

Firewalls/Detection

CS 161: Computer Security

Prof. Raluca Ada Popa

March 8, 2018

Controlling Networks ... On The Cheap

- Motivation: How do you harden a set of systems against external attack?
 - *Key Observation:*
 - *The more network services your machines run, the greater the risk*
 - Due to larger **attack surface**
- One approach: on each system, turn off unnecessary network services
 - But you have to know **all** the services that are running
 - And sometimes some trusted remote users still require access

Controlling Networks ... On The Cheap

- Motivation: How do you harden a set of systems against external attack?
 - *Key Observation:*
 - *The more network services your machines run, the greater the risk*
 - Due to larger **attack surface**
- One approach: on each system, turn off unnecessary network services
 - But you have to know **all** the services that are running
 - And sometimes some trusted remote users still require access
- Plus key question of **scaling**
 - What happens when you have to secure 100s/1000s of systems?
 - Which may have different OSs, hardware & users ...
 - Which may in fact not all even be identified ...

Taming Management Complexity

- Possibly more scalable defense: Reduce risk by blocking *in the network* **outsiders** from having unwanted access your network services
 - Interpose a **firewall** into the traffic to/from the outside must traverse
 - **Chokepoint** can cover thousands of hosts
 - Where in everyday experience do we see such chokepoints?



Selecting a Security Policy

- Firewall enforces an (access control) **policy**:
 - *Who is allowed to talk to whom, accessing what service?*
- Distinguish between **inbound** & **outbound** connections
 - **Inbound**: attempts by external users to connect to services on internal machines
 - **Outbound**: internal users to external services
 - Why? Because fits with a common **threat model**. There are thousands of internal users (and we've vetted them). There are billions of outsiders.
- Conceptually simple **access control policy**:
 - Permit inside users to connect to any service
 - External users restricted:
 - **Permit** connections to services meant to be externally visible
 - **Deny** connections to services not meant for external access

How To Treat Traffic Not Mentioned in Policy?


- **Default Allow**: start off permitting external access to services
 - Shut them off as problems recognized

How To Treat Traffic Not Mentioned in Policy?

- **Default Allow:** start off permitting external access to services
 - Shut them off as problems recognized
- **Default Deny:** start off permitting just a few known, well-secured services
 - Add more when users as they complain (and mgt. approves)

Pros and cons?

How To Treat Traffic Not Mentioned in Policy?

- **Default Allow:** start off permitting external access to services
 - Shut them off as problems recognized
- **Default Deny:**  rt off permitting just a few known, well-secured services
 - Add more when users complain (and mgt. approves)
- Pros & Cons?
 - Flexibility vs. conservative design
 - Flaws in Default Deny get **noticed** more quickly / less painfully

In general, use Default Deny

Types of firewalls

1. Packet filters (stateless)
2. Stateful packet filter
3. Application-level firewall

Packet filter

- A packet filter is a firewall that inspects each packet for certain filtering rules to determine whether to pass or block it
- Filtering rules are based on the network and transport layer: source IP address, destination IP address, Layer 4 (that is, TCP/UDP) source port, and Layer 4 destination port
- Pro: very fast, can be implemented in routers
- Con:
 - They have no logging facility that can be used to detect when a break-in has occurred
 - Ports can be spoofed

Stateful Packet Filter

- Stateful packet filter **keeps track of all connections** (inbound/outbound)
 - Each rule specifies which connections are allowed/denied (*access control policy*)
 - A packet is forwarded if it is part of an allowed connection



Example Rule

```
allow tcp connection 4.5.5.4:* -> 3.1.1.2:80
```

- Firewall should **permit** TCP connection that's:
 - Initiated by host with Internet address 4.5.5.4 **and**
 - Connecting to port 80 of host with IP address 3.1.1.2
- Firewall should permit any packet associated with this connection
- Thus, firewall keeps a table of (allowed) active connections. When firewall sees a packet, it checks whether it is part of one of those active connections. If yes, forward it; if no, drop it.

Example Rule

```
allow tcp connection *:*/int -> 3.1.1.2:80/ext
```

- Firewall should **permit** TCP connection that's:
 - Initiated by host with any internal host **and**
 - Connecting to port 80 of host with IP address 3.1.1.2 on external Internet
- Firewall should permit any packet associated with this connection

- The **/int** indicates the network interface.

Example Ruleset

```
allow tcp connection */*/int -> */*/ext  
allow tcp connection */*/ext -> 1.2.2.3:80/int
```

- Firewall should permit outbound TCP connections (i.e., those that are initiated by internal hosts)
- Firewall should permit inbound TCP connection to our public webserver at IP address 1.2.2.3

Stateful Filtering

Discussion question:

Suppose you want to allow inbound connection to a FTP server (FTP= file transfer protocol), but block any attempts to login as “root”. How would you build a stateful packet filter to do that? In particular, what state would it keep, for each connection?

- assume traffic is unencrypted

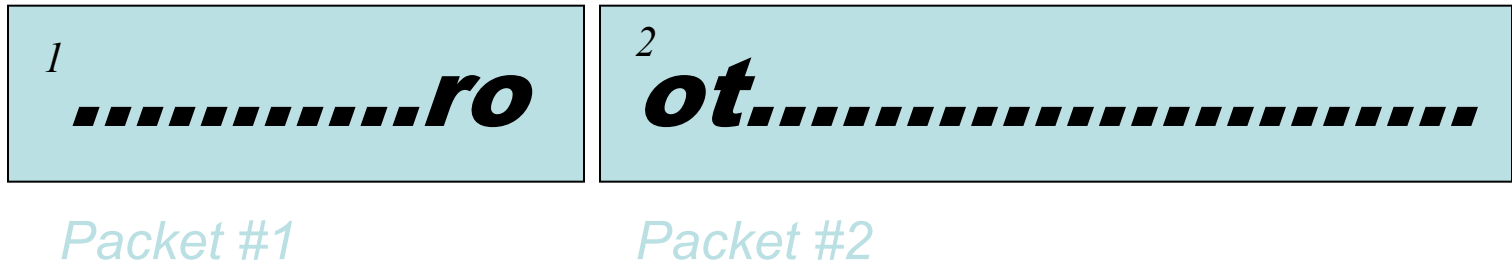
Discuss with a partner.

State Kept

- No state – just drop any packet with root in them
- State ideas:
 - Is it a FTP connection?
 - Where in FTP state (e.g. command, what command)
 - Src ip addr, dst ip addr, src port, dst port
 - Inbound/outbound connection
 - Keep piece of login command until it's completed
 - only first 5 bytes of username

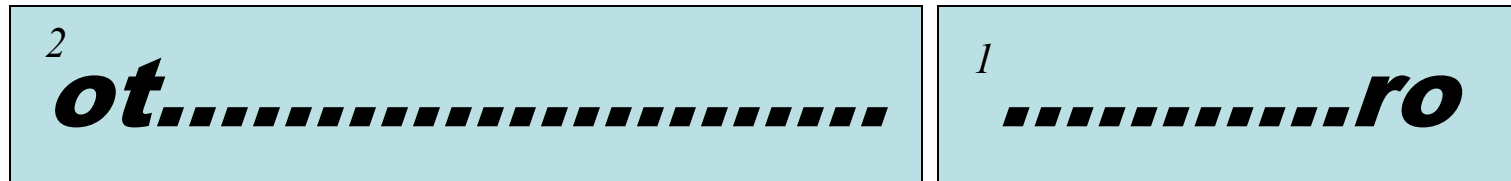
Beware!

- Sender might be malicious and trying to sneak through firewall
- “root” might span packet boundaries



Beware!

- Packets might be re-ordered

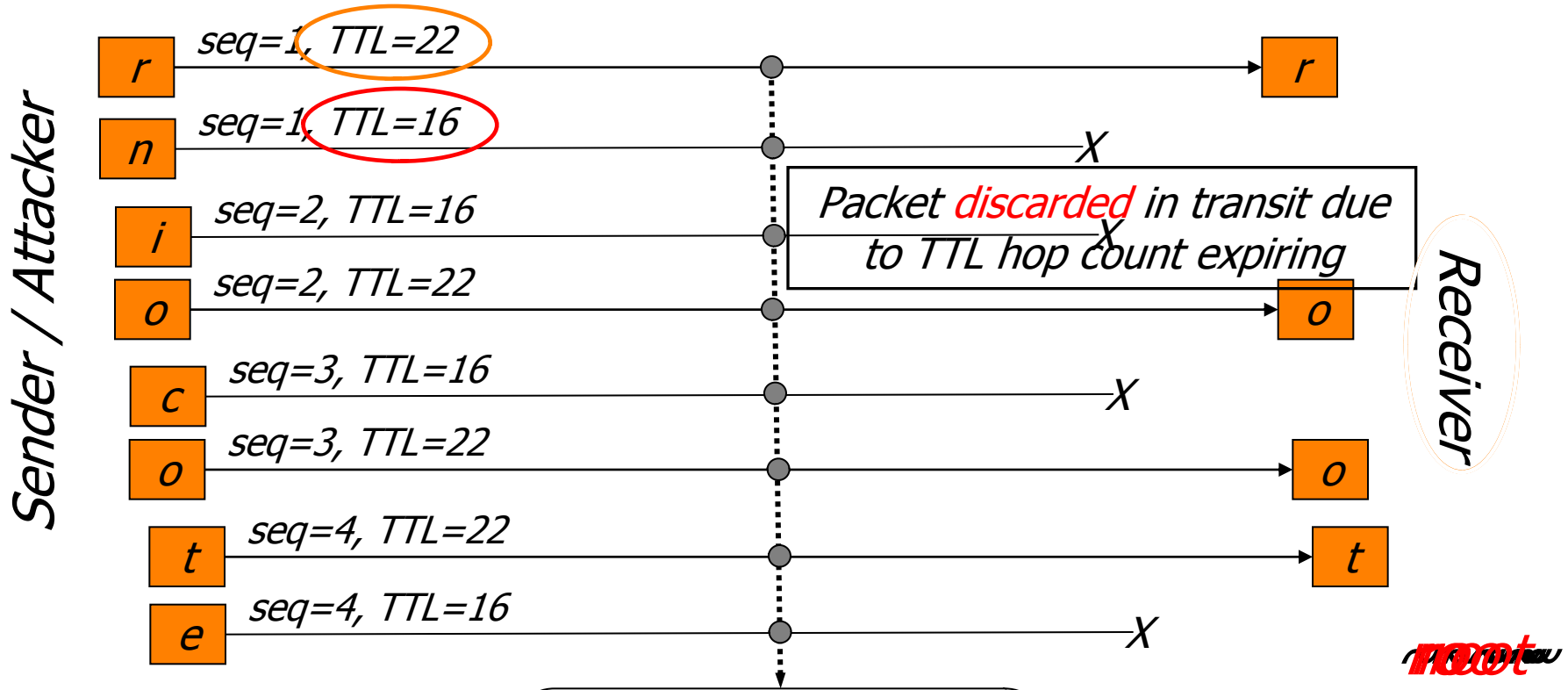


How to address this?

- TCP reconstruction: the stateful packet filter will reconstruct the sequence of packets by putting them in order based on their sequence numbers and examining the payload as it spans packet boundaries

Can an attacker sending the string “root” to the destination still evade being caught?

Beware!



TTL field in IP header specifies maximum forwarding hop count

Firewall

rice? roce? rict? roct? riot?
 root? roie? robe? rice?
 rice? rict? roie? robe?
 root? roie? nooe?

Assume the Receiver is 20 hops away

Assume firewall is 15 hops away

Application-level firewall

- Firewall acts as a proxy server that provides access control at the application layer.
- TCP connection from client to firewall, which then makes a second TCP connection from firewall to server.
- Pro: can examine traffic in detail (including payload=content of packet), so a more secure type of firewall, and it can log.
- Con: processing intensive and can become a bottleneck under heavy traffic conditions.

Why Have Firewalls Been Successful?

- *Central control* – *easy administration and update*
 - Single point of control: update one config to change security policies
 - Potentially allows rapid response
- *Easy to deploy* – *transparent to end users*
 - Easy incremental/total deployment to protect 1000's
- *Addresses an important problem*
 - Security vulnerabilities in network services are rampant
 - Easier to use firewall than to directly secure code ...

Attacks to Firewalls Don't Stop?

Discussion question:

Suppose you wanted to attack a company protected by a firewall. What attacks might you try?

Discuss with a partner.

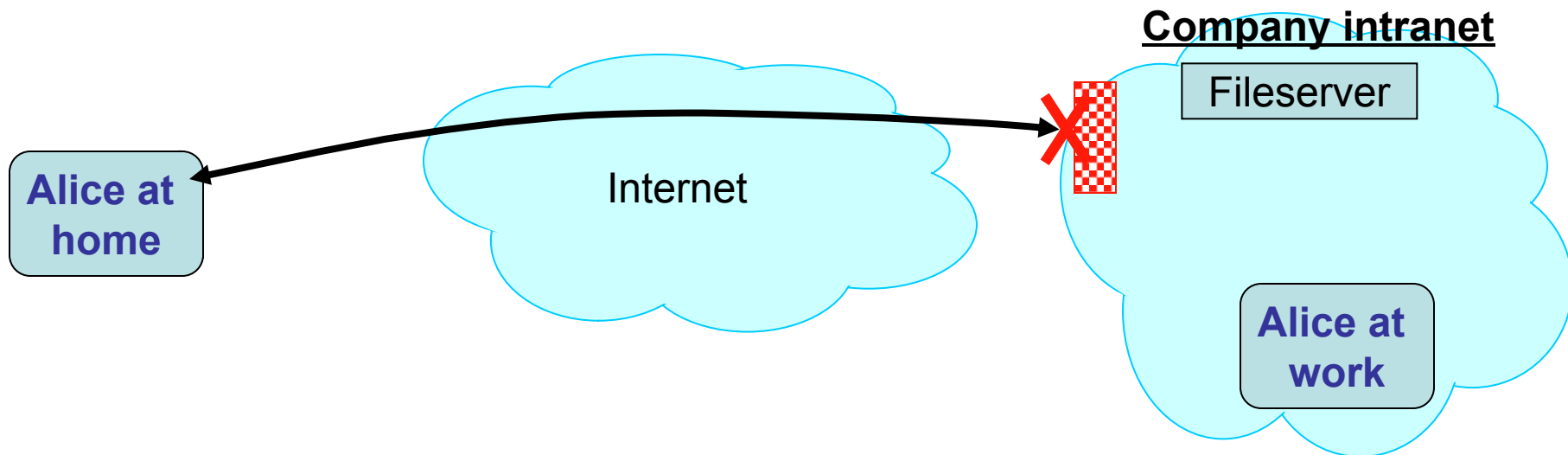
Firewall Disadvantages

- *Functionality loss – less connectivity, less risk*
 - May reduce network's usefulness
 - Some applications don't work with firewalls
 - Two peer-to-peer users behind different firewalls
- *The malicious insider problem*
 - Assume insiders are trusted
 - Malicious insider (or anyone gaining control of internal machine) can wreak havoc
- Firewalls establish a *security perimeter*
 - Like *Eskimo Pies*: “hard crunchy exterior, soft creamy center”
 - Threat from travelers with laptops, ...

Takeaways on Firewalls

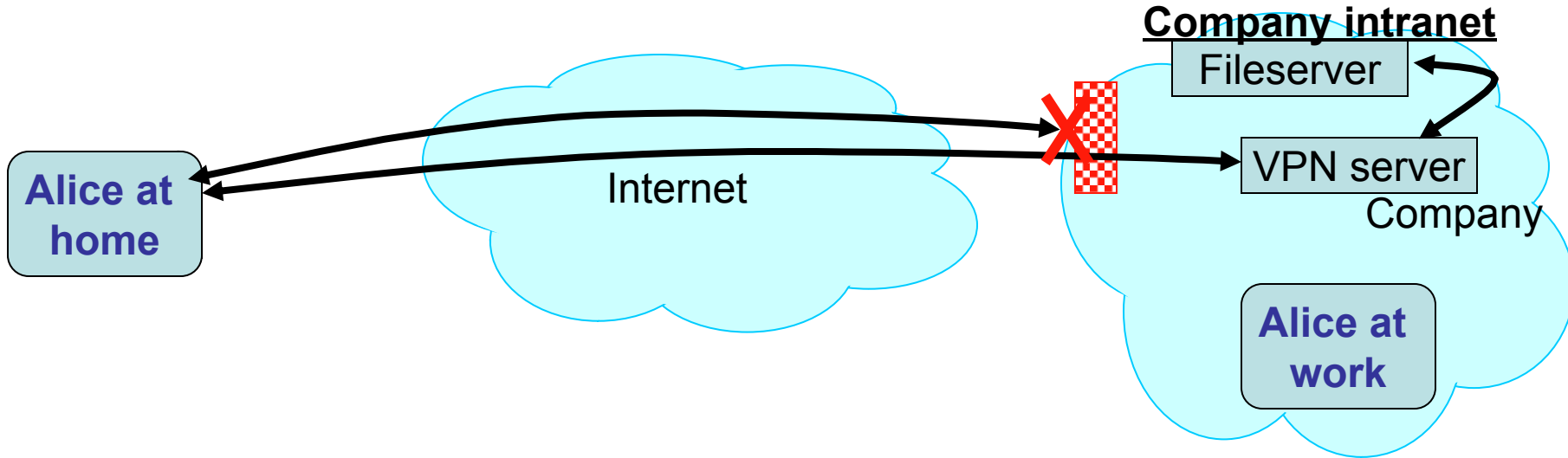
- Firewalls: Reference monitors and access control all over again, but at the network level
- Attack surface reduction
- Centralized control

Secure External Access to Inside Machines



- Often need to provide secure remote access to a network protected by a firewall
 - Remote access, telecommuting, branch offices, ...
- Alice wants to access work network from home, e.g., contact file server, over the public Internet. Firewall does not allow outside access.
- How can we give Alice access since she works for the company?

Secure External Access to Inside Machines



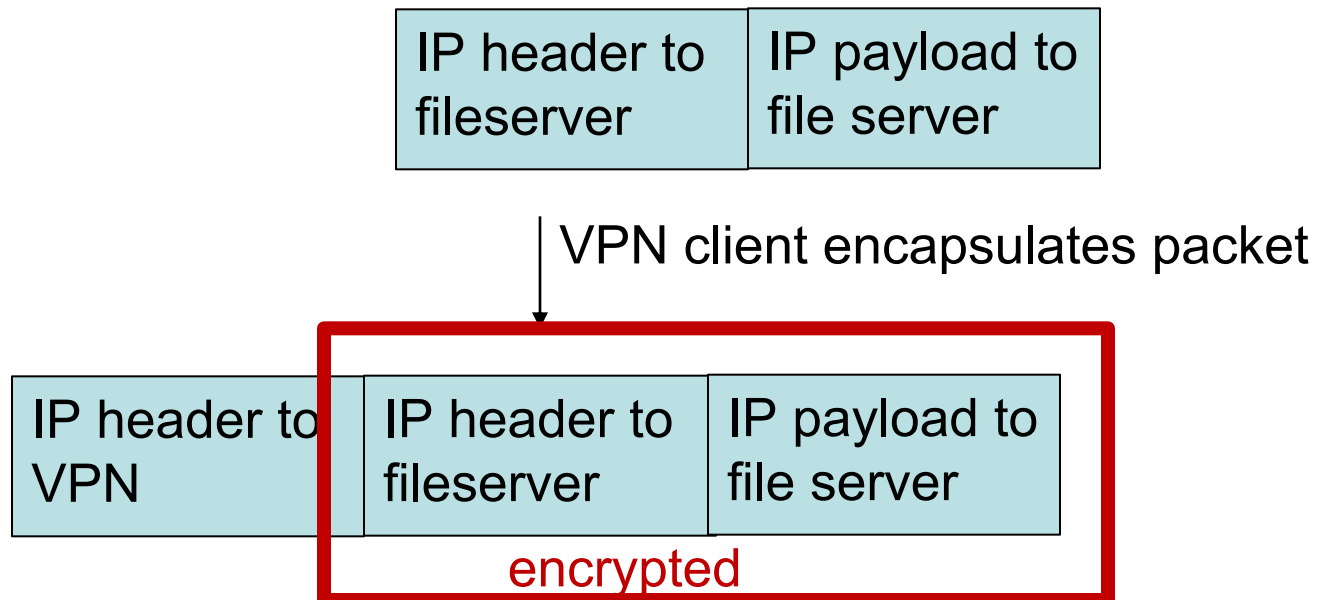
- Create secure channel (*Virtual Private Network*, or **VPN**) to **tunnel** traffic from outside host/network to inside network

VPN

- An intermediary to which an outside user authenticates and forwards packets to the destination
- Firewall can be configured to allow traffic going to the VPN server
- If Alice can authenticate to VPN, the VPN will make internal requests for her
- VPN can forward packets both inside and outside

Tunneling

- The VPN client running on Alice's machine will create a tunnel with the VPN
- Tunneling is a mechanism for encapsulating one protocol in another protocol.



VPN

- Provides **Authentication, Confidentiality, Integrity** via encryption and digital signatures/MAC
- Try it yourself at <http://www.net.berkeley.edu/vpn/>

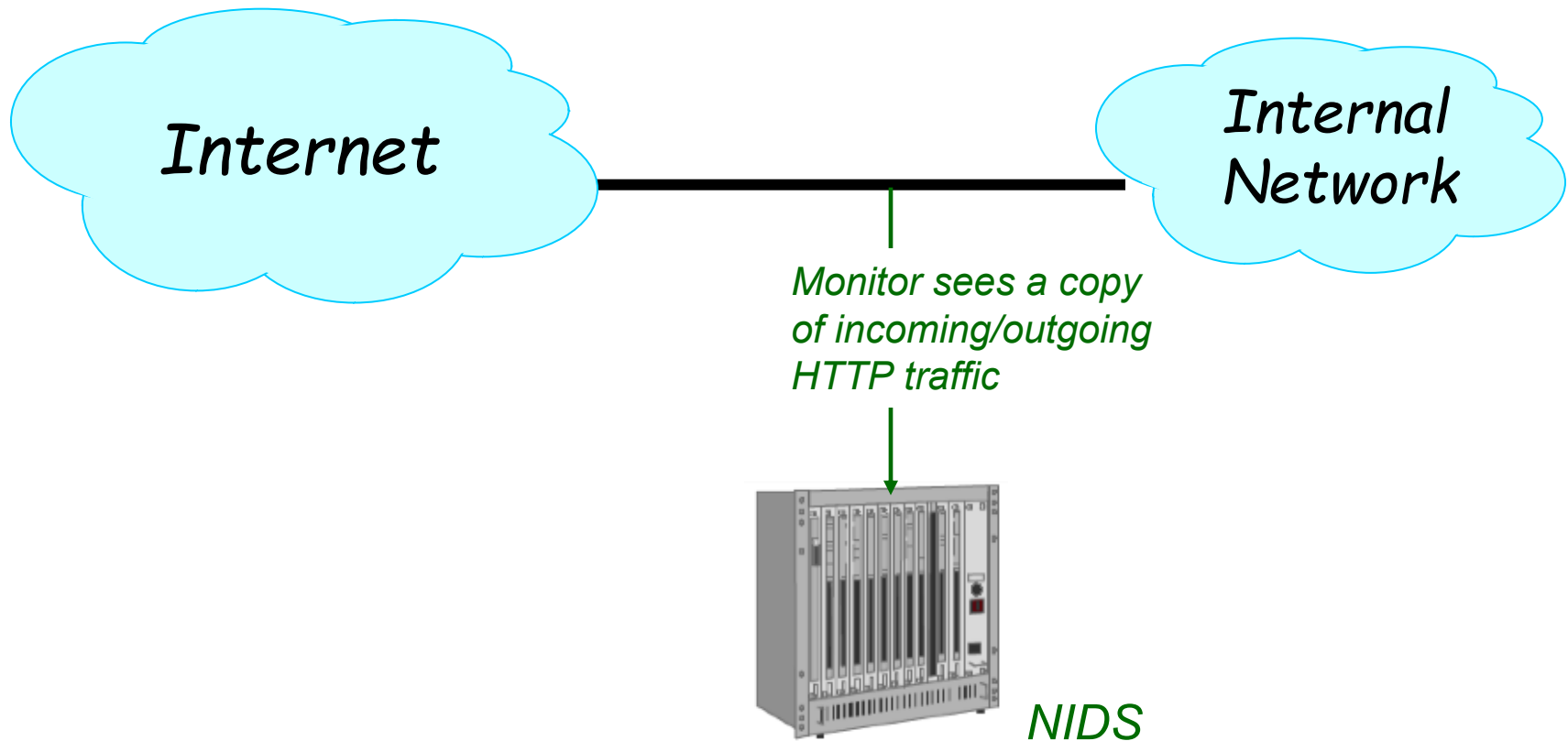
A familiar example

- You want to download paper from IEEE society's website. Can access it from Berkeley's campus because campus pays dues to IEEE
- Cannot access from outside
- Connect to Berkeley VPN from outside and you can access it

Detecting Attacks

Network Intrusion Detection (NIDS)

- Passively monitor network traffic for signs of attack
 - Look for `/etc/passwd`

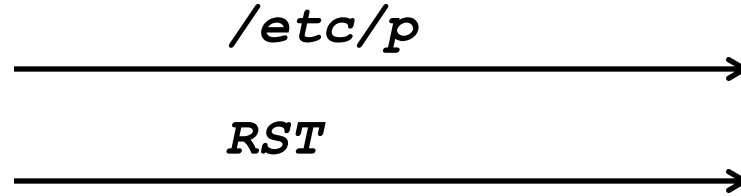


Network Intrusion Detection (NIDS)

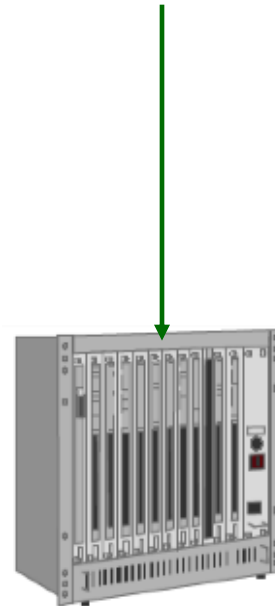
- NIDS has a table of all active connections, and maintains state for each
 - e.g., has it seen a partial match of /etc/passwd?
- What do you do when you see a new packet not associated with any known connection?
 - Create a new connection: when NIDS starts it doesn't know what connections might be existing

Evasion

- What should NIDS do if it sees a RST packet?



- (a) Assume RST will be received
- (b) Assume RST won't be received
- (c) Other (please specify)



NIDS

Evasion

- What should NIDS do if it sees this?

/%65%74%63/%70%61%73%73%77%64

- (a) Alert – it's an attack
- (b) No alert – it's all good
- (c) Other (please specify)



NIDS

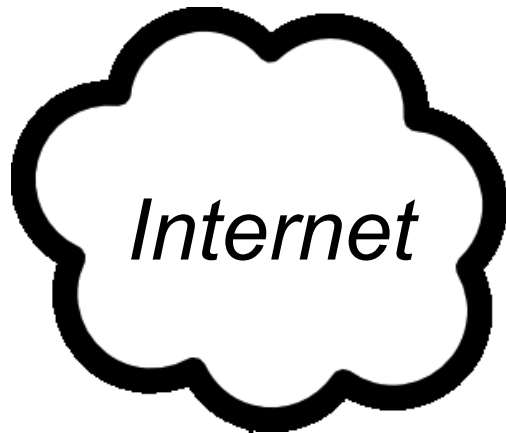
Evasion

- Evasion attacks arise when you have “double parsing”
- *Inconsistency* – interpreted differently
- *Ambiguity* – information needed to interpret is missing

Evasion Attacks (High-Level View)

- Some evasions reflect **incomplete analysis**
 - In our example, hex escapes or “../////..//..//” alias
 - In principle, can deal with these with implementation care (make sure we **fully understand the spec**)
- Some are due to **imperfect observability**
 - For instance, if what NIDS sees doesn't exactly match what arrives at the destination

Structure of FooCorp Web Services



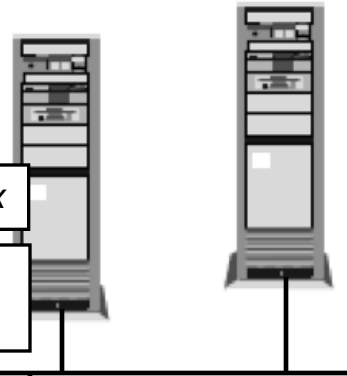
Remote client



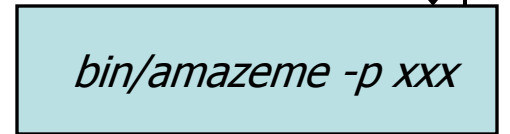
FooCorp's border router

2. `GET /amazeme.exe?profile=xxx`

8. **200 OK**
Output of `bin/amazeme`



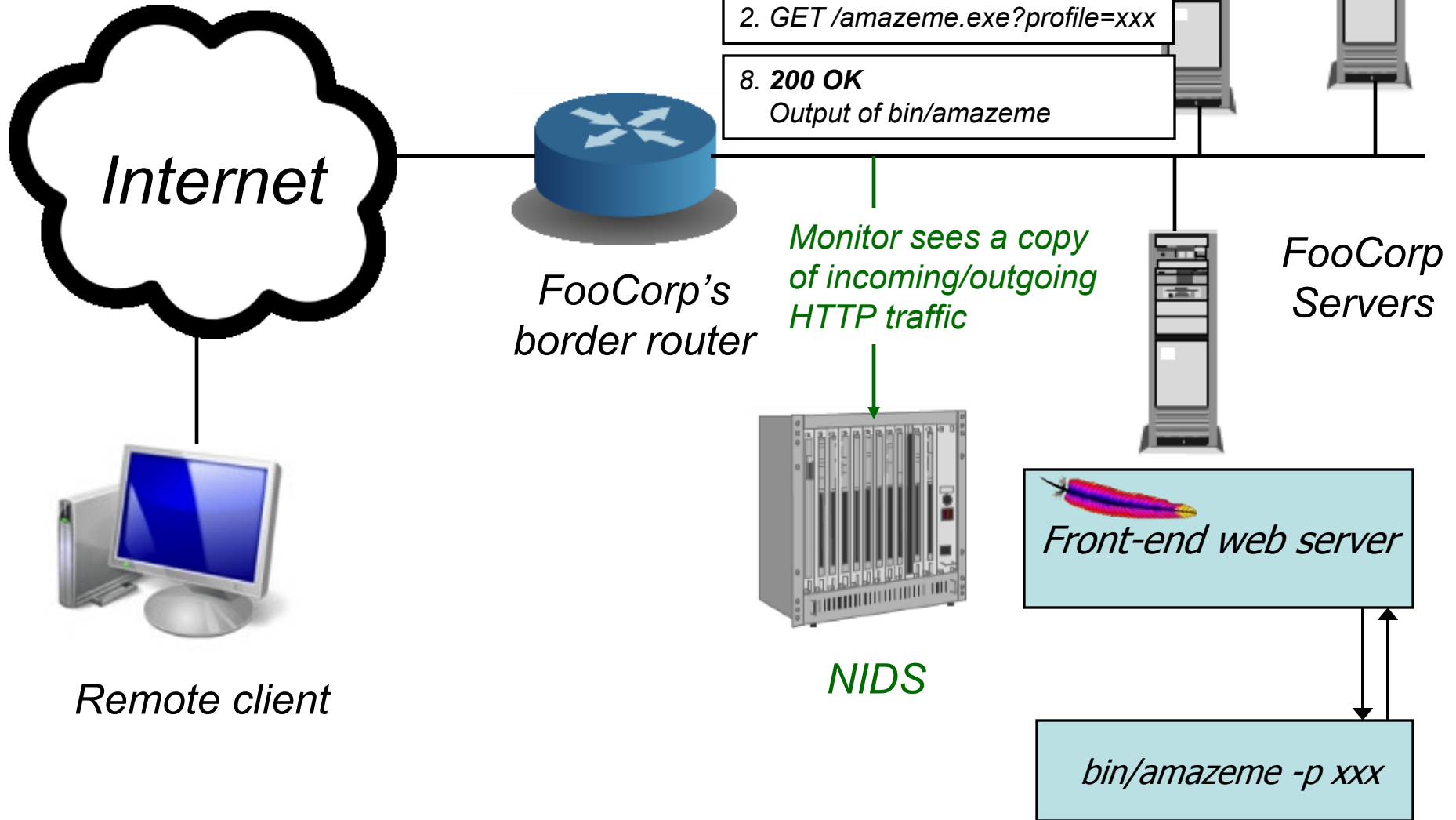
FooCorp Servers



Network Intrusion Detection

- Approach #1: look at the network traffic
 - (a “NIDS”: rhymes with “kids”)
 - Scan HTTP requests
 - Look for “/etc/passwd” and/or “../..”

Structure of FooCorp Web Services



Network Intrusion Detection

- Approach #1: look at the network traffic
 - (a “NIDS”: rhymes with “kids”)
 - Scan HTTP requests
 - Look for “/etc/passwd” and/or “../..”
- Pros:
 - No need to **touch or trust** end systems
 - Can “bolt on” security
 - **Cheap**: cover many systems w/ single monitor
 - **Cheap**: centralized management

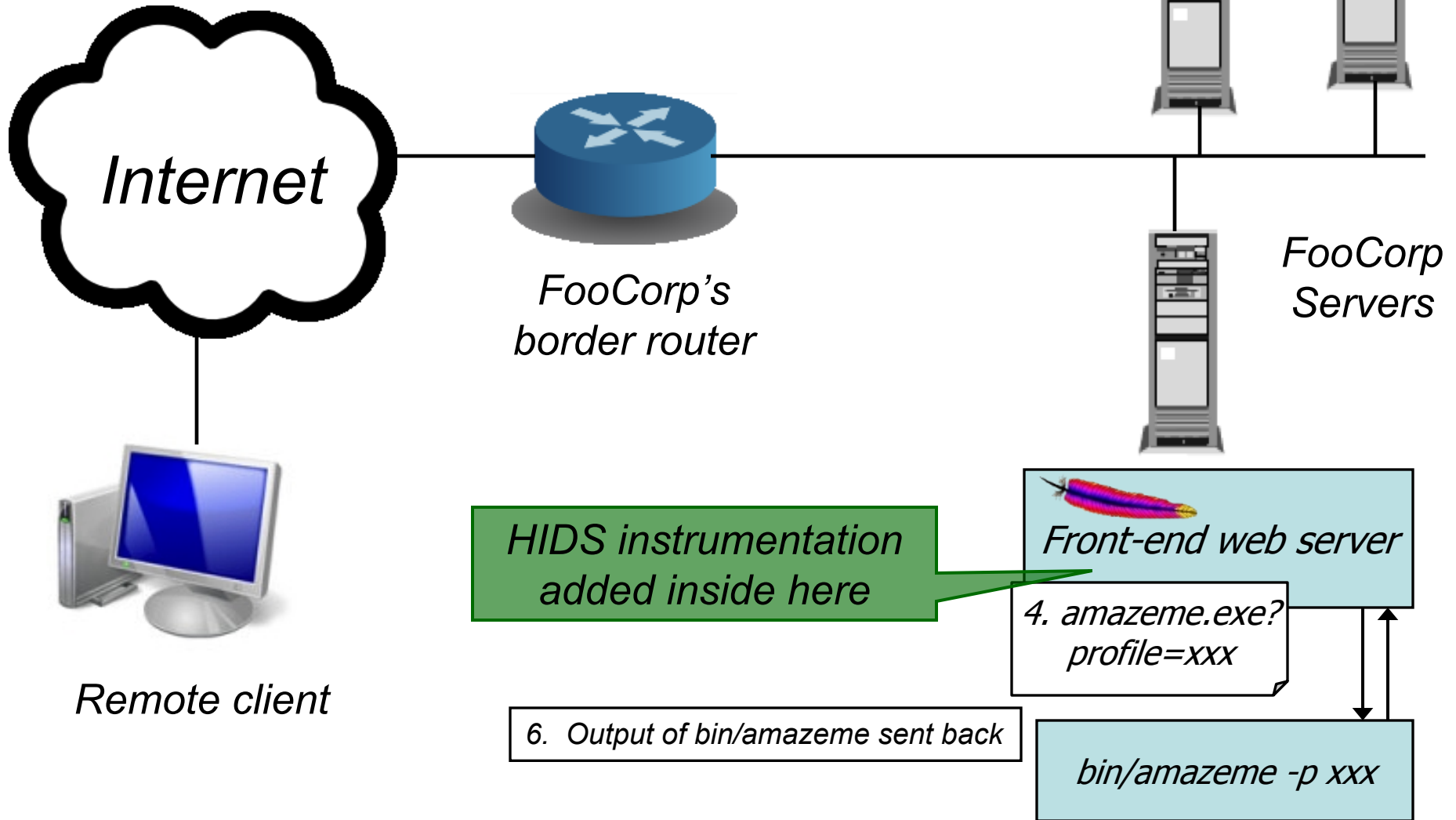
Network-Based Detection

- Issues:
 - Scan for “/etc/passwd”?
 - What about *other* sensitive files?
 - Scan for “../..”?
 - Sometimes seen in legit. requests (= *false positive*)
 - What about “%2e%2e%2f%2e%2e%2f”? (= *evasion*)
 - Okay, need to do full HTTP parsing
 - What about “..///.///..////”?
 - Okay, need to understand Unix filename semantics too!
 - Overall problem: don’t understand applications
 - What if it’s HTTPS and not HTTP?
 - Need access to decrypted text / session key – yuck!

Host-based Intrusion Detection

- Approach #2: instrument the web server
 - Host-based IDS (sometimes called “HIDS”)
 - Scan arguments sent to back-end programs
 - Look for “/etc/passwd” and/or “../..”

Structure of FooCorp Web Services



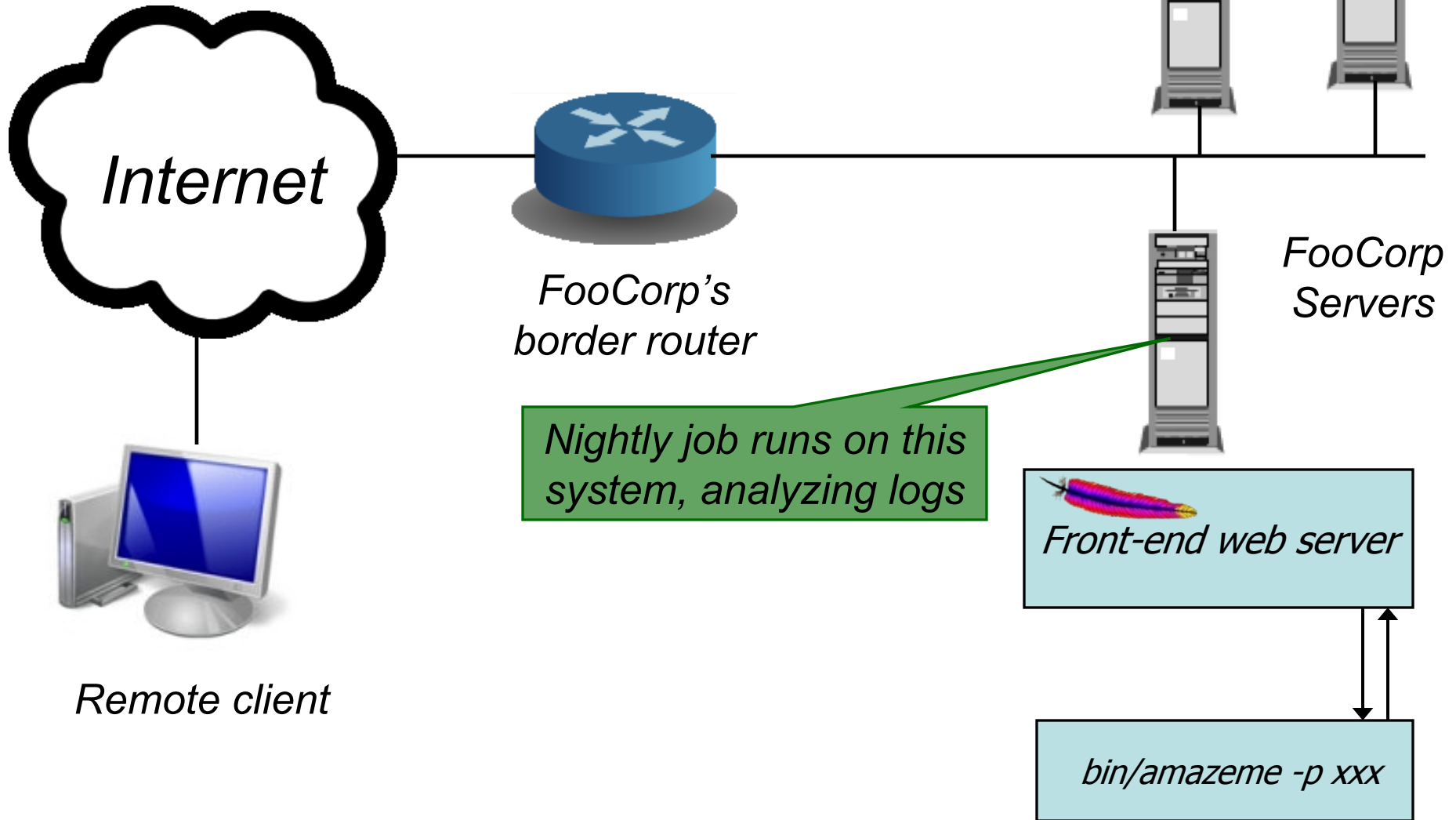
Host-based Intrusion Detection

- Approach #2: instrument the web server
 - Host-based IDS (sometimes called “HIDS”)
 - Scan arguments sent to back-end programs
 - Look for “/etc/passwd” and/or “../..”
- Pros:
 - No problems with HTTP complexities like %-escapes
 - Works for encrypted HTTPS!
- Issues:
 - Have to add code to each (possibly different) web server
 - And that effort only helps with detecting web server attacks
 - Still have to consider Unix filename semantics (“../////..”)
 - Still have to consider other sensitive files

Log Analysis

- Approach #3: each night, script runs to analyze **log files** generated by web servers
 - Again scan arguments sent to back-end programs

Structure of FooCorp Web Services



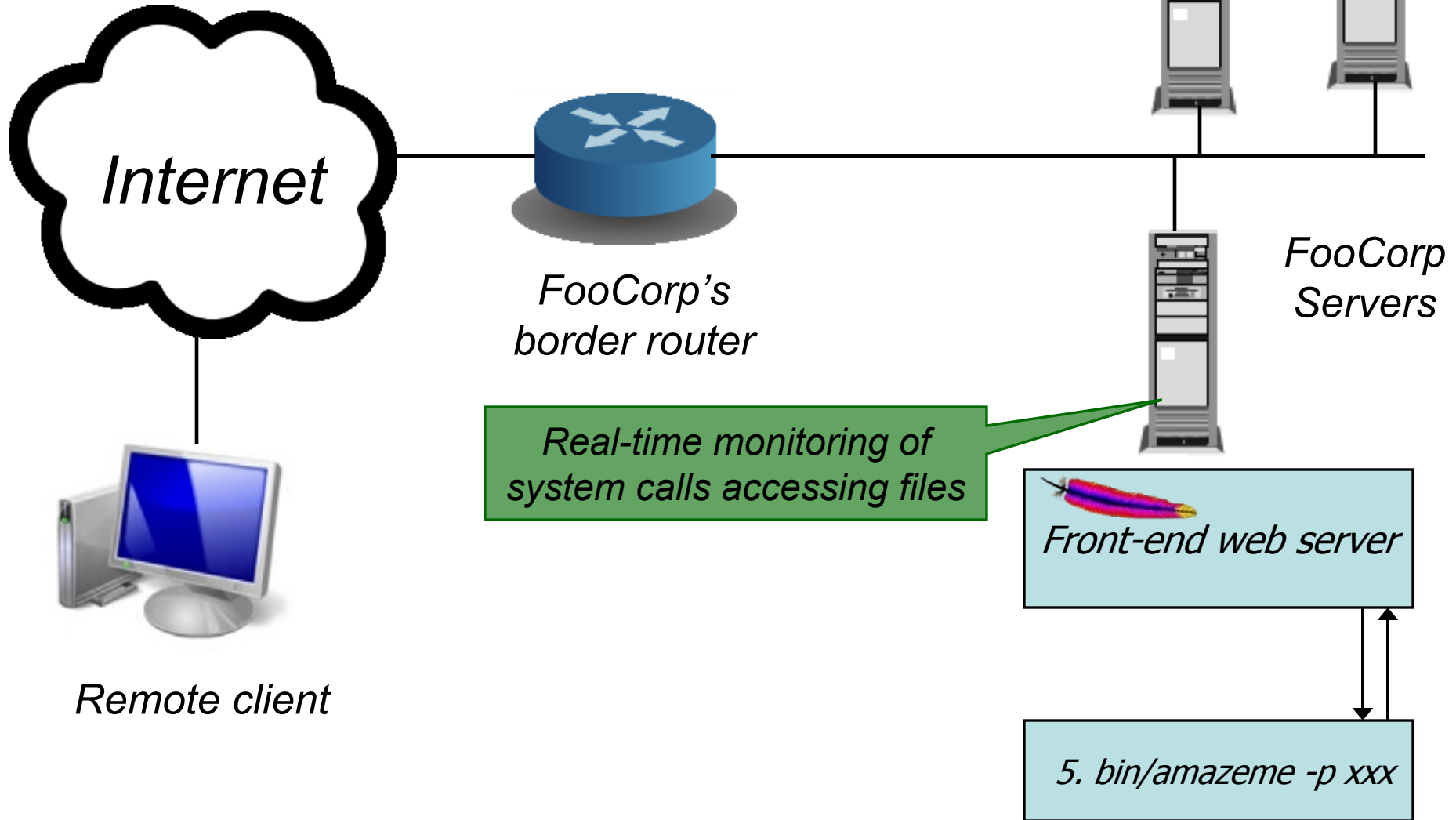
Log Analysis

- Approach #3: each night, script runs to analyze log files generated by web servers
 - Again scan arguments sent to back-end programs
- Pros:
 - **Cheap**: web servers generally already have such logging facilities built into them
 - No problems like %-escapes, encrypted HTTPS
- Issues:
 - Again must consider filename tricks, other sensitive files
 - Can't block attacks & prevent from happening
 - Detection **delayed**, so attack damage may **compound**
 - If the attack is a compromise, then malware might be able to **alter the logs** before they're analyzed
 - (Not a problem for directory traversal information leak example)

System Call Monitoring (HIDS)

- Approach #4: monitor **system call activity** of backend processes
 - Look for access to /etc/passwd

Structure of FooCorp Web Services



System Call Monitoring (HIDS)

- Approach #4: monitor system call activity of backend processes
 - Look for access to /etc/passwd
- Pros:
 - No issues with any HTTP complexities
 - May avoid issues with filename tricks
 - Attack only leads to an “**alert**” if attack succeeded
 - Sensitive file was indeed accessed
- Issues:
 - Maybe other processes make **legit** accesses to the sensitive files (*false positives*)
 - Maybe we’d like to detect attempts even if they fail?
 - “situational awareness”

Detection Accuracy

- Two types of detector errors:
 - **False positive** (FP): alerting about a problem when in fact there was no problem
 - **False negative** (FN): failing to alert about a problem when in fact there was a problem
- Detector accuracy is often assessed in terms of rates at which these occur:
 - Define **I** to be the event of an instance of intrusive behavior occurring (something we want to detect)
 - Define **A** to be the event of detector generating alarm
- Define:
 - *False positive rate* = $P[A | \neg I]$
 - *False negative rate* = $P[\neg A | I]$

Perfect Detection

- Is it possible to build a detector for our example with a false negative rate of **0%**?
- Algorithm to detect bad URLs with **0% FN rate**:

```
void my_detector_that_never_misses(char *URL)
{
    printf("yep, it's an attack!\n");
}
```

 - In fact, it works for detecting **any** bad activity with no false negatives! **Woo-hoo!**
- Wow, so what about a detector for bad URLs that has **NO FALSE POSITIVES**?!
 - `printf("nope, not an attack\n");`

Detection Tradeoffs

- The art of a good detector is achieving an **effective balance** between FPs and FNs
- Suppose our detector has an FP rate of 0.1% and an FN rate of 2%. Is it good enough? Which is better, a very low FP rate or a very low FN rate?
 - Depends on the **cost** of each type of error ...
 - E.g., FP might lead to paging a duty officer and consuming hour of their time; FN might lead to \$10K cleaning up compromised system that was missed
 - ... but also **critically** depends on the rate at which actual attacks occur in your environment

Base Rate Fallacy

- Suppose our detector has a FP rate of 0.1% (!) and a FN rate of 2% (not bad!)
- Scenario #1: our server receives 1,000 URLs/day, and 5 of them are attacks
 - Expected # FPs each day = $0.1\% * 995 \approx 1$
 - Expected # FNs each day = $2\% * 5 = 0.1$ (< 1/week)
 - Pretty good!
- Scenario #2: our server receives 10,000,000 URLs/day, and 5 of them are attacks
 - Expected # FPs each day $\approx 10,000$:-)
- *Nothing changed about the detector*; only our **environment** changed
 - Accurate detection very challenging when **base rate** of activity we want to detect is quite low

Conclusion

- NIDS attempt to detect network attacks by monitoring traffic or application behavior
- Different types of NIDS have different pros/cons and false positive/false negatives tradeoffs