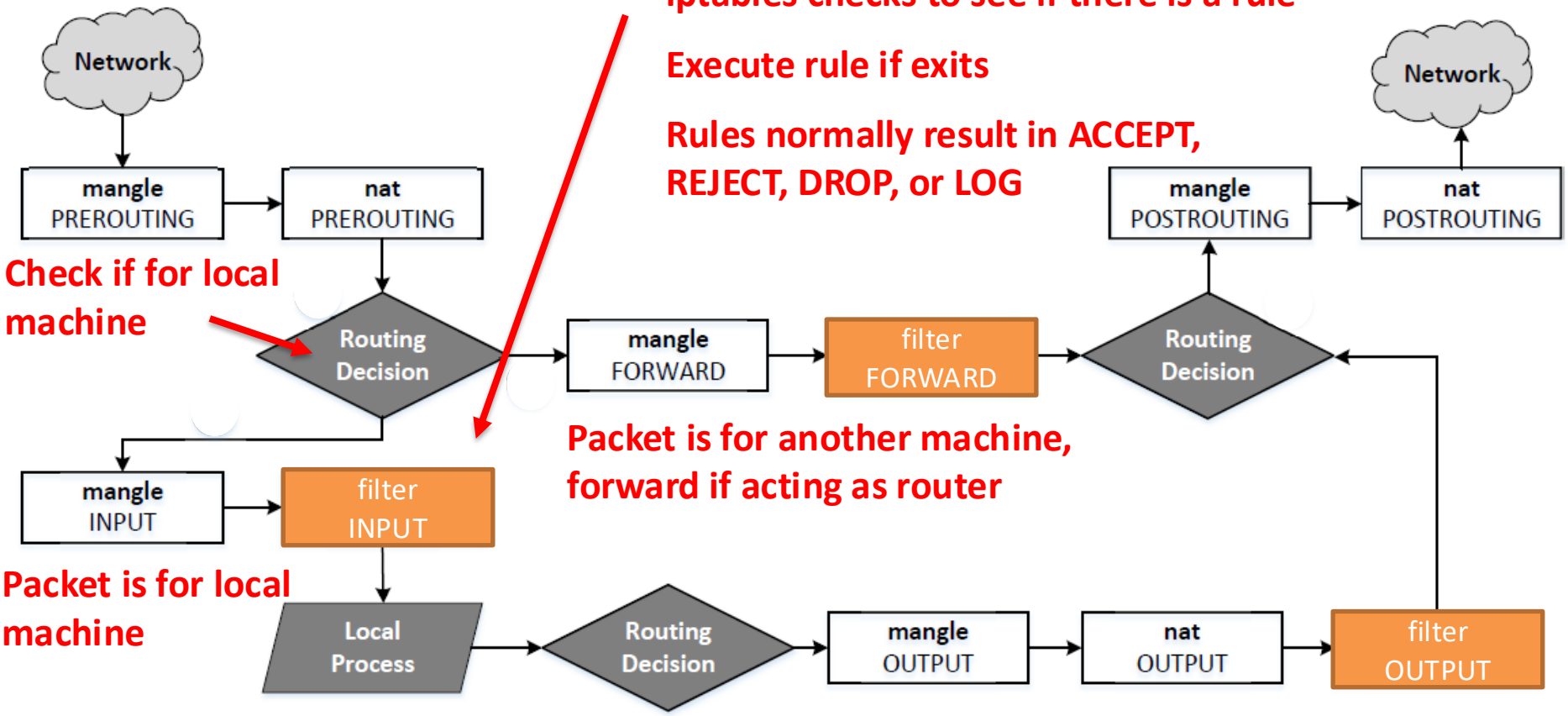
Iptables

Multiple tables per netfilter hook

As packets move through each chain iptables checks to see if there is a rule

Execute rule if exits

Rules normally result in ACCEPT, REJECT, DROP, or LOG



Adapted from: Internet Security: A hands-on approach by Du

iptables is powerful, the commands to create rules initially look difficult but aren't

Iptables

Matching rule (most complicated part)

$ iptables [-t filter] –A CHAIN <rule> -j <action>

Command

Specify table
- **filter (default)**
- **nat**
- **mangle**

Which chain and which action apply
- **INPUT**
- **FORWARD**
- **OUTPUT**
- **PRE/POST ROUTING**

Flags
- **-A = append rule**
- **-C = check if rule exists**
- **-D = delete rule**
- **-P = change policy**
- **Others**

What to do with packets that match this rule
- **ACCEPT**
- **REJECT (returns ICMP unreachable)**
- **DROP**
- **LOG**
- **Others..**

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

iptables is powerful, the commands to create rules initially look difficult but aren't

Iptables

Matching rule (most complicated part)

$ iptables [-t filter] –A CHAIN <rule> -j <action>

Specifying rule:

-i incoming interface (e.g., eth0) Layer 2
-o outgoing interface (e.g., eth1)

-s source IP (/mask) (e.g., 192.168.1.2 or 192.168.1.0/24))
-d destination IP (/mask) Layer 3

-p protocol
  -p tcp –dport 22 Layer 4
  -p udp
  -p icmp

Get help: iptables -p tcp -h

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

iptables is powerful, the commands to create rules initially look difficult but aren't

Iptables

$ iptables [-t filter] –A CHAIN <rule> -j <action>

Specifying target action:

ACCEPT
REJECT
DROP
LOG



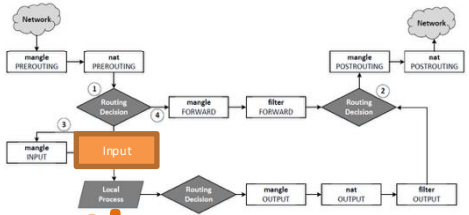
What to do with packets that match this rule

Example:
-j ACCEPT

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

Example: block a specific IP address from communicating with a network

Block IP address



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A INPUT –s 192.168.30.6 –d 192.168.0.0/16 -j DROP

Filter table used by default (if no –t)

Apply to INPUT chain

Look at packets coming from here

Going here (any address that starts with 192.168)

Drop packets like they're hot

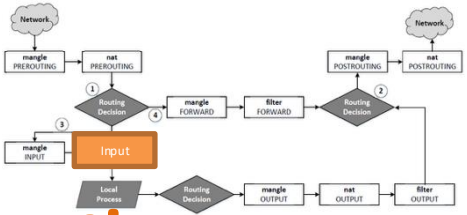
Chain choices:
PREROUTING
INPUT
FORWARD
OUTPUT
POSTROUTING

When to use a rule like this?
Quarantine host if it is known to be compromised

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

Example: accept SSH (port 22) and HTTP (port 80)

Open port 22 and 80



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A INPUT –p tcp --dport 22 –j ACCEPT
$ sudo iptables –A INPUT –p tcp --dport 80 –j ACCEPT

Accept matching packets

Apply to transport layer TCP packets

Look at ports 22 and 80 (notice double dash)

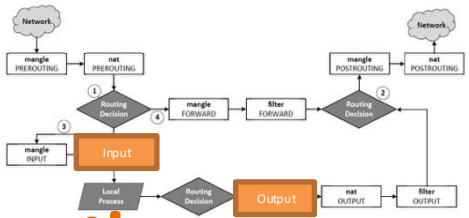
All other traffic is not necessarily rejected, packets continue on path

- **Rules evaluated in order**
- **Once a packet matches a rule and is accepted, it proceeds on**
- **Packets matching this rule will be accepted and proceed on, even if later rules would drop them**

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

We can set the default behavior to ACCEPT or DROP packets with -P

Allow all outgoing TCP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –P INPUT DROP
$ sudo iptables –P OUTPUT DROP
$ sudo iptables –P FORWARD DROP

Default now is to drop all traffic unless allowed by other rules

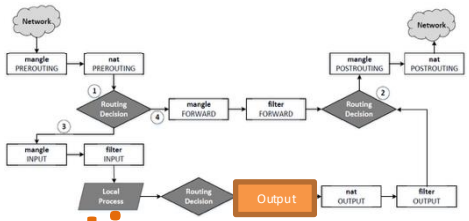
Note:

- **uses –P (policy) for default rules**
- **no –j on action**

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

Example: allow all outgoing TCP traffic

Allow all outgoing TCP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A OUTPUT –p tcp –j ACCEPT

Apply to outbound traffic

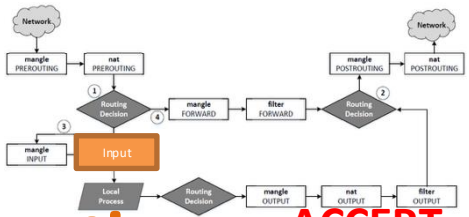
Look for TCP protocol

Accept matching packets

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

iptables can also set limits

Limit ICMP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

ACCEPT

$ sudo iptables –A INPUT -p icmp –m limit --limit 10/min --limit-burst 5 –j ACCEPT

Tables defaulted to FILTER

INPUT chain

ICMP protocol

-m limit = set a limit

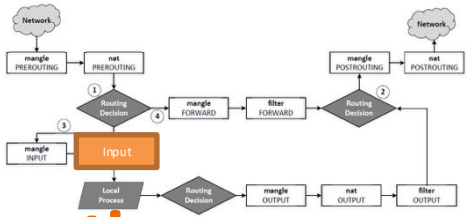
-- limit 10/min = allow 10 ICMP packets per minute

-- limit-burst = initially allow 5 packets

Adapted from: Internet Security: A hands-on approach by Du

iptables can also set limits

Limit ICMP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A INPUT -p icmp –m limit --limit 10/min --limit-burst 5 –j ACCEPT

$ sudo iptables –A INPUT -m statistic --mode-random --probability .50 –j DROP

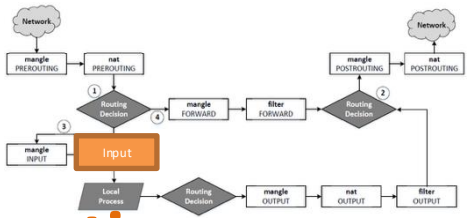
-m statistic = look at traffic statistics
-- Randomly drop packet
-- With 50% probability

Can be used for load balancing to forward packets to different servers randomly

Adapted from: Internet Security: A hands-on approach by Du

iptables can also set limits

Limit ICMP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A INPUT -p icmp –m limit --limit 10/min --limit-burst 5 –j ACCEPT

$ sudo iptables –A INPUT -m statistic --mode-random --probability .50 –j DROP

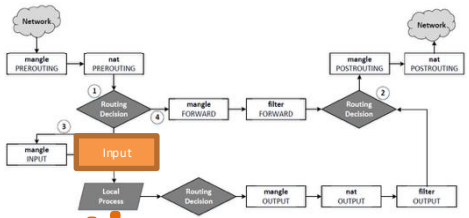
$ sudo iptables –A INPUT -m statistic --mode nth --every 3 --packet –j DROP

Drop every 3rd packet

Adapted from: Internet Security: A hands-on approach by Du

iptables can also set limits

Limit ICMP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –A INPUT -p icmp –m limit --limit 10/min --limit-burst 5 –j ACCEPT

$ sudo iptables –A INPUT -m statistic --mode-random --probability .50 –j DROP

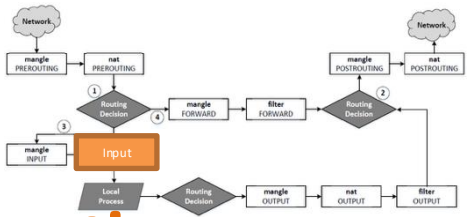
$ sudo iptables –A INPUT -m statistic --mode nth --every 3 --packet –j DROP

Can also act based on process owner

Adapted from: Internet Security: A hands-on approach by Du

Example: add five to time-to-live (ttl)

Allow all outgoing TCP traffic



$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables –t mangle –A PREROUTING –j TTL –ttl-inc 5

We are going to change packet, so use mangle table instead of filter table

Use PREROUTING so all packets are changed before moving on

Action is to increase time to live by 5

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

We can flush all rules with -F

Allow all outgoing TCP traffic

$ iptables [-t filter] –A CHAIN <rule> -j <action>

$ sudo iptables -F

Removes rules (keeps default settings)
Can be done on per table basis

https://help.ubuntu.com/community/IptablesHowTo?action=show&redirect=Iptables

We can list and remove rules

list all the rules for a table

$ sudo iptables –t filter –L

$ sudo iptables –t filter –L --line-numbers

flash (remove) all rules for a table

$ sudo iptables –t nat –F

remove a single rule

$ sudo iptables –t filter –L –line-numbers

```
Chain INPUT (policy ACCEPT)
num  target     prot opt source               destination
1    ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http

Chain FORWARD (policy ACCEPT)
num  target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
```

remove rule 1 from input chain on filter table

$ iptables –D INPUT 1

Can put commands into a shell script for convenience

iptables is extremely powerful, we've just barely scratched the surface today

To reset to default, run this shell script

clean_up.sh

```
#!/bin/sh

# Set up all the default policies to ACCEPT packets
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

#Flush all existing configurations.
iptables -F
```


https://github.com/kevin-w-du/BookCode/blob/master/Firewall/iptables/cleanup.sh

Agenda

1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises

Looking at the context of a packet can provide better control than just filtering

```
//open port 22 (ssh) and 80 (http)
$ sudo iptables –A INPUT –p tcp --dport 22 –j ACCEPT
$ sudo iptables –A INPUT –p tcp --dport 80 –j ACCEPT
```

This examples allows communication over port 22 and 80
An adversary can send TCP packets out port 80, but to clients other than one that has established a connection
Add stateful inspect to prevent communication to clients that have not completed 3-way handshake
States are:

- NEW: Connection starting 3-way TCP handshake
- ESTABLISHED: Connection established
- RELATED: Establishes relationship between connections
- INVALID: Used for packets that do not follow protocol

Looking at the context of a packet can provide better control than just filtering

```
//open port 22 (ssh) and 80 (http)
$ sudo iptables –A INPUT –p tcp --dport 22 –j ACCEPT
$ sudo iptables –A INPUT –p tcp --dport 80 –j ACCEPT
$ sudo iptables –A INPUT –p tcp –m conntrack
                        --ctstate ESTABLISHED, RELATED – j ACCEPT
```

This examples allows communication over port 22 and 80

An adversary can send TCP packets out port 80, but to clients other than one that has established a connection

Add stateful inspect to prevent communication to clients that have not completed 3-way handshake

States are:

- NEW: Connection starting 3-way TCP handshake
- ESTABLISHED: Connection established
- RELATED: Establishes relationship between connections
- INVALID: Used for packets that do not follow protocol

Conntrack package for iptables allow state tracking

Looking at the context of a packet can provide better control than just filtering

```
//allow SYN packets to port 22 and 80
$ sudo iptables –A FORWARD –p tcp --dport 22 --syn
     -m conntrack –ctstate NEW  –j ACCEPT
$ sudo iptables –A FORWARD –p tcp --dport 80 --syn
     -m conntrack –ctstate NEW  –j ACCEPT
//allow TCP if they belong to an existing connection
$ sudo iptables –A FORWARD –p tcp  -m conntrack
     --ctstate ESTABLISHED, RELATED  –j ACCEPT
//set default
$ sudo iptables –P FORWARD DROP
```

States are:

- NEW: Connection starting 3-way TCP handshake
- ESTABLISHED: Connection established
- RELATED: Establishes relationship between connections
- INVALID: Used for packets that do not follow protocol

Agenda

1. Threats
2. What are firewalls?
3. Building a simple firewall using Netfilter
4. Using iptables firewall
5. Stateful firewall
6. Exercises



Exercises

Write iptables rules to:

- Block all inbound traffic to the host running iptables
- Block all inbound traffic to 192.168.1.0/24 subnet
- Allow only SSH (port 22) traffic from a specific IP (e.g., 192.168.1.100), while blocking all other inbound traffic
- Reject access to a specific domain (e.g., example.com) using its IP address
- Allow established and related connections, block new ones

Exercise Solutions

Write iptables rules to:

- Block all inbound traffic to the host running iptables
  iptables -P INPUT DROP

- Block all inbound traffic to a /24 subnet
  iptables -A FORWARD -d 192.168.1.0/24 -j DROP

- Allow only SSH (port 22) traffic from a specific IP (e.g., 192.168.1.100), while blocking all other inbound traffic
  iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT
  iptables -A INPUT -j DROP

- Block access to a specific domain (e.g., example.com) using its IP address
  iptables -A OUTPUT -d 93.184.216.34 -j REJECT

- Allow established and related connections, block new ones
  iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
  iptables -A INPUT -m conntrack --ctstate NEW -j DROP