

# CSE 127: Introduction to Security

## Lecture 12: Network Defenses

**George Obaido**

UCSD

Spring 2022

# 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)
- Typical elements:
  - Firewalls
  - Network Address Translation
  - Application Proxies (e.g., Web Application Firewalls)
  - Network Intrusion Detection Systems (NIDS)



# Firewall

# 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 kind of information do you want to filter?
  - What are the examples of firewalls?

# Kinds of Firewalls

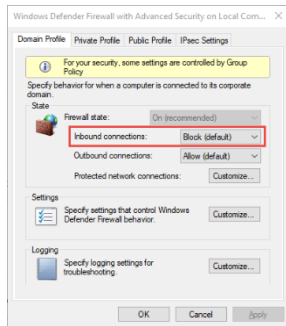
- Personal firewalls
  - Run on end-hosts
  - Has application/user-specific information
- Network firewalls
  - Intercept communications from many hosts

# 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



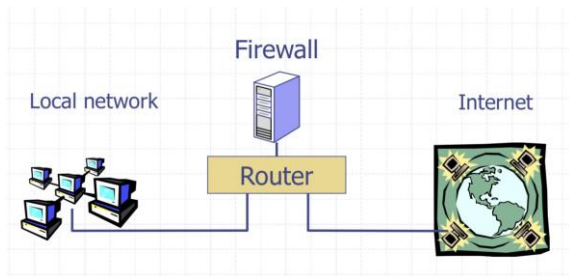
# Kinds of Firewalls



Network firewalls

Personal (host-based) firewalls

# 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: Attempts by external users to connect to services on internal machines
  - Outbound: Internal users to external services

# 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**

- Permit all services, shut off for specific problems

## **Default deny**

- Permit only a few well-known services

# 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.

```
# sudo su
# 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 outgoing packets with forged internal addresses

## Example packet filtering rules

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

## Example packet filtering rules

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

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”.

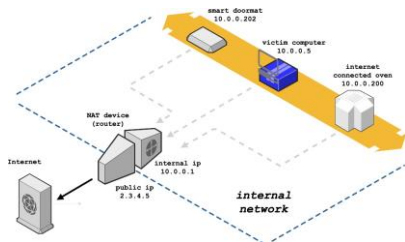


# Network Address Translation

20

# 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.



## 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

<https://samy.pl/slipstream/>

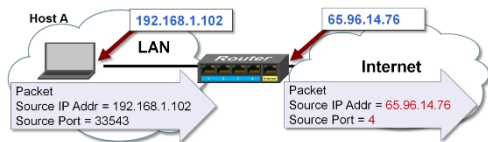
# Typical NAT Behavior

- NAT maintains a table of the form:  
<client IP> <client port> <NAT ID>



# 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



NAT Translation Table				
	Local IP Address	Source Port #	Internet IP Address	Source Port #
process X, Host A	192.168.1.101	54,847	= 65.96.14.76	1
Host B	192.168.1.103	24,123	= 65.96.14.76	2
process Y, Host A	192.168.1.101	42,156	= 65.96.14.76	3
Host C	192.168.1.102	33,543	= 65.96.14.76	4

# 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

- Pros
  - 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
  - Breaks some protocols
    - e.g., in FTP IP address appear in the content of the packet
    - e.g., some streaming protocols have client invoke server and then server opens a new connection to the client
  - Vulnerable to NAT slipstream attack (<https://samy.pl/slipstream/>)



# Application Proxies

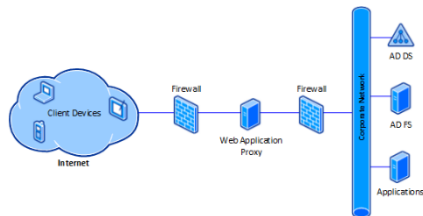
26

# 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.




# Application Proxies

## **Pros:**

- For high-level security, application proxy is the appliance of choice.
- Application proxies are high on performance.

## **Cons**

- Some application proxies can be expensive to maintain.
- Some application proxies have shown to be easily hacked despite using SSL certificate.

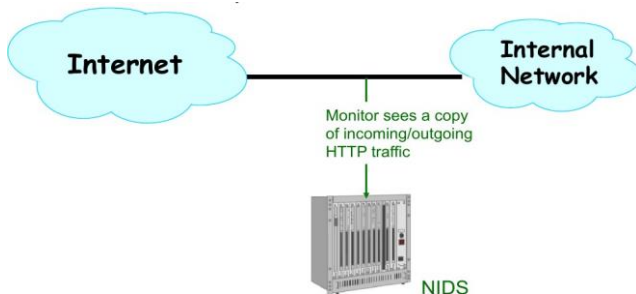


# Network Intrusion Detection System

29

# 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)

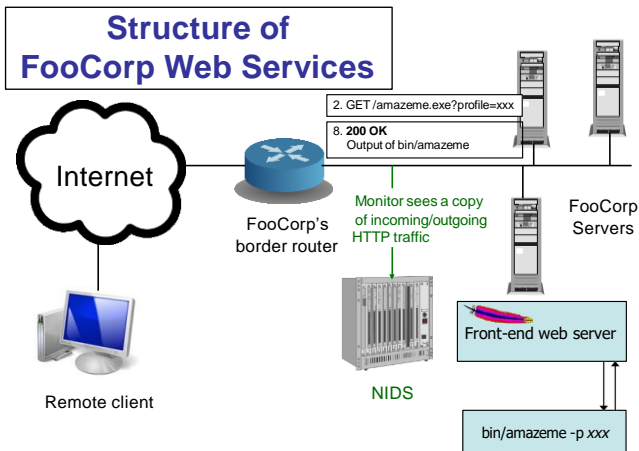
- NIDS has a table of all active connections, and maintains state for each
  - E.g., has it seen partial match of `/etc/passwd`
- What do you do when you see a new packet not associated with any known connection?
  - Create a new connection: when NIDS starts, it doesn't know what connections might be 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  $\approx$  1M packets/second  $\approx$  ns/packet

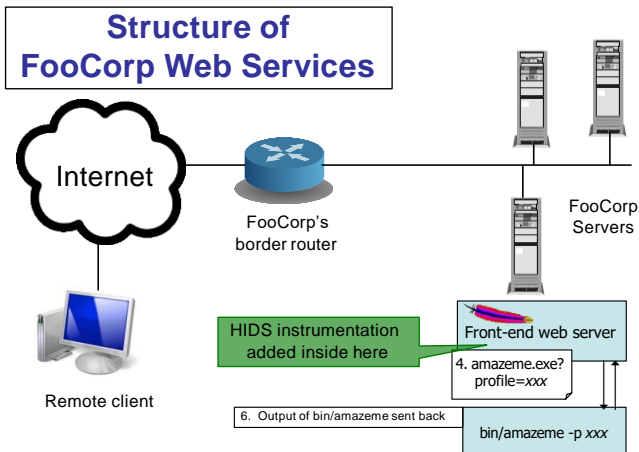
# Network-based Detection Pros and Cons

- Benefits
- Don't need to *modify* or *trust* end systems
  - Cover many systems with single monitor
  - Centralized management
- Issues
- **Expensive**: 10Gbps link  $\approx$  1M packets/second  $\approx$  ns/packet
  - Vulnerable to evasion attacks
    - Some evasions reflect **incomplete analysis**
      - E.g., hex escape or `..////.////.////`
      - 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
- 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

- Detect inconsistencies on a single host
- 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 systems?



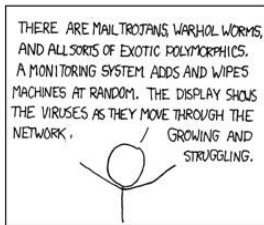
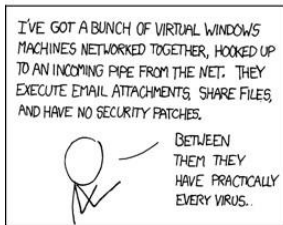
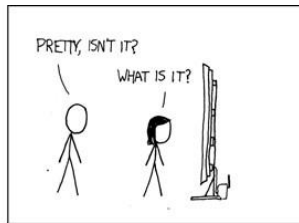
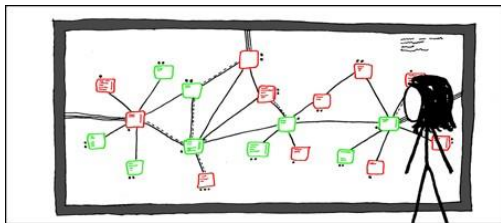
# Honeypots

**Idea:** Deploy a sacrificial system that has no operational purpose (NIDS)

- 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.





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

<https://linux.die.net/man/8/arpwatch>

## 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.)

# Example: arpwatch

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



**Cindy Moore**

to Deian, Rlad ▾

11:33 AM (52 minutes ago)



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

----- Forwarded message -----

From: **Arpwatch** [sysnet.sysnet.ucsd.edu](mailto:sysnet.sysnet.ucsd.edu) <[arpwatch@sysnet.sysnet.ucsd.edu](mailto:arpwatch@sysnet.sysnet.ucsd.edu)>

Date: Sat, Nov 9, 2019 at 12:23 PM

Subject: flip flop ([elk.sysnet.ucsd.edu](mailto:elk.sysnet.ucsd.edu)) eno1

To: <[root@sysnet.sysnet.ucsd.edu](mailto:root@sysnet.sysnet.ucsd.edu)>

hostname: [elk.sysnet.ucsd.edu](http://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