CSE 127: Introduction to Security

Lecture 14: Network Defenses

Deian Stefan

UCSD

Fall 2020

Material from Nadia Heninger, Stefan Savage, David Wagner, and Zakir Durumeric

Defending Networks

- How do you harden a set of systems against external attack?
  - The more network services your machines run, the greater the risk (i.e., the attack surface is larger)
- One approach: Turn off unnecessary network services on each system
- Why is this hard?

Defending Networks

- How do you harden a set of systems against external attack?
  - The more network services your machines run, the greater the risk (i.e., the attack surface is larger)
- One approach: Turn off unnecessary network services on each system
- Why is this hard?
  - Requires knowing all the services that are running
  - What if you have hundreds or thousands of systems?
  - Systems may have different OSes, hardware, and users

Network Perimeter Defense

- Idea: Network defenses on “outside” of organization (e.g. between org and Internet)
  - Assumption?
- Typical elements:
  - Firewalls
  - Network Address Translation
  - Application Proxies
  - Network Intrusion Detection Systems (NIDS)

Firewalls

- Problem: Protecting or isolating one part of the network from other parts
  - Typically: Protect your network from global Internet
  - Sometimes: Protect Internet from infected machines in your network
- Need to filter or otherwise limit network traffic
- Questions:
  - What information do you use to filter?
  - Where do you do the filtering?

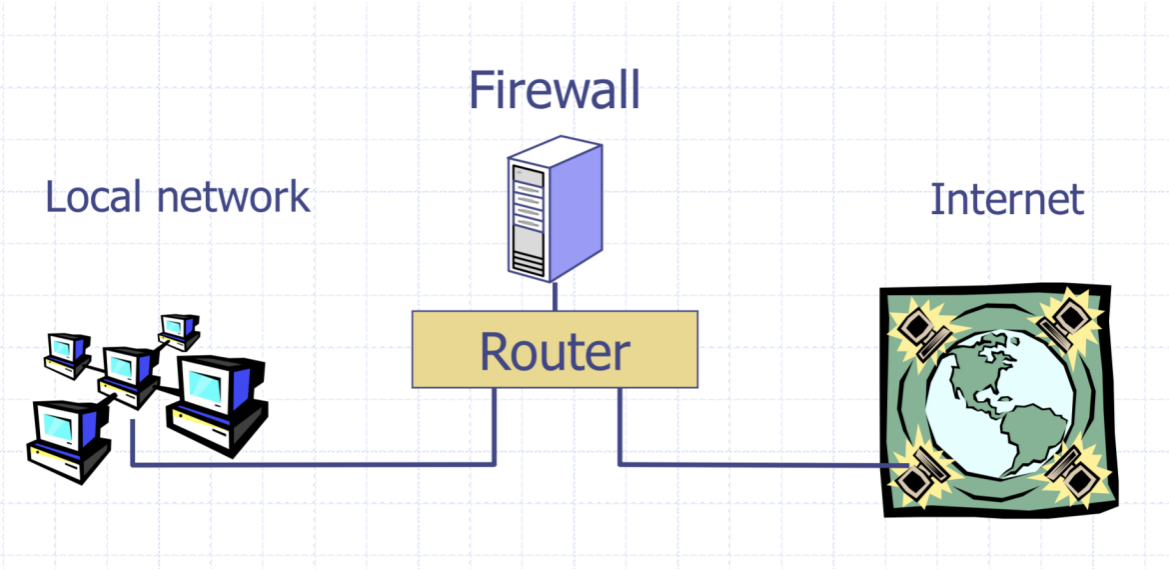
Kinds of Firewalls

- Personal firewalls
  - Run on end-hosts
  - Has application/user-specific information
- Network firewalls
  - Intercept communications from many hosts

Kinds of Firewalls

- Personal firewalls
  - Run on end-hosts
  - Has application/user-specific information
- Network firewalls
  - Intercept communications from many hosts
- Filter-based
  - Operates by filtering on packet headers
- Proxy-based
  - Operates at the level of the application
  - e.g. HTTP web proxy

Network Firewalls



- Filters protect against “bad” communications.
- Protect services offered internally from outside access.
- Provide outside services to hosts located inside.

Access Control Policies

- A firewall enforces an access control policy
  - Who is talking to whom and accessing what service?
- Distinguish btw inbound and outbound connections
  - Inbound:
  - Outbound:

Access Control Policies

- A firewall enforces an access control policy
  - Who is talking to whom and accessing what service?
- Distinguish btw inbound and outbound connections
  - Inbound: Attempts by external users to connect to services on internal machines
  - Outbound: Internal users to external services
- Conceptually simple access control policy:
  - Permit users inside to connect to any service
  - External users are restricted
    - Allow connections to services meant to be external
    - Deny connections to services not meant to be external

Access Control Policies

How to treat traffic not mentioned in policy?

Default allow

-

Default deny

-

In general, default deny is safer. Why?

-
-

Access Control Policies

How to treat traffic not mentioned in policy?

Default allow

- Permit all services, shut off for specific problems

Default deny

- Permit only a few well-known services

In general, default deny is safer. Why?

- Conservative design
- Flaws in default deny get noticed more quickly

Example Firewall Policy

- Configure: Only allow SSH.

```
# ufw default deny
# ufw allow from 100.64.0.0/24
# ufw allow ssh
```

- Status: Only allow SSH.

```
# ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
Anywhere                   ALLOW       100.64.0.0/24
22 (v6)                    ALLOW       Anywhere (v6)
```

Packet Filtering Firewalls

- Define list of access-control rules
- Check every packet against rules and forward or drop
- Packet-filtering firewalls can take advantage of the following information from network and transport layer headers:
  - Source IP
  - Destination IP
  - Source Port
  - Destination Port
  - Flags (e.g. ACK)

Example packet filtering rules

- Block incoming DNS (port 53) except known trusted servers
- Block incoming HTTPS (port 443) except to company IP addresses
- Block forged internal addresses

Limitations of stateless filtering:

-

Some firewalls keep state about open TCP connections.

-

Example packet filtering rules

- Block incoming DNS (port 53) except known trusted servers
- Block incoming HTTPS (port 443) except to company IP addresses
- Block forged internal addresses

Limitations of stateless filtering:

- A stateless packet filter can’t distinguish packets associated with a connection from those that are not.

Some firewalls keep state about open TCP connections.

- Allows conditional filtering rules of the form “if internal machine has established the TCP connection, permit inbound reply packets”.

Circumventing simple firewall rules

Idea 1: Send traffic on a port usually allocated for another service.

Circumventing simple firewall rules

Idea 1: Send traffic on a port usually allocated for another service.

Idea 2: Tunneling

- Encapsulate one protocol inside another
- Recipient of outer protocol decapsulates to recover inner protocol
- Examples:
  - iodine IP over DNS
  - ssh tunnel
  - VPN (Virtual Private Network)

Stateful Packet Filtering Example

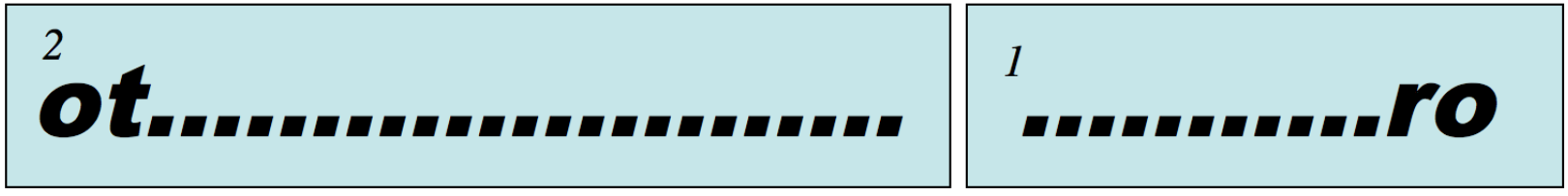
Suppose you want to allow inbound connections to a server, but block any attempts to log in as “root”.

- How would you do this?
- What state do you need to keep?

Stateful Packet Filtering is Hard

- Sender might and try to sneak through firewall
- “root” might span packet boundaries

- Packets might be reordered

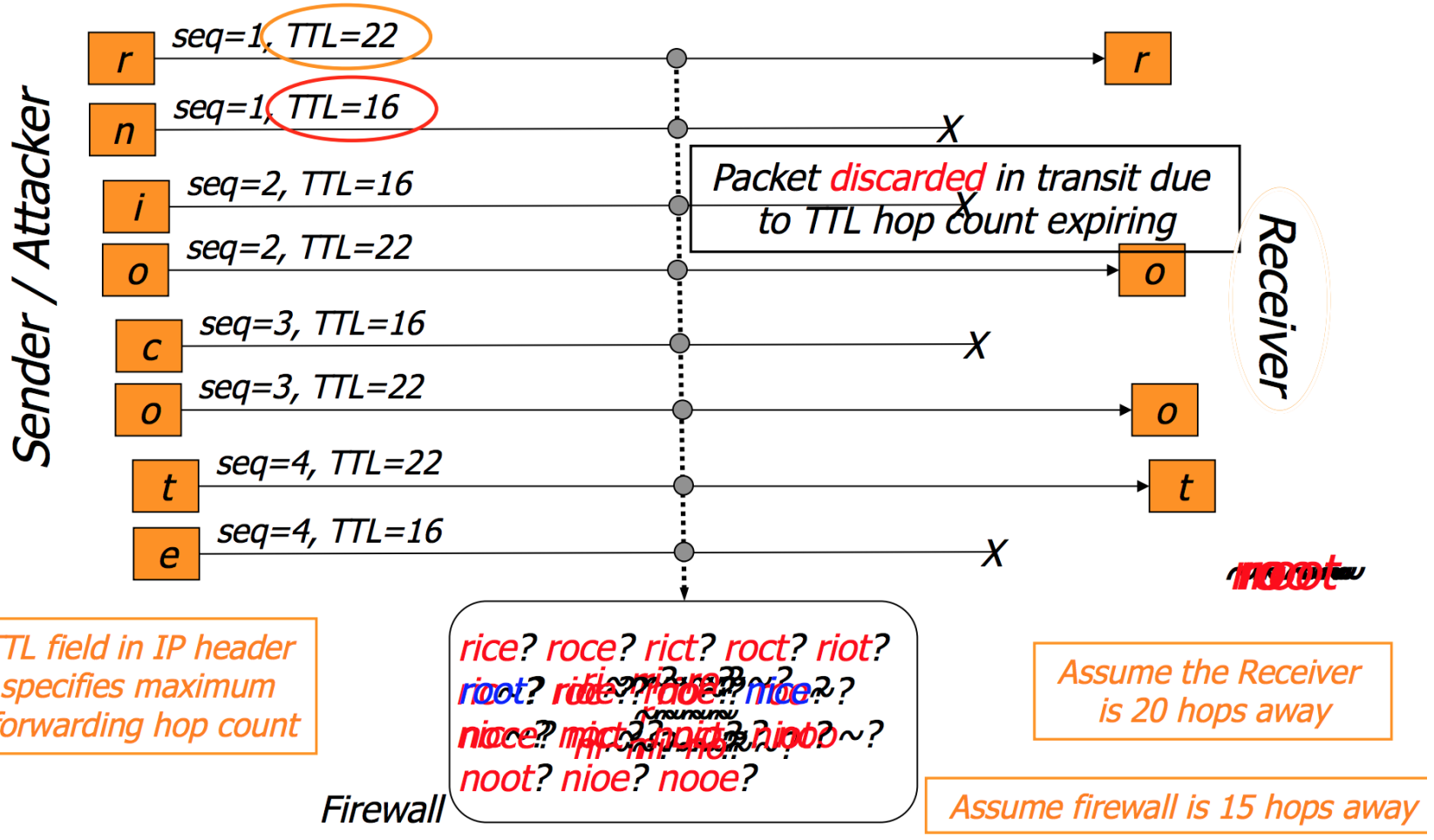


Stateful Packet Filtering is Hard

- Can send more packets than will make it to host (TTL evasion)

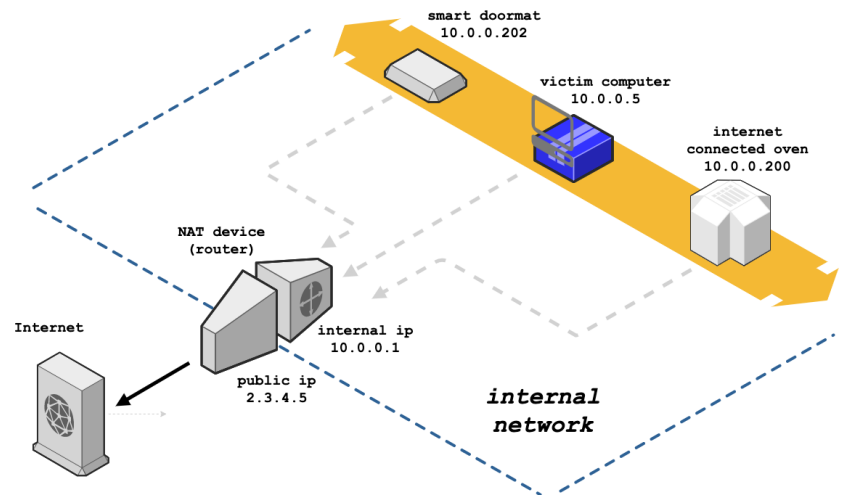
Stateful Packet Filtering is Hard

- Can send more packets than will make it to host (TTL evasion)



Network Address Translation (NAT)

- Idea: IP addresses do not need to be globally unique
- NATs map between two different address spaces.
- Most home routers are NATs and firewalls.



https://samy.pl/slipstream/

Private Subnets

10.0.0.0–10.255.255.255
172.16.0.0–172.31.255.255
192.168.0.0–192.168.255.255

Typical NAT Behavior

- NAT maintains a table of the form:
  `<client IP> <client port> <NAT ID>`
- Outgoing packets (on non-NAT port):
- Incoming packets (on NAT port)

Typical NAT Behavior

- NAT maintains a table of the form:
  `<client IP> <client port> <NAT ID>`
- Outgoing packets (on non-NAT port):
  - Look for client IP address, client port in mapping table
  - If found, replace client port with previously allocated NAT ID (same size as port number)
  - If not found, allocate a new NAT ID and replace source port with NAT ID
  - Replace source address with NAT address
- Incoming packets (on NAT port)
  - Look up destination port as NAT ID in port mapping table
  - If found, replace destination address and port with client entries from the mapping table
  - If not found, the packet should be rejected
- Table entries expire after 2–3 minutes of no activity to allow them to be garbage collected

NAT Pros and Cons

- Profs
  - Only allows connections to the outside that are established from inside.
    - Hosts from outside can only contact internal hosts that appear in the mapping table, and they're only added when they establish the connection.
  - Don't need as large an external address space
    - i.e. 10 machines can share 1 IP address

- Costs
  - Rewriting IP addresses isn't so easy.
    - IP addresses may appear in the content of the packet in some protocols like FTP.
  - Breaks some protocols
    - e.g. some streaming protocols have client invoke server and then server opens a new connection to the client

NAT Slipstreaming (10/31/2020)

“NAT Slipstreaming allows an attacker to remotely access any TCP/UDP service bound to a victim machine, bypassing the victim’s NAT/firewall (arbitrary firewall pinhole control), just by the victim visiting a website.”

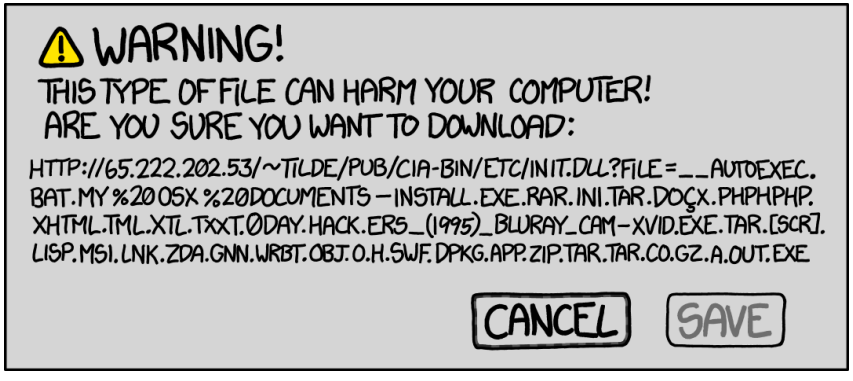
`https://samy.pl/slipstream/`

Application Proxies

Idea: Control apps by requiring them to pass through proxy

- Proxy is application-level man-in-the-middle
- Enforce policy for specific protocols:
  - SMTP: Scan for viruses, reject spam
  - SSH: Log authentication, inspect encrypted text
  - HTTP: Block forbidden URLs

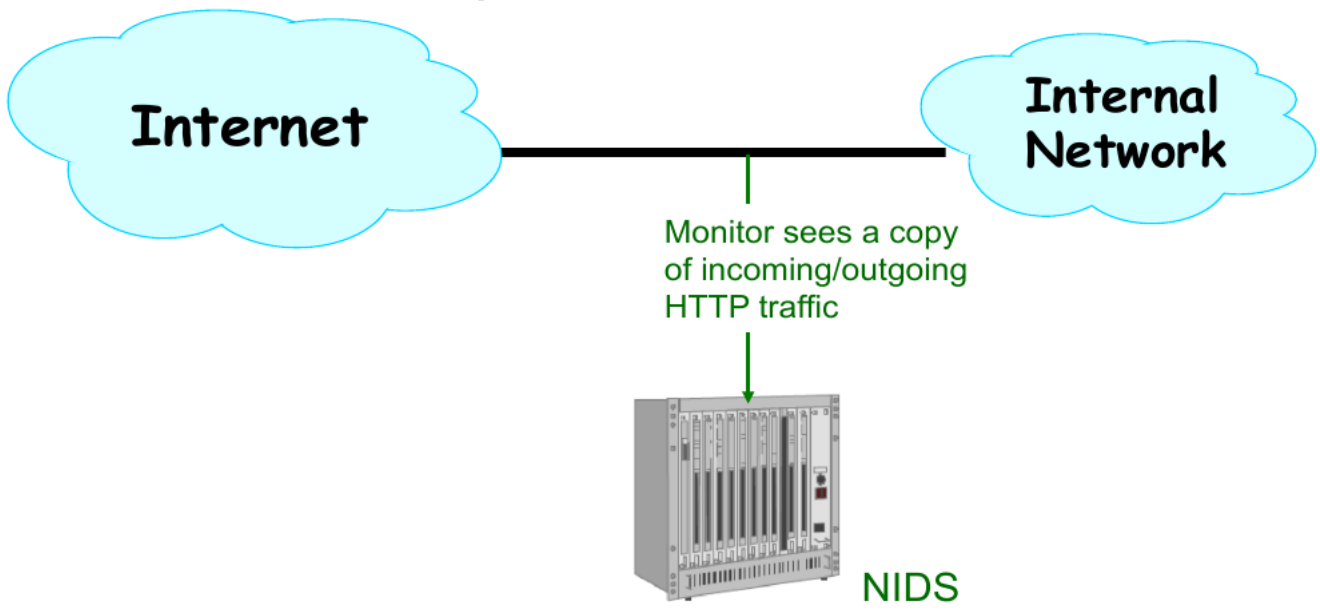
Companies inspect outbound traffic, will install root certificates on employee workstations to monitor TLS traffic.



XKCD

Network Intrusion Detection Systems (NIDS)

- Idea: Passively monitor network traffic for signs of attack (e.g., look for `/etc/passwd`)



Network Intrusion Detection Systems (NIDS)

- NIDS has a table of all active connections, and maintains state for each
  - E.g., has it seen partial match of `/etc/passwd`
- What do you do when you see a new packet not associated with any known connection?

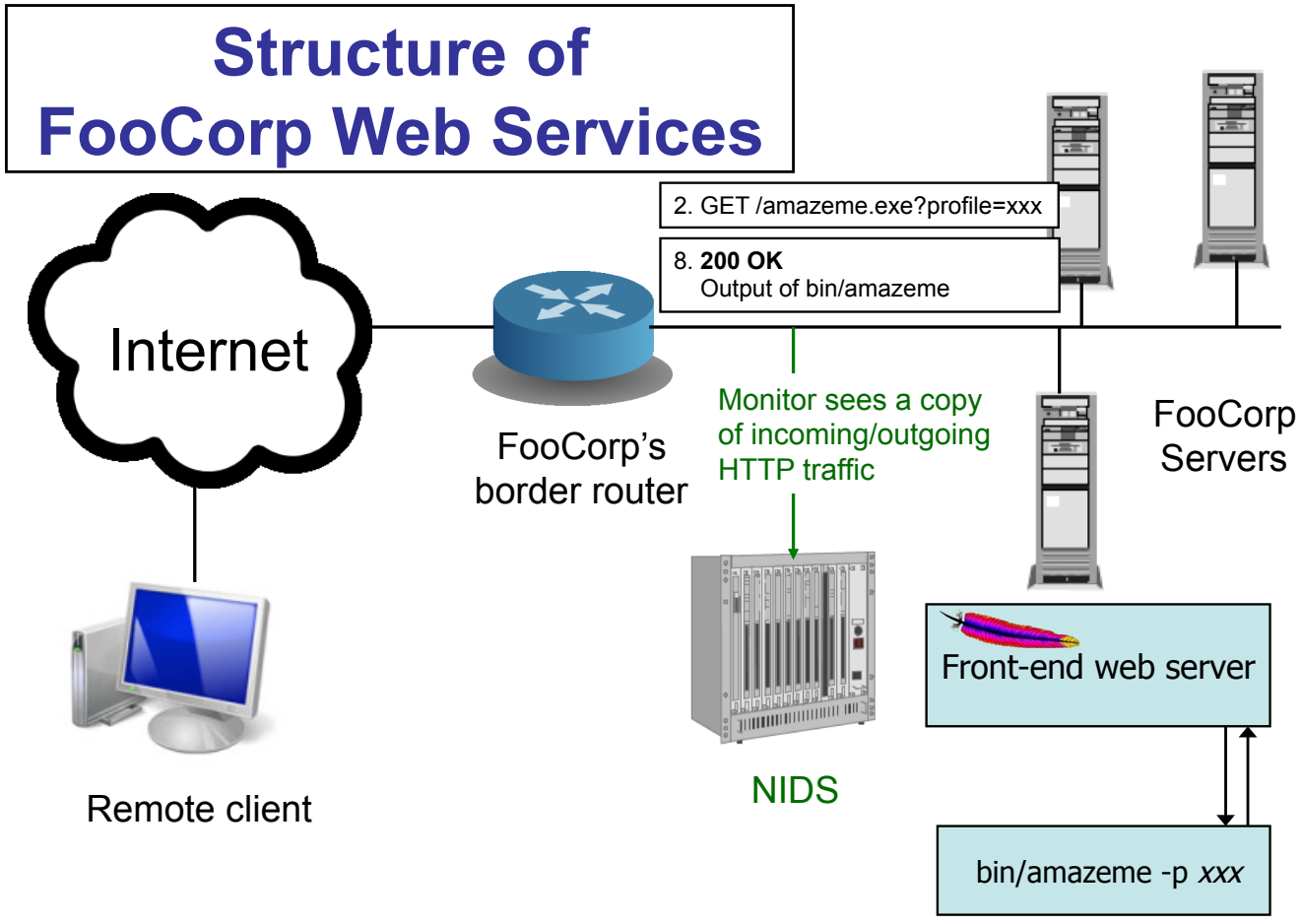
Network Intrusion Detection Systems (NIDS)

- NIDS has a table of all active connections, and maintains state for each
  - E.g., has it seen partial match of `/etc/passwd`
- What do you do when you see a new packet not associated with any known connection?
  - Create a new connection: when NIDS starts, it doesn't know what connections might be existing

Network Intrusion Detection Systems (NIDS)

- NIDS has a table of all active connections, and maintains state for each
  - E.g., has it seen partial match of `/etc/passwd`
- What do you do when you see a new packet not associated with any known connection?
  - Create a new connection: when NIDS starts, it doesn't know what connections might be existing
- Where should you do the detection?
  - Network, host, or both?

Approach #1: Network-based Detection



- Look at network traffic, scanning HTTP requests
  - E.g., look for `/etc/password` or `../../`

Network-based Detection Pros and Cons

Benefits

- Don’t need to *modify* or *trust* end systems
- Cover many systems with single monitor
- Centralized management

Network-based Detection Pros and Cons

Benefits

- Don't need to *modify* or *trust* end systems
- Cover many systems with single monitor
- Centralized management

Issues

- Expensive: 10Gbps link ≈ 1M packets/second ≈ ns/packet

Network-based Detection Pros and Cons

Benefits

- Don't need to *modify* or *trust* end systems
- Cover many systems with single monitor
- Centralized management

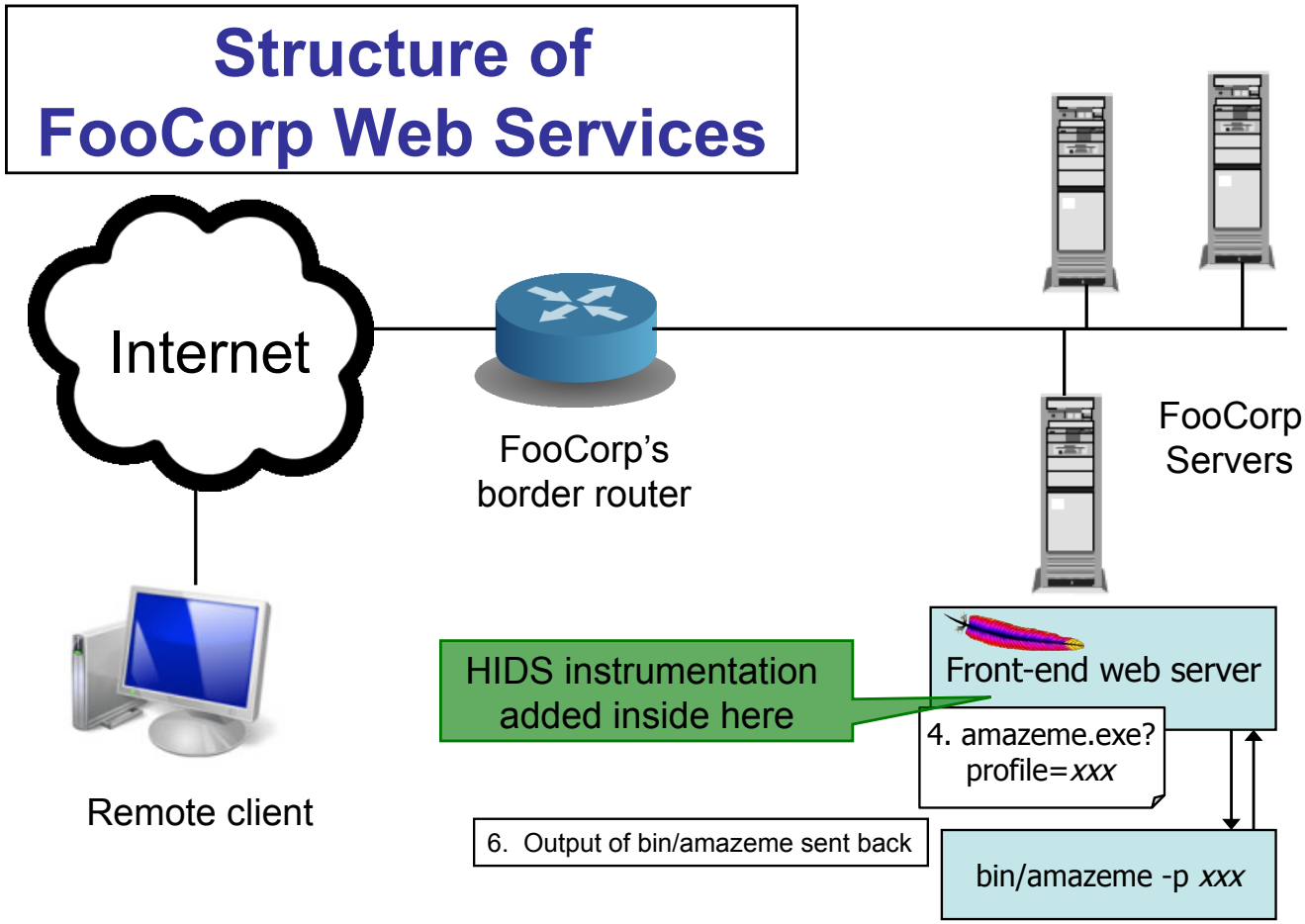
Issues

- Expensive: 10Gbps link ≈ 1M packets/second ≈ ns/packet
- Vulnerable to evasion attacks
  - Some evasions reflect incomplete analysis
    - E.g., hex escape or `../%2f/%2f../%2f/%2f`
    - In principle, can deal with these with implementation care
  - Some are due to imperfect observability
    - E.g., what if what NIDS sees doesn't exactly match what arrives at destination?

Understanding the Downsides

- Does `/etc/passwd` exist on all systems? Do you include rules for all OSes?
- Are all requests with `../../` *necessarily* bad?
  - False positives: Sometimes seen in legit requests
- Do you handle all encodings and semantic meaning?
  - Evasion: Abusing URL encodings (`%2e%2e%2f%2e%2e%2f`)
  - Evasion: Abusing UNIX semantics (`..///.///..////`)
- What if the traffic is encrypted (HTTPS)?
  - Need access to session key or decrypted text
  - Why might you not want to give the NIDS your TLS keys?

Approach #2: Host-based Detection



- Instrument web server, scan arguments sent to back-end programs (and outbound requests)
  - E.g., look for `/etc/password` or `../../`

Host-based Detection Pros and Cons

Benefits

- The semantic gap is smaller, have understanding of URLs (and thus `%2e%2e%2f%2e%2e%2f`)
- Don’t need to intercept HTTPS

Host-based Detection Pros and Cons

Benefits

- The semantic gap is smaller, have understanding of URLs (and thus `%2e%2e%2f%2e%2e%2f`)
- Don't need to intercept HTTPS

Issues

- Expensive: Add code to each server
- Still have to consider e.g., UNIX filename semantics `../../../../`
- Still have to consider other sensitive files, databases, etc.

Host-based Detection Pros and Cons

Benefits

- The semantic gap is smaller, have understanding of URLs (and thus `%2e%2e%2f%2e%2e%2f`)
- Don't need to intercept HTTPS

Issues

- Expensive: Add code to each server
- Still have to consider e.g., UNIX filename semantics `../////.////..////`
- Still have to consider other sensitive files, databases, etc.
- Only (kind of) helps with web server attacks; what do you do about other end systesm?

Example: arpwatch

Fwd: flip flop (elk.sysnet.ucsd.edu) eno1 Inbox ×

Cindy Moore 11:33 AM (52 minutes ago)

to Deian, Riad

Anything in particular going on? I should probably check with you guys on elk's status?

---------- Forwarded message ---------
From: **Arpwatch sysnet.sysnet.ucsd.edu** <arpwatch@sysnet.sysnet.ucsd.edu>
Date: Sat, Nov 9, 2019 at 12:23 PM
Subject: flip flop (elk.sysnet.ucsd.edu) eno1
To: <root@sysnet.sysnet.ucsd.edu>

hostname: elk.sysnet.ucsd.edu
ip address: 137.110.222.162
interface: eno1
ethernet address: c2:50:dd:1e:64:c8
ethernet vendor: <unknown>
old ethernet address: ac:1f:6b:8d:2f:88
old ethernet vendor: <unknown>
timestamp: Saturday, November 9, 2019 12:23:15 -0800
previous timestamp: Saturday, November 9, 2019 12:20:28 -0800
delta: 2 minutes

die.net
Site Search
Library
linux docs
linux man pages
page load time
Toys
world sunlight
moon phase
trace explorer

arpwatch(8) - Linux man page

Name

arpwatch - keep track of ethernet/ip address pairings

Synopsis

arpwatch [**-dN**] [**-f** *datafile*] [**-i** *interface*]

[**-n** *net*[/*width*]] [**-r** *file*] [**-u** *username*] [**-e** *username*] [**-s** *username*]

Description

Arpwatch keeps track for ethernet/ip address pairings. It syslogs activity and reports certain changes via email. **Arpwatch** uses **pcap**(3) to listen for arp packets on a local ethernet interface.

The **-d** flag is used enable debugging. This also inhibits forking into the background and emailing the reports. Instead, they are sent to *stderr*.

The **-f** flag is used to set the ethernet/ip address database filename. The default is *arp.dat*.

The **-i** flag is used to override the default interface.

The **-n** flag specifies additional local networks. This can be useful to avoid "bogon" warnings when there is more than one network running on the same wire. If the optional *width* is not specified, the default netmask for the network's class is used.

The **-N** flag disables reporting any bogons.

The **-r** flag is used to specify a savefile (perhaps created by **tcpdump**(1) or **pcapture**(1)) to read from instead of reading from the network. In this case, **arpwatch** does not fork.

If **-u** flag is used, **arpwatch** drops root privileges and changes user ID to *username* and group ID to that of the primary group of *username*. This is recommended for security reasons.

If the **-e** flag is used, **arpwatch** sends e-mail messages to *username* rather than the default (root). If a single '-' character is given for the username, sending of e-mail is suppressed, but logging via syslog is still done as usual. (This can be useful during initial runs, to collect data without being flooded with messages about new stations.)

Approach #3: Log analysis

- Log analysis: run scripts to analyze system log files (e.g., every night, hour, etc.)

  Benefits
  - Cheap: Servers already have logging facilities
  - No escaping issues (logging done by server)

Approach #3: Log analysis

- Log analysis: run scripts to analyze system log files (e.g., every night, hour, etc.)

Benefits	• Cheap: Servers already have logging facilities • No escaping issues (logging done by server)
Issues	• Reactive: detection delayed, can't block attacks

Approach #3: Log analysis

- Log analysis: run scripts to analyze system log files (e.g., every night, hour, etc.)

  Benefits
  - Cheap: Servers already have logging facilities
  - No escaping issues (logging done by server)

  Issues
  - Reactive: detection delayed, can't block attacks
  - Still need to worry about UNIX filename semantics

Approach #3: Log analysis

- Log analysis: run scripts to analyze system log files (e.g., every night, hour, etc.)

Benefits	• Cheap: Servers already have logging facilities • No escaping issues (logging done by server)
Issues	• Reactive: detection delayed, can't block attacks • Still need to worry about UNIX filename semantics • Malware may be able to modify logs

Example: fail2ban

Fail2ban *scans log files (e.g.* `/var/log/apache/error_log`*) and bans IPs that show the malicious signs – too many password failures, seeking for exploits, etc. Generally Fail2Ban is then used to update firewall rules to reject the IP addresses for a specified amount of time, although any arbitrary other* ***action*** *(e.g. sending an email) could also be configured. Out of the box Fail2Ban comes with* ***filters*** *for various services (apache, courier, ssh, etc).*

```
┌[d@elk ~ ]
└─➤  sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 114
|  |- Total failed:     387
|  `- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 7
   |- Total banned:     10
   `- Banned IP list:   159.192.137.41 159.192.137.43 3.83.151.154 61.19.193.158 64.39
.111.121 64.39.111.138 92.23.95.101
┌[d@elk ~ ]
└─➤
```

But this has its own issues

- Filters are complicated regular expressions
- Can accidentally block self
- Can be tricked into blocking others

```
# !!! WARNING !!!
#   Since UDP is connection-less protocol, spoofing of IP and imitation
#   of illegal actions is way too simple.  Thus enabling of this filter
#   might provide an easy way for implementing a DoS against a chosen
#   victim. See
#    http://nion.modprobe.de/blog/archives/690-fail2ban-+-dns-fail.html
#   Please DO NOT USE this jail unless you know what you are doing.
#
# IMPORTANT: see filter.d/named-refused for instructions to enable logging
# This jail blocks UDP traffic for DNS requests.
# [named-refused-udp]
#
# filter   = named-refused
# port     = domain,953
# protocol = udp
# logpath  = /var/log/named/security.log

# IMPORTANT: see filter.d/named-refused for instructions to enable logging
# This jail blocks TCP traffic for DNS requests.
```

Detection Accuracy

- Two types of detector errors:
  - False positives (FPs): alerting about a non-problem
  - False negatives (FNs): failing to alert about a real problem
- Detector accuracy is often addressed in terms of rates:
  - Let I be the event of an instance of intrusive behavior
  - Let A be the event of detector generating an alarm
  - We then define: FP rate $= P[A|\neg I]$ and FN rate $= P[\neg A|I]$
- Can we build a perfect detector?

Detection Tradeoffs

- The art of a good detector is achieving **effective balance** between FP and FN rate.
  - Is low FP rate more better than low FN rate?
  - Is an FP rate of 0.1% and FN rate of 2% good?
- It depends...

Detection Tradeoffs

- The art of a good detector is achieving **effective balance** between FP and FN rate.
  - Is low FP rate more better than low FN rate?
  - Is an FP rate of 0.1% and FN rate of 2% good?
- It depends…
  - on cost of each type of error (e.g., FPs can waste an engineer's time; FN might lead to huge clean up fee)
  - on rate at which attacks occur (e.g., your laptop vs. Google's servers)

Vulnerability Scanning

Idea: Rather than detect attacks, launch them yourself.

- Probe internal systems with a range of attacks
- Patch/fix/block any that succeed.
- Pros:
  - Accurate: If your scanning tool is good, it finds real problems
  - Proactive: Can prevent future misuse
  - Intelligence: Can ignore IDS alarms you know can’t succeed
- Issues:

Vulnerability Scanning

Idea: Rather than detect attacks, launch them yourself.

- Probe internal systems with a range of attacks
- Patch/fix/block any that succeed.
- Pros:
  - Accurate: If your scanning tool is good, it finds real problems
  - Proactive: Can prevent future misuse
  - Intelligence: Can ignore IDS alarms you know can't succeed
- Issues:
  - Can take a lot of work
  - Not helpful for systems you can't modify
  - Dangerous for disruptive attacks

Vulnerability Scanning

Idea: Rather than detect attacks, launch them yourself.

- Probe internal systems with a range of attacks
- Patch/fix/block any that succeed.
- Pros:
  - Accurate: If your scanning tool is good, it finds real problems
  - Proactive: Can prevent future misuse
  - Intelligence: Can ignore IDS alarms you know can't succeed
- Issues:
  - Can take a lot of work
  - Not helpful for systems you can't modify
  - Dangerous for disruptive attacks
- In practice, this approach is prudent and widely used.
- Good complement to running an IDS

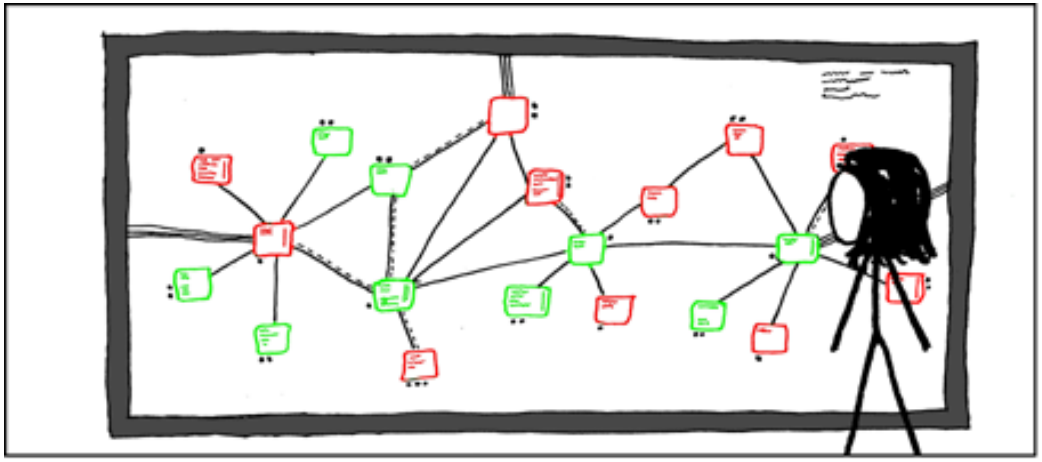
Honeypots

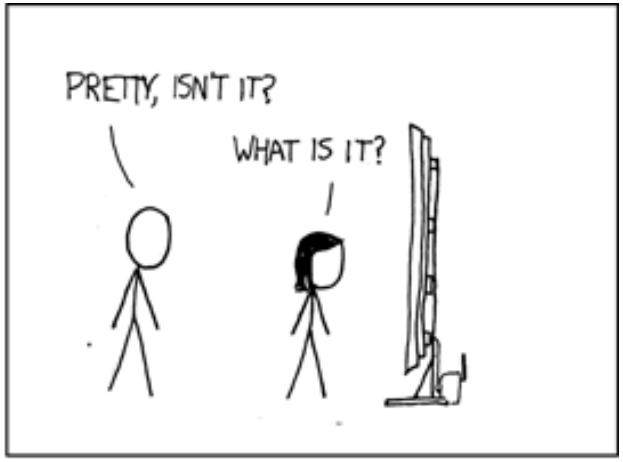
Idea: Deploy a sacrificial system that has no operational purpose

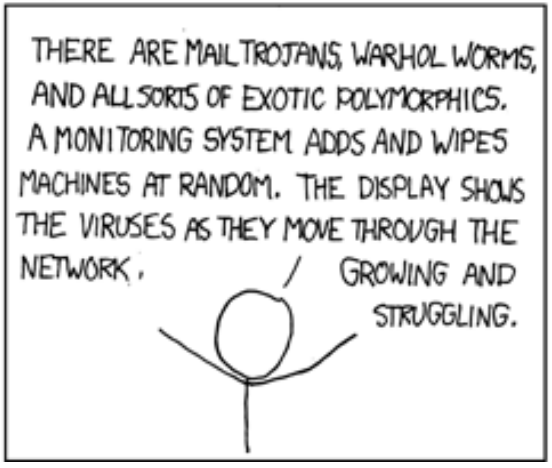
- Designed to lure attackers
- Any access is by definition not authorized, and is either an intruder or a mistake
- Provides opportunity to:
  - Identify intruders
  - Study what they’re up to
  - Divert them from legitimate targets

Honeypots

Honeypots for automated attacks easier than building a convincing environment for dedicated attackers.







XKCD