Network reconnaissance and Intrusion Detection

CS642: Computer Security

Let’s play over the network …

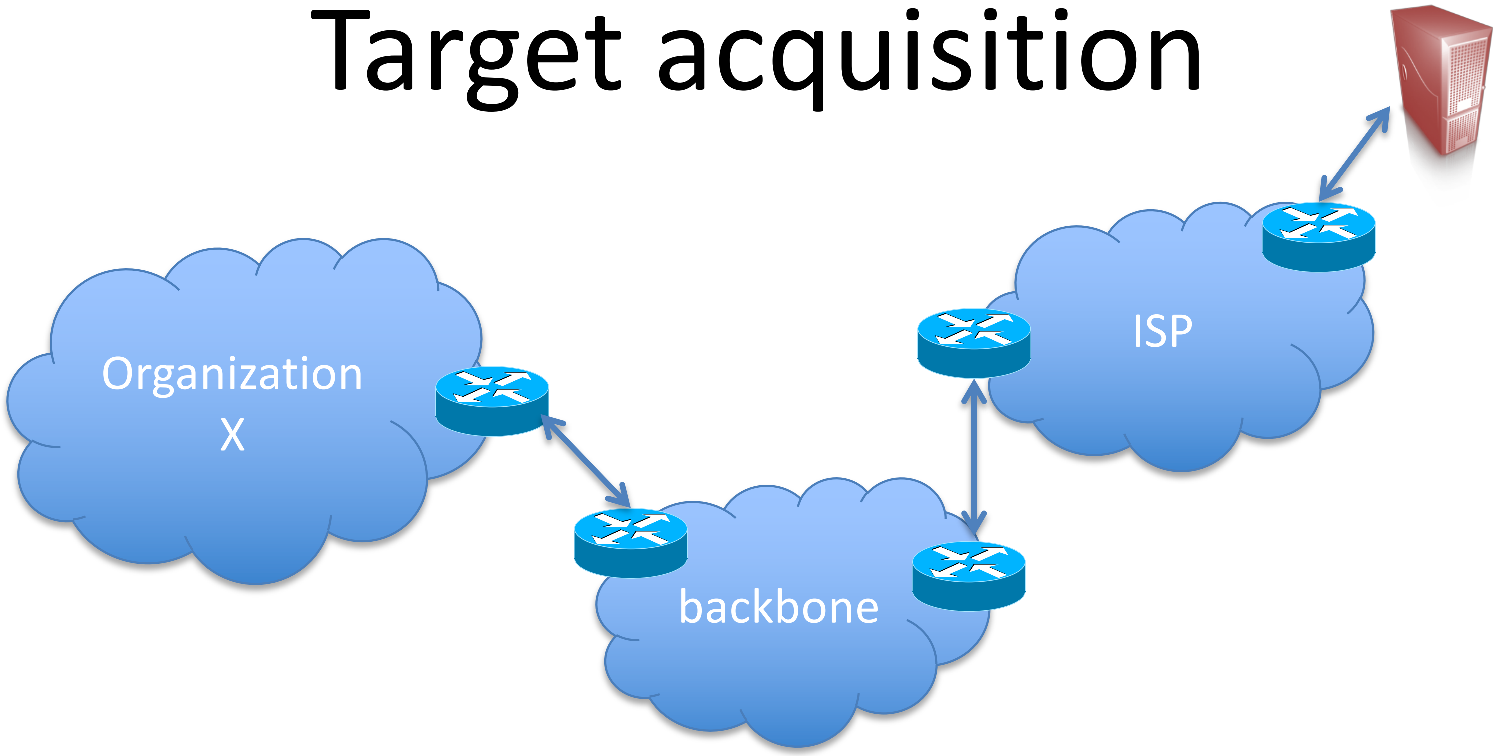
Port scanning

Host fingerprinting, NMAP

Network IDS basics

Avoiding IDS

Target acquisition



How do we find vulnerable server(s) within a target organization?

Starting point: one or more publicly routable IP addresses

- WHOIS queries are good way to find them
- Can be used to identify blocks of IP addresses owned

WHOIS fun

```
[swift:~] whois 127.217.0.0
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

inetnum:      127.0.0.0 - 127.255.255.255
organisation: IANA - Loopback
status:       RESERVED

remarks:      127.0.0.0/8 reserved for Loopback [RFC1122], section
remarks:      3.2.1.3. Reserved by protocol. For authoritative
remarks:      registration, seeiana-ipv4-special-registry.

changed:      1981-09
source:       IANA
```

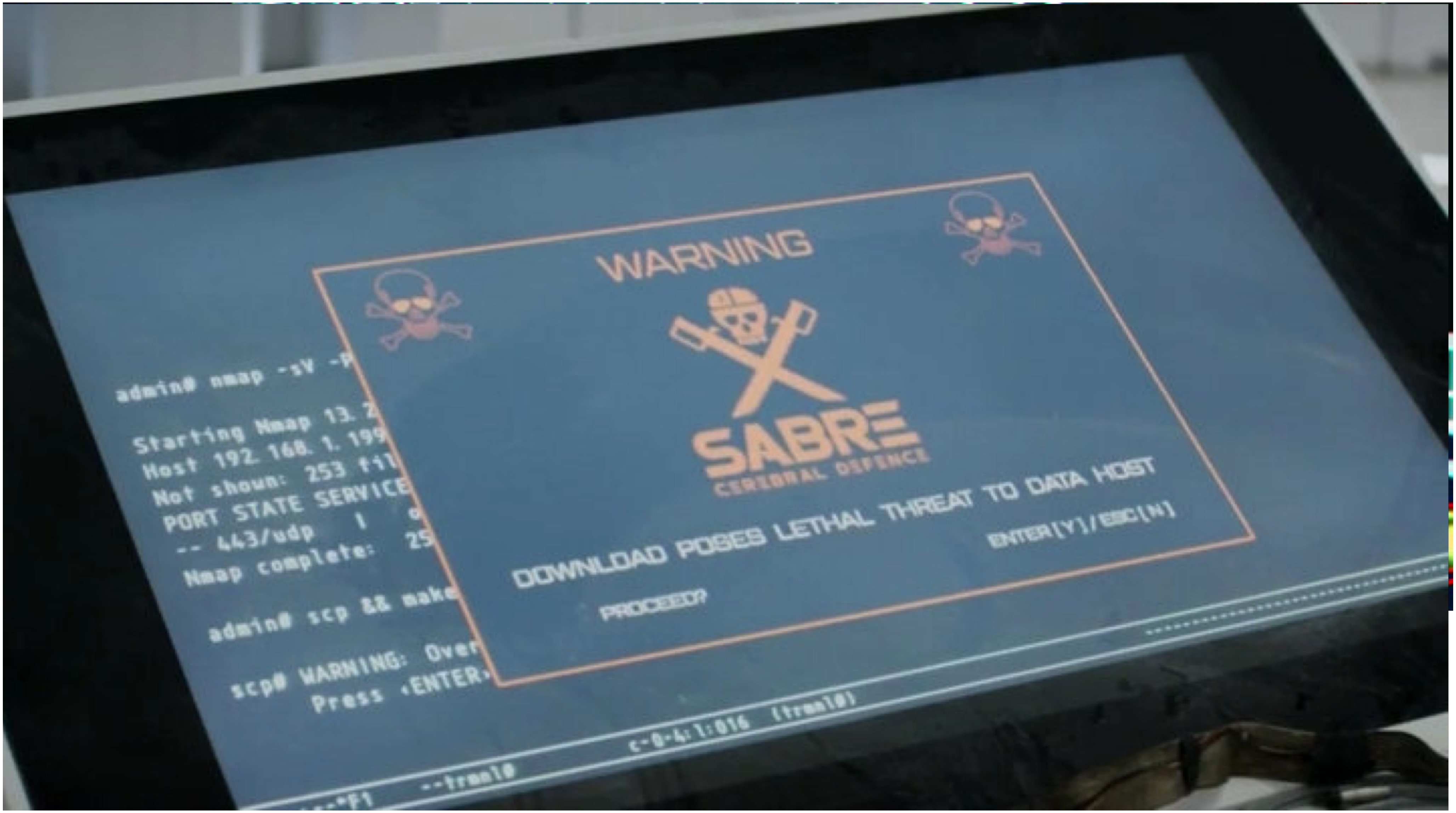
We’ve identified target (range of) IPs, now what?

- Host discovery
  - Narrow broad swath of potential IPs to ones that have hosts associated with them
- Service discovery
  - For a particular host, identify running services
  - E.g., is it accepting SSH connections (22) or HTTP (80)?
- OS fingerprinting
  - Identify the OS software version running
  - E.g., Windows vs Linux?
- Application fingerprinting
  - same at higher level
  - Apache version 1.3 or 2.0+?

NMAP

- Network map tool
- De-facto standard for network reconnaissance, testing
- Numerous built in scanning methods

Used in the Movies



nmap –PN –sT –p 22 192.168.1.0/24

-PN treat all hosts as up
-sT is tcp connect scan
-p 22 is port number

```
[swift:642/background] sudo nmap -PN -sT -p 22 192.168.0.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-01 19:20 CDT
Nmap scan report for 192.168.0.0
Host is up (0.000076s latency).

PORT    STATE    SERVICE
22/tcp filtered ssh

Nmap scan report for 192.168.0.1
Host is up (0.0082s latency).

PORT    STATE SERVICE
22/tcp open  ssh

Nmap scan report for 192.168.0.2
Host is up.

PORT    STATE    SERVICE
22/tcp filtered ssh

Nmap scan report for 192.168.0.3
Host is up.
```

Some of the NMAP status messages

- open
  - host is accepting connections on that port
- closed
  - host responds to NMAP probes on port, but does not accept connections
- filtered
  - NMAP couldn't get packets through to host on that port.
  - Firewall?

nmap –PN –sT –p 22 192.168.1.0/24

```
[swift:642/background] sudo nmap -PN -sT -p 22 192.168.0.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-01 19:20 CDT
Nmap scan report for 192.168.0.0
Host is up (0.000076s latency).

PORT   STATE    SERVICE
22/tcp filtered ssh

Nmap scan report for 192.168.0.1
Host is up (0.0082s latency).

PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for 192.168.0.2
Host is up.

PORT   STATE    SERVICE
22/tcp filtered ssh

Nmap scan report for 192.168.0.3
Host is up.
```

Port scan of host

```
[swift:642/background]  sudo nmap 192.168.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-01 19:21 CDT
Nmap scan report for 192.168.0.1
Host is up (0.0044s latency).
Not shown: 968 closed ports, 28 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
1900/tcp  open  upnp
20005/tcp open  btx
MAC Address: F4:F2:6D:2D:57:C6 (Tp-link Technologies)

Nmap done: 1 IP address (1 host up) scanned in 1.84 seconds
```

Service detection

```
[swift:642/background] sudo nmap -sV 192.168.0.1
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-01 19:22 CDT
Nmap scan report for 192.168.0.1
Host is up (0.0068s latency).
Not shown: 968 closed ports, 28 filtered ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     Dropbear sshd 2012.55 (protocol 2.0)
80/tcp    open  http    TP-LINK Archer C9 WAP http config
1900/tcp  open  upnp?
20005/tcp open  btx?
MAC Address: F4:F2:6D:2D:57:C6 (Tp-link Technologies)
Service Info: OS: Linux; Device: WAP; CPE: cpe:/o:linux:linux_kernel, cpe:/h:tp-
link:archer_c9
```

Gentoo Linux: CVE-2012-0920: Dropbear: Multiple vulnerabilities VULNERABILITY

Severity: 7

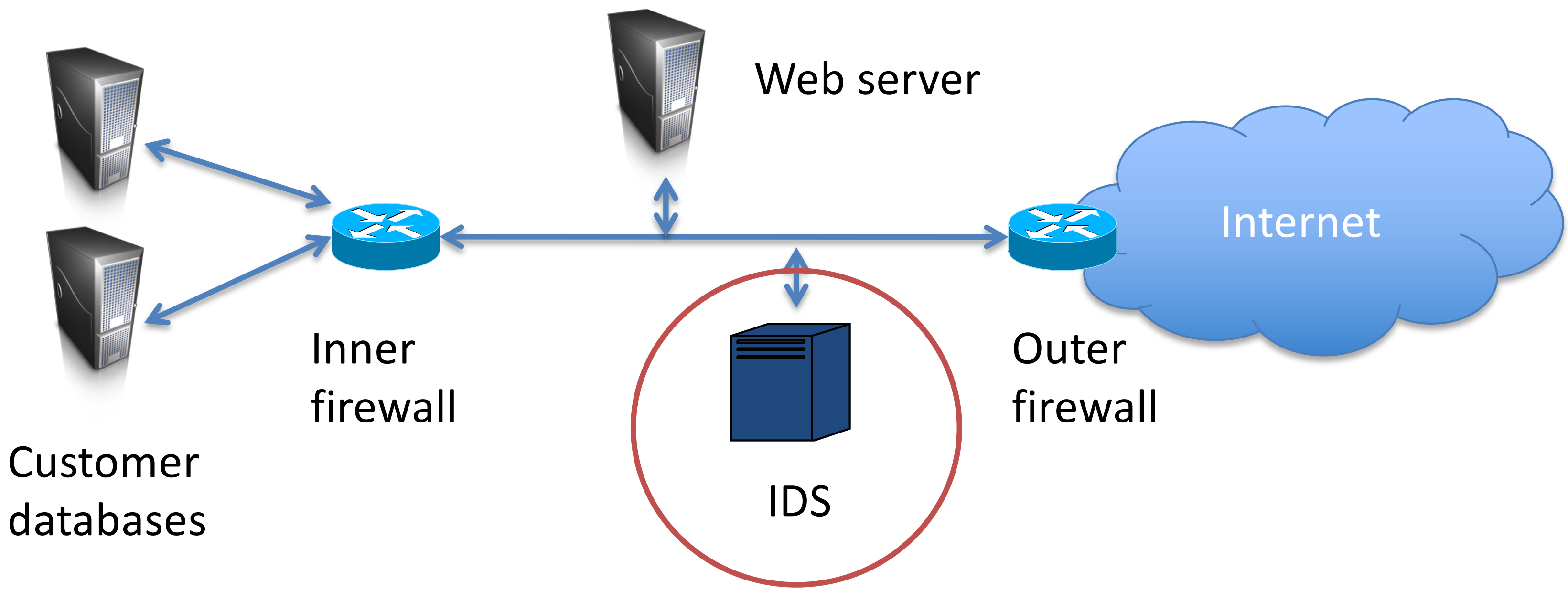
Published: June 05, 2012

Use-after-free vulnerability in Dropbear SSH Server 0.52 through 2012.54, when command restriction and public key authentication are enabled, allows remote authenticated users to execute arbitrary code and bypass command restrictions via multiple crafted command requests, related to "channels concurrency."

OS fingerprinting

```
[swift:~] sudo nmap -O 192.168.0.1
Password:
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-01 19:23 CDT
Nmap scan report for 192.168.0.1
Host is up (0.0032s latency).
Not shown: 968 closed ports, 28 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
1900/tcp  open  upnp
20005/tcp open  btx
MAC Address: F4:F2:6D:2D:57:C6 (Tp-link Technologies)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.31 - 2.6.35
Network Distance: 1 hop
```

Securing Internet Connections



DMZ (demilitarized zone) helps isolate public network components from private network components

Firewall rules to disallow traffic from Internet to internal services

Intrusion Detection/Prevention Systems

- IDS: **monitor** traffic, **alert** operator on attack
- IPS: prevent unsafe **traffic** from passing
- Firewall: prevent unsafe **packets** from passing

- Signature based
  - Define some explicit traffic patterns as bad
  - Flag them
  - E.g., regular expressions
- Anomaly detection
  - What does “normal” traffic look like?
  - Flag abnormal traffic

Taxonomy

- Approach: Policy vs Anomaly
- Location: Network vs. Host
- Action: Detect vs. Prevent
- Semantics: IP vs TCP vs App

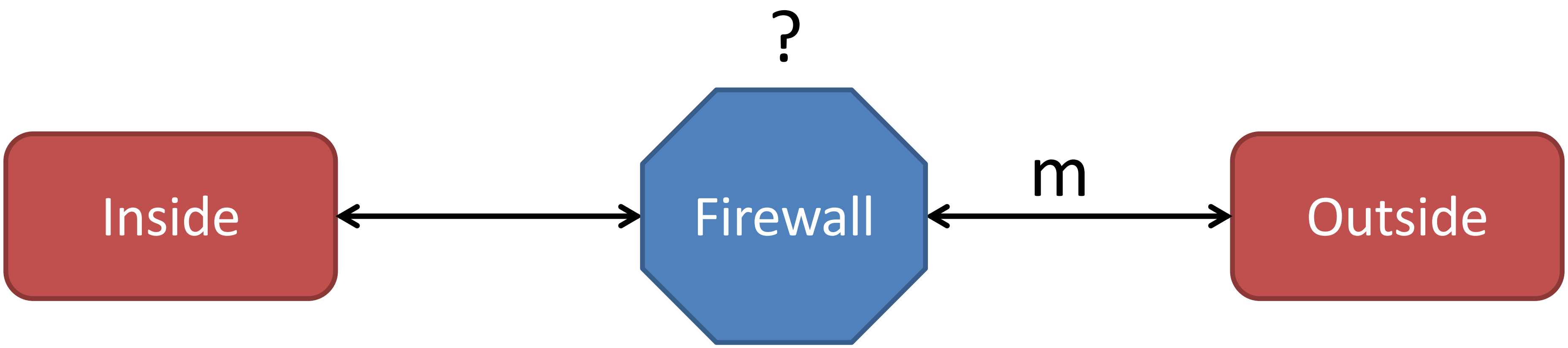
Type	Example
Host, Rule, IDS	Tripwire
Host, Rule, IPS	Personal Firewall
Net, Rule, IDS	Snort
Net, Rule, IPS	Network firewall
Host, Anomaly, IDS	System call monitoring
Net, Anomaly, IDS	Working set of connections
Net, Anomaly, IPS	

Firewall Goals

Provide defense in depth by:

1. Blocking attacks against hosts and services
2. Control traffic between zones of trust

Logical Viewpoint

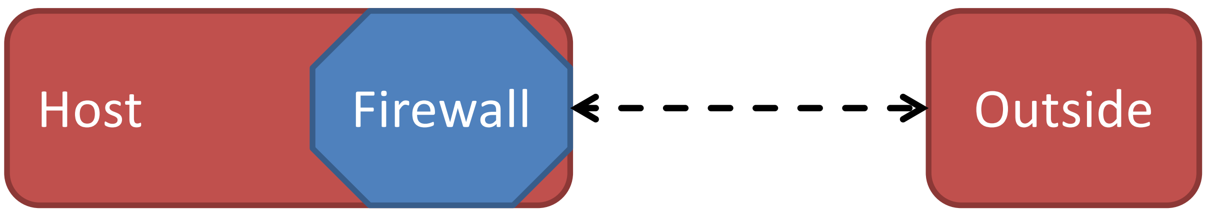


For each message m, either:

- Allow with or without modification
- Block by dropping or sending rejection notice
- Queue

Placement

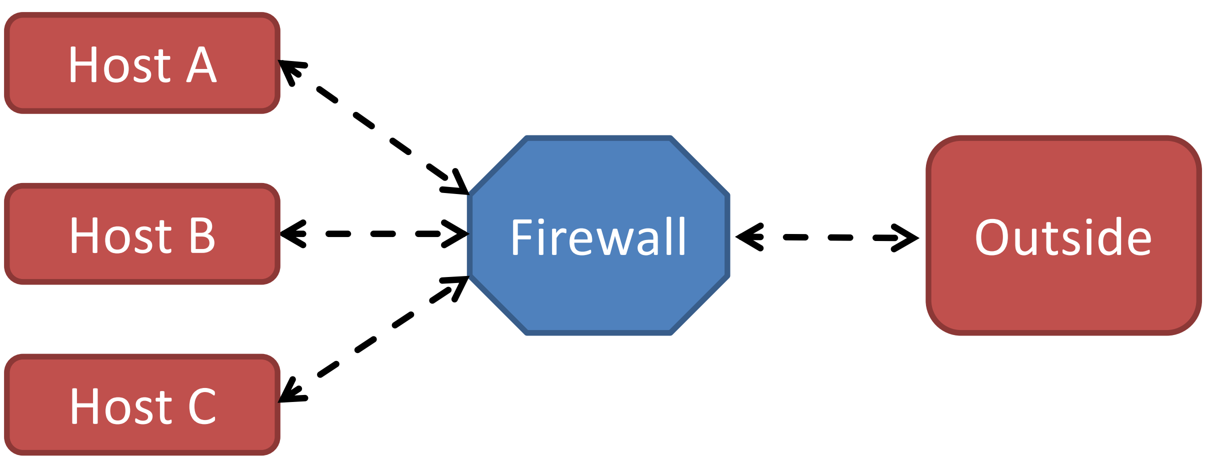
Host-based Firewall



Features:

- Faithful to local configuration
- Travels with you

Network-Based Firewall



Features:

- Protect whole network
- Can make decisions on all of traffic (traffic-based anomaly)

Parameters

Types of Firewalls

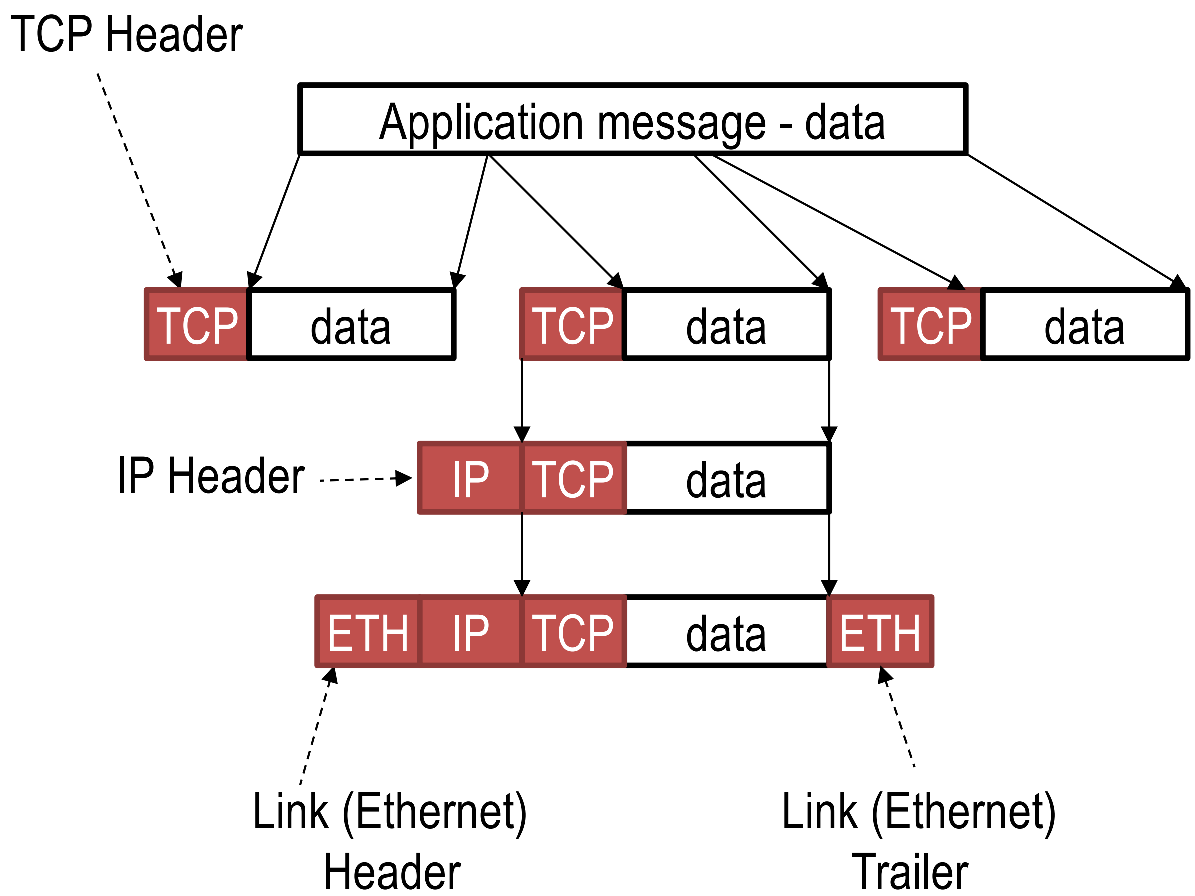
1. Packet Filtering
2. Stateful Inspection
3. Application proxy

Policies

1. Default allow
2. Default deny

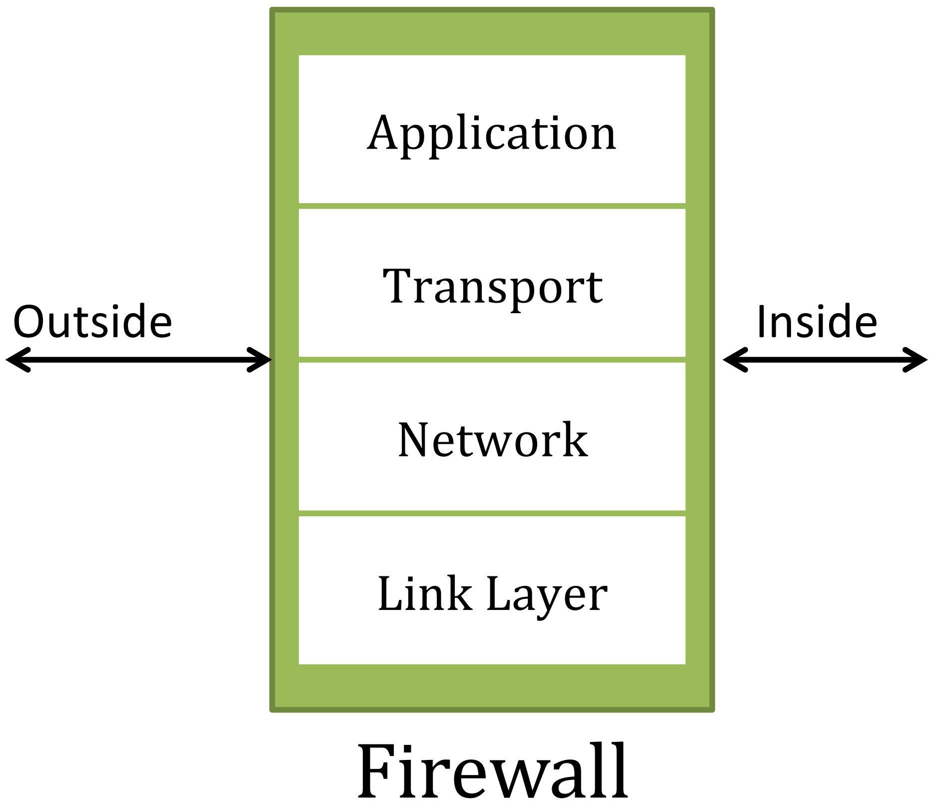
Recall: Protocol Stack

Application (e.g., SSL)
Transport (e.g., TCP, UDP)
Network (e.g., IP)
Link Layer (e.g., ethernet)
Physical



Stateless Firewall

e.g., ipchains in Linux 2.2



Filter by packet header fields

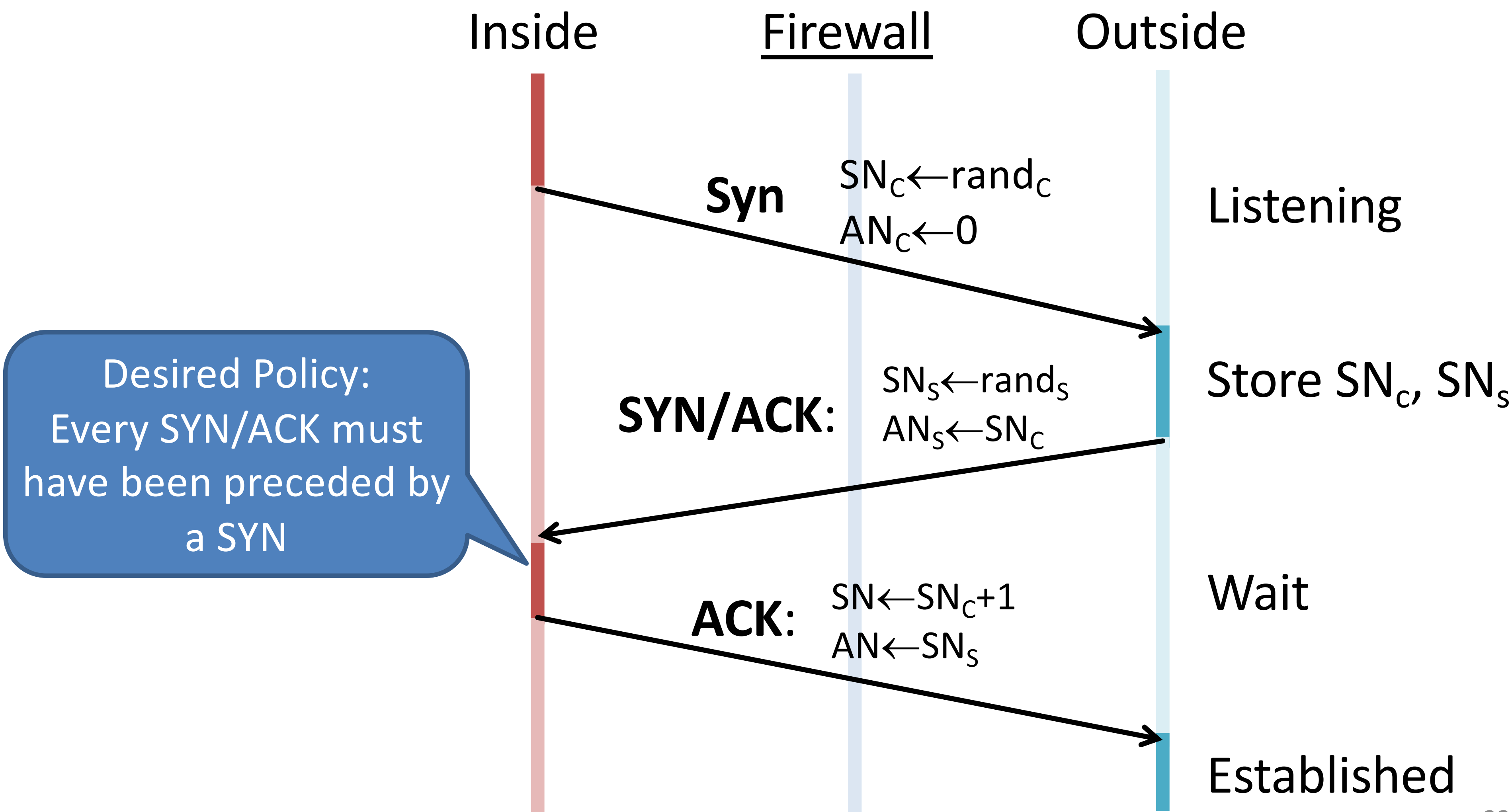
1. IP Field
   (e.g., src, dst)
2. Protocol
   (e.g., TCP, UDP, ...)
3. Flags
   (e.g., SYN, ACK)

Example: only allow incoming DNS packets to nameserver A.A.A.A.

Allow UDP port 53 to A.A.A.A
Deny UDP port 53 all

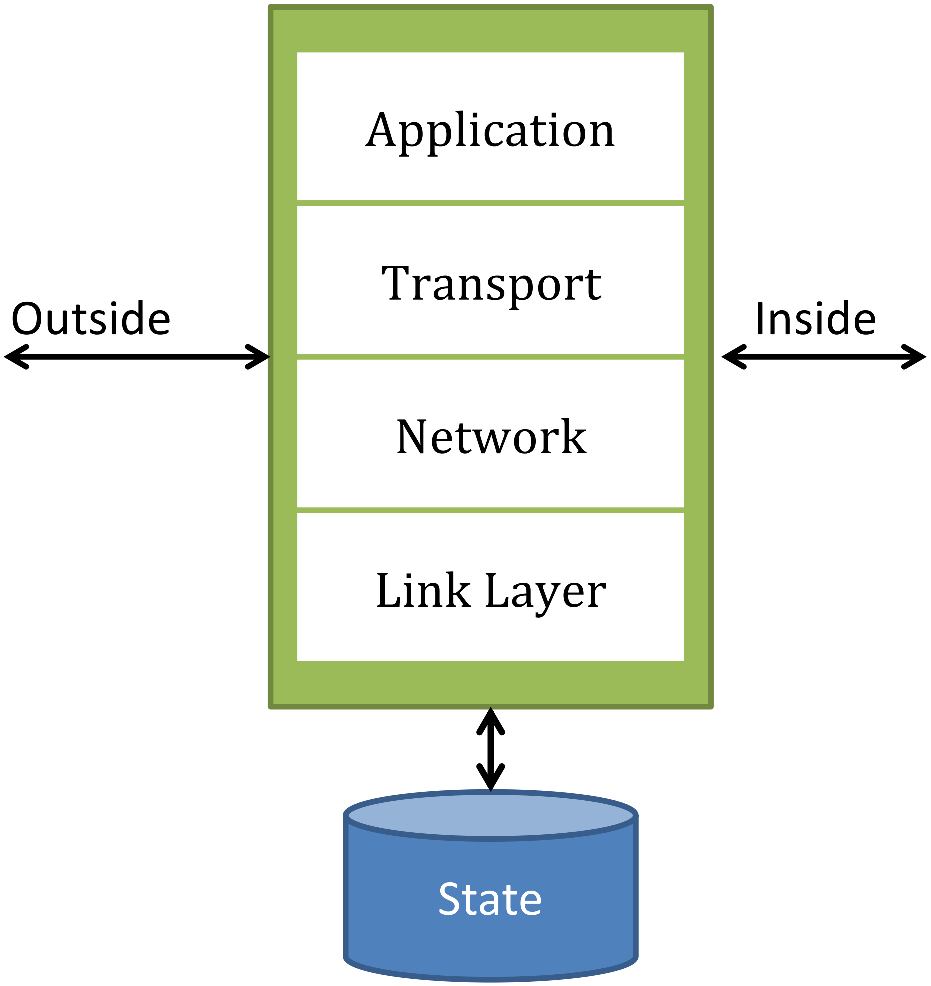
Fail-safe good practice

Need to keep state
Example: TCP Handshake



Stateful Inspection Firewall

e.g., iptables in Linux 2.4

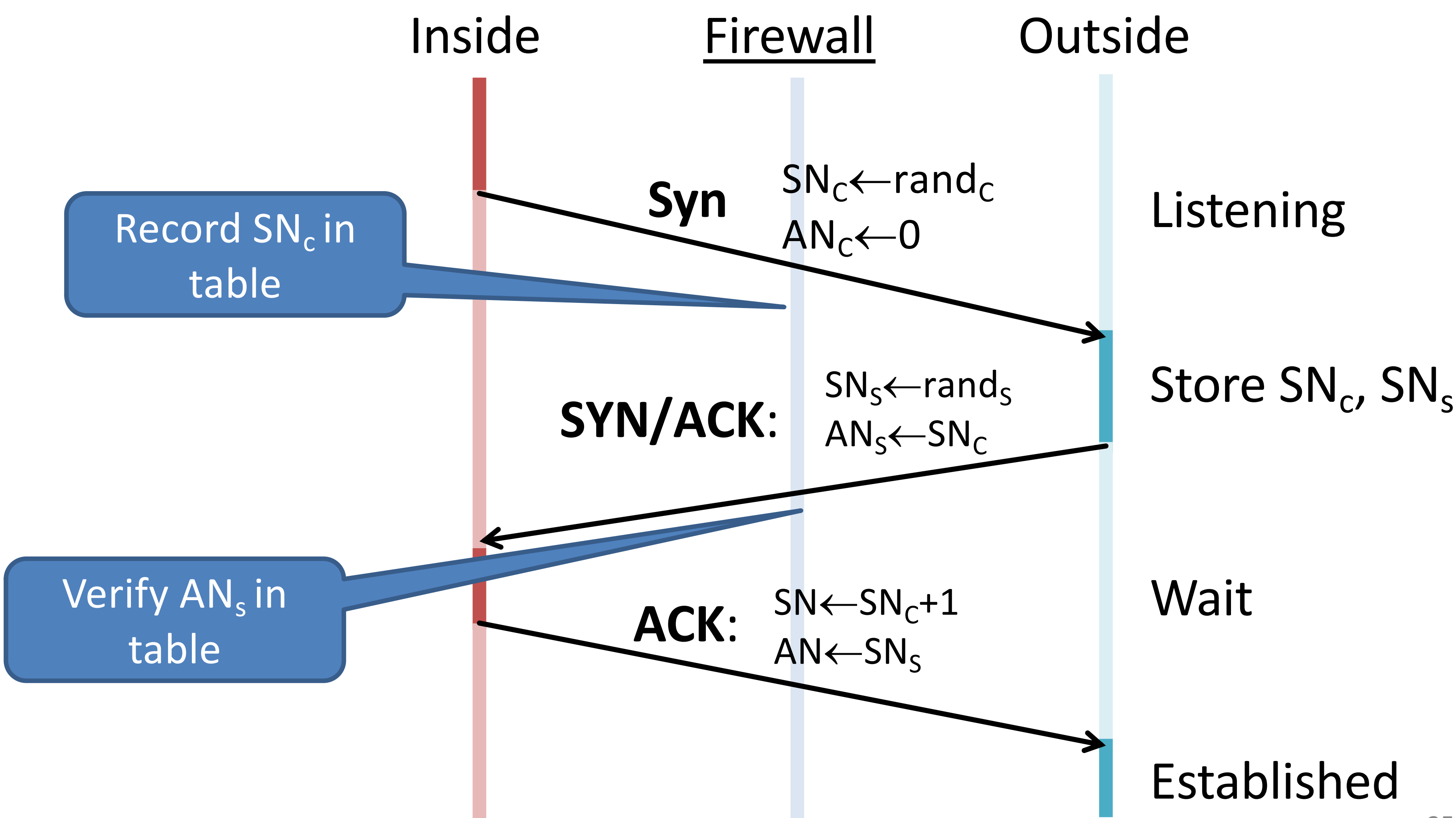


Added state
(plus ***obligation*** to manage)

- Timeouts
- Size of table

Stateful More Expressive

Example: TCP Handshake



Stateful Firewalls

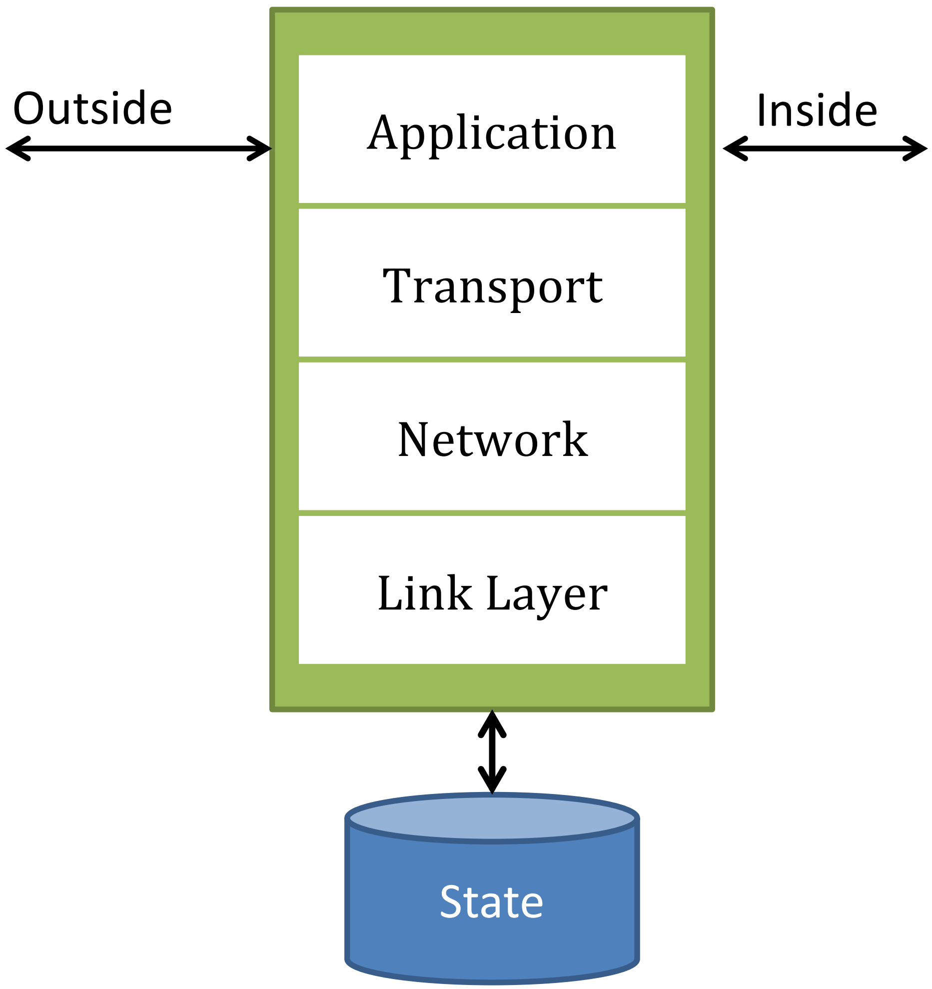
Pros

- More expressive

Cons

- State-holding attack
- Mismatch between firewalls understanding of protocol and protected hosts

Application Firewall



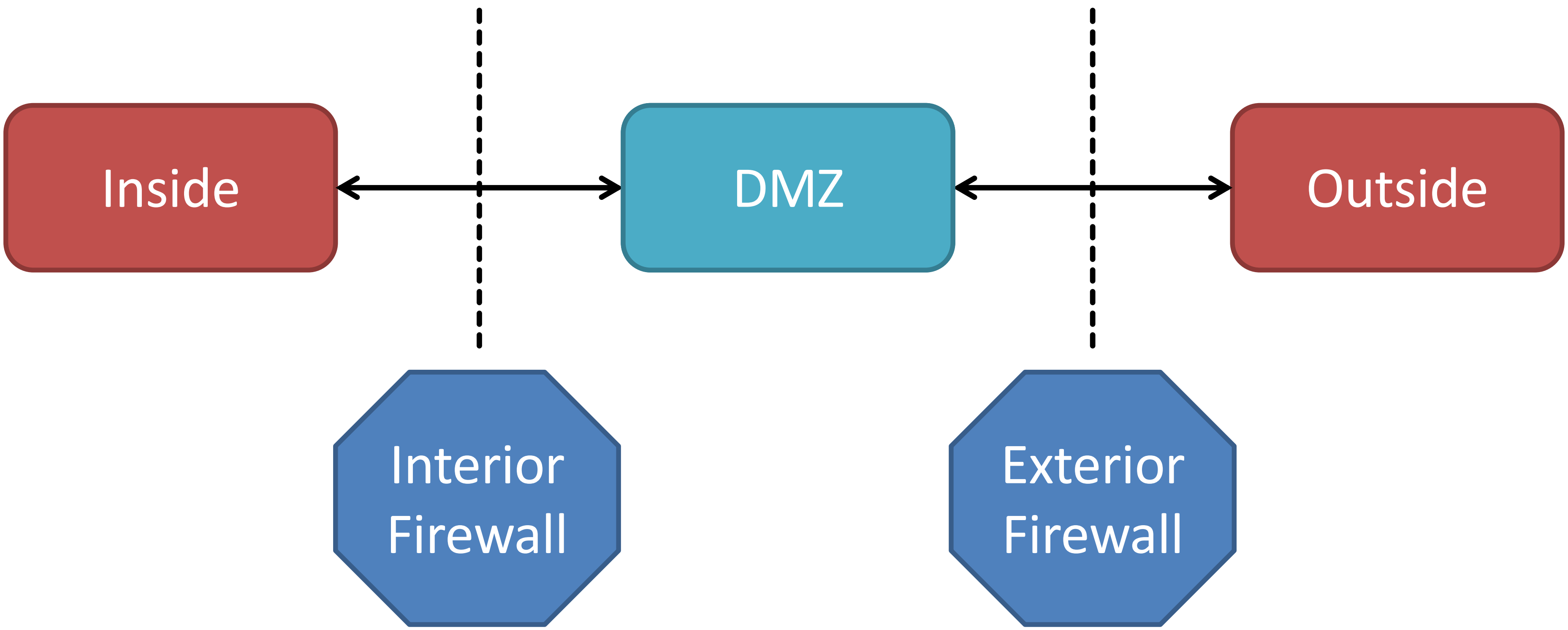
Check protocol messages directly

Clients connect to firewall, firewall connects to server

Examples:

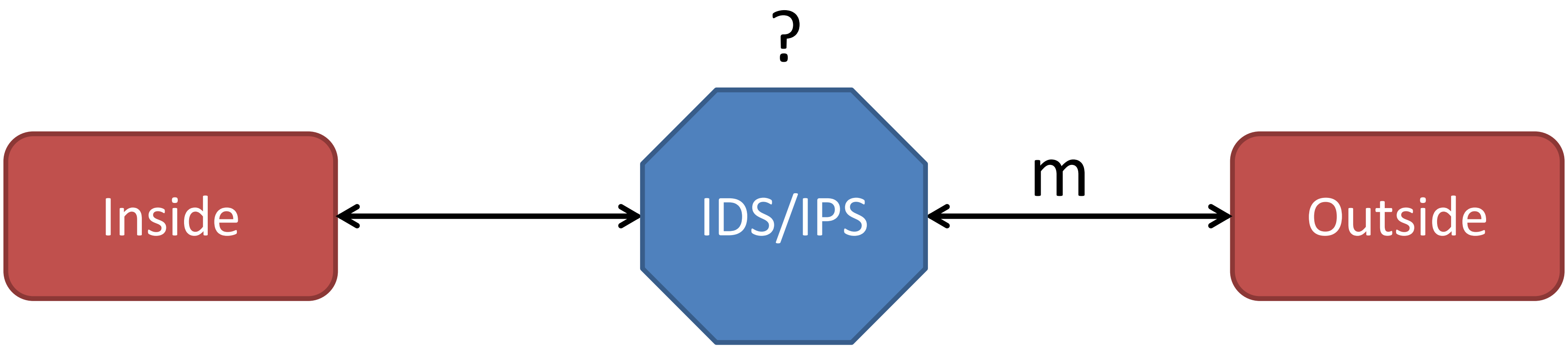
- Web Proxies

Dual Firewall



INTRUSION DETECTION AND PREVENTION SYSTEMS

Logical Viewpoint



For each message m, either:

- Report *m* (IPS: drop or log)
- Allow *m*
- Queue

Overview

- <u>Approach:</u> Policy vs Anomaly
- <u>Location:</u> Network vs. Host
- <u>Action:</u> Detect vs. Prevent
- <u>Semantics:</u> “looks deeper”

Policy-Based IDS

Use pre-determined rules to detect attacks

Examples: Regular expressions (snort),
Cryptographic hash (tripwire, snort)

Detect any fragments less than 256 bytes

```
alert tcp any any -> any any (minfrag: 256; msg:
  "Tiny fragments detected, possible hostile activity";)
```

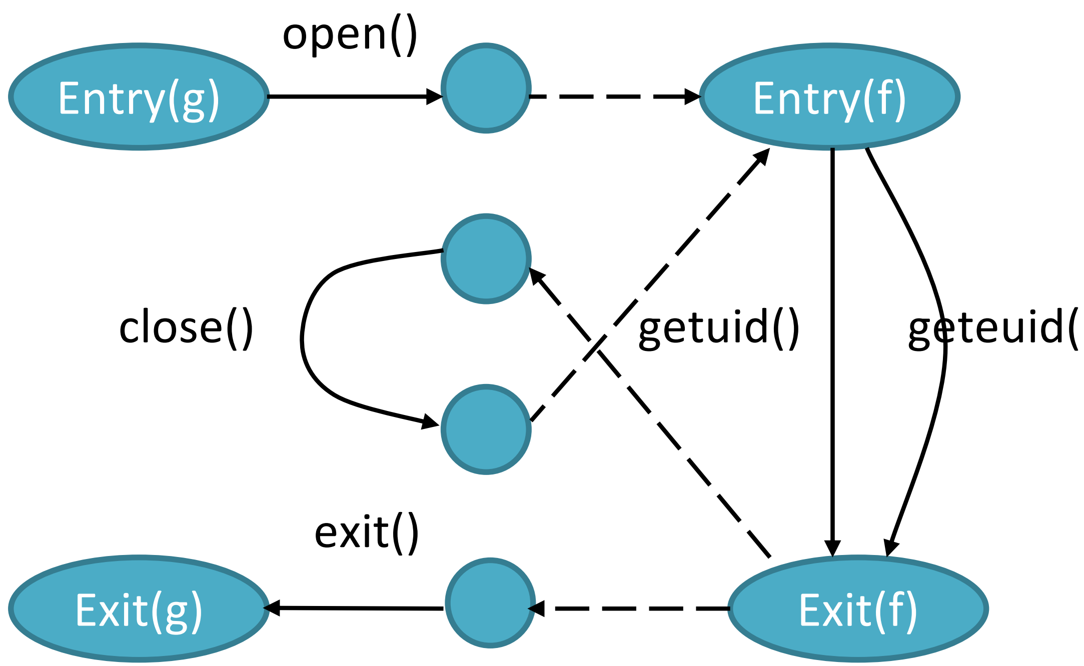
Detect IMAP buffer overflow

```
alert tcp any any -> 192.168.1.0/24 143 (
  content: "|90C8 C0FF FFFF|/bin/sh";
  msg: "IMAP buffer overflow!”;)
```

Example Snort rules

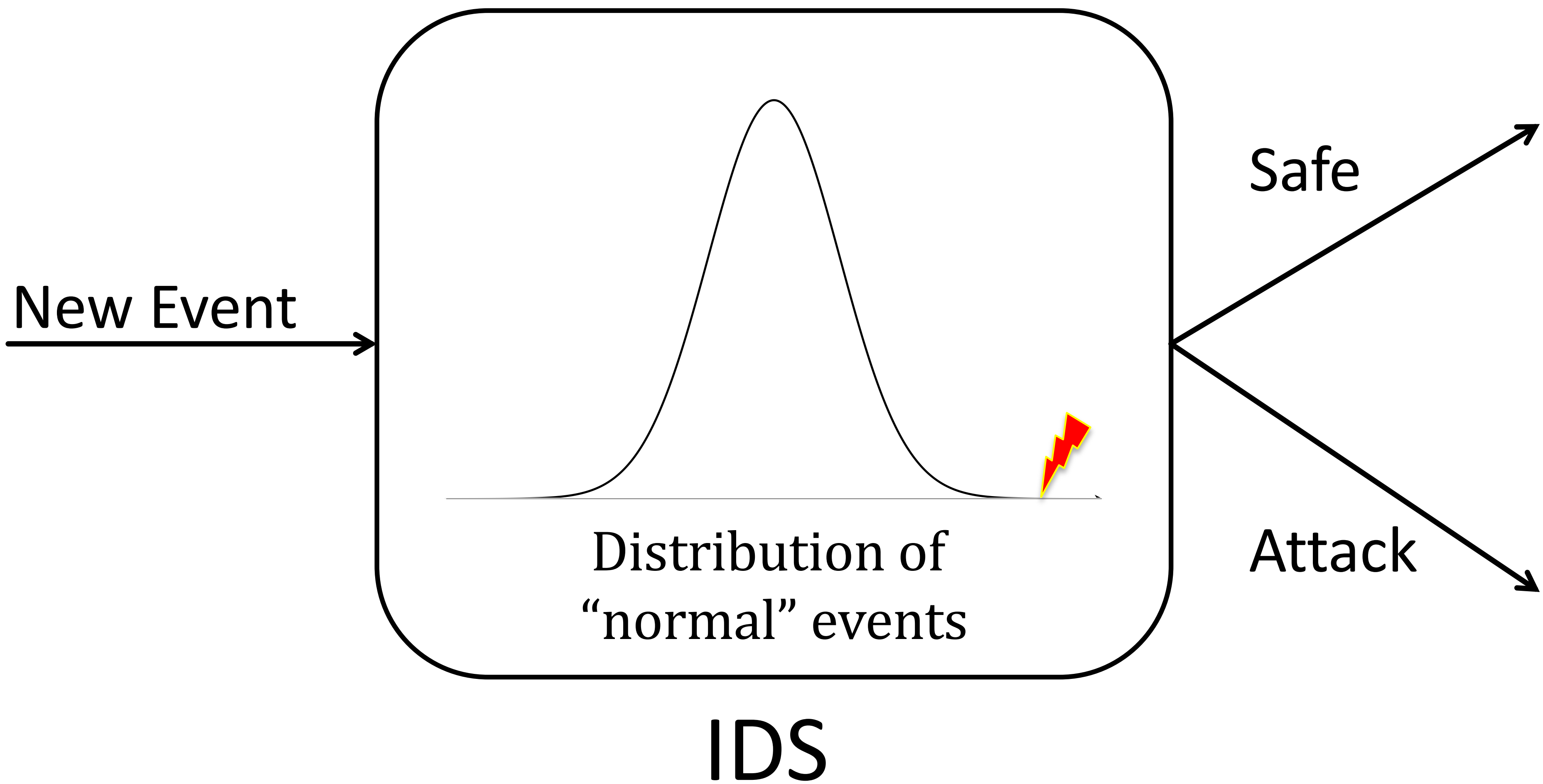
OS Intrusion Detection via System Calls
[wagner&dean 2001]

```
f(int x) {
  if(x){ getuid(); } else{ geteuid();}
  x++;
}
g() {
  fd = open("foo", O_RDONLY);
  f(0); close(fd); f(1);
  exit(0);
}
```

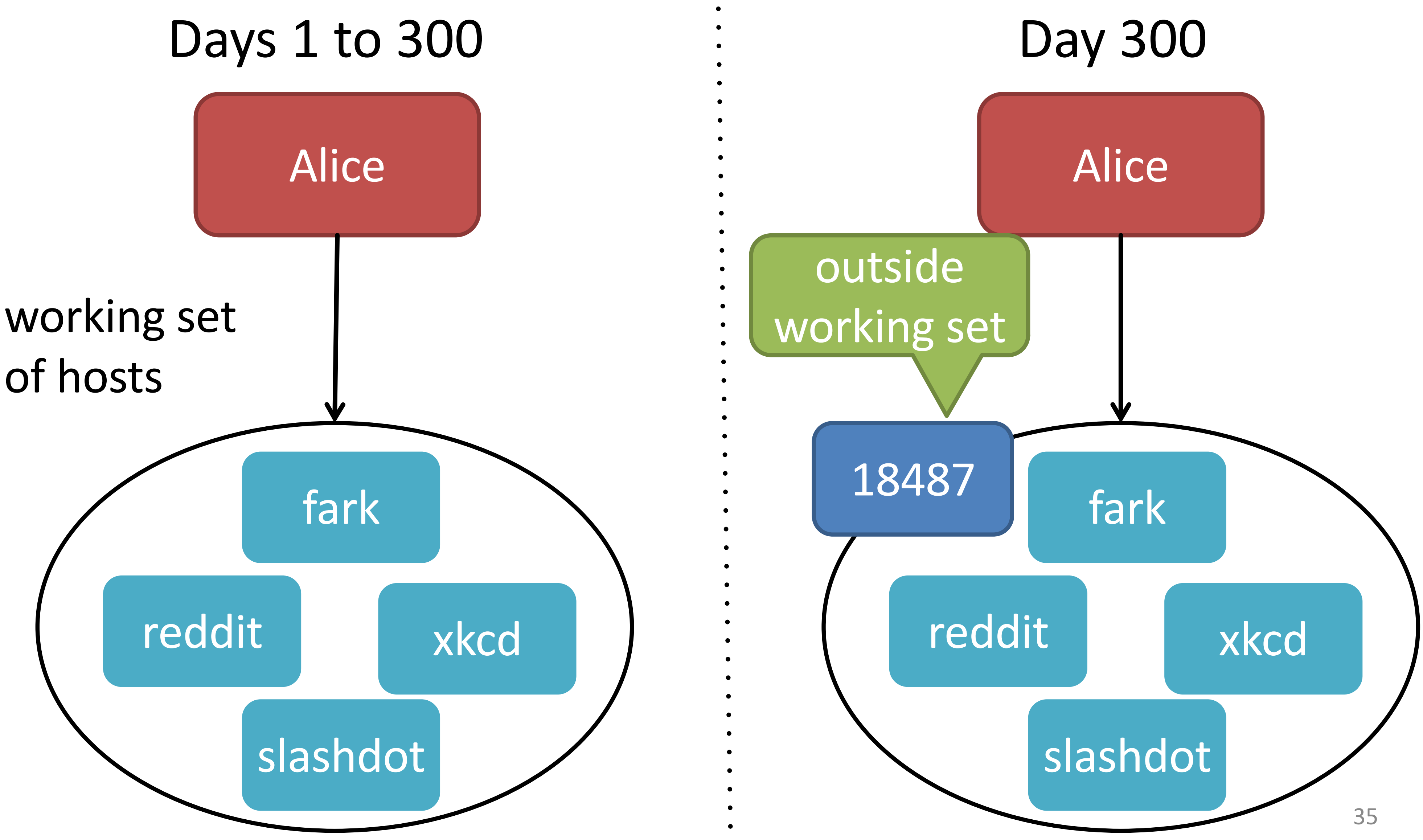


Execution inconsistent with automata indicates attack

Anomaly Detection



Example: Working Sets



Anomaly Detection

Pros

- Does not require pre-determining policy (an “unknown” threat)

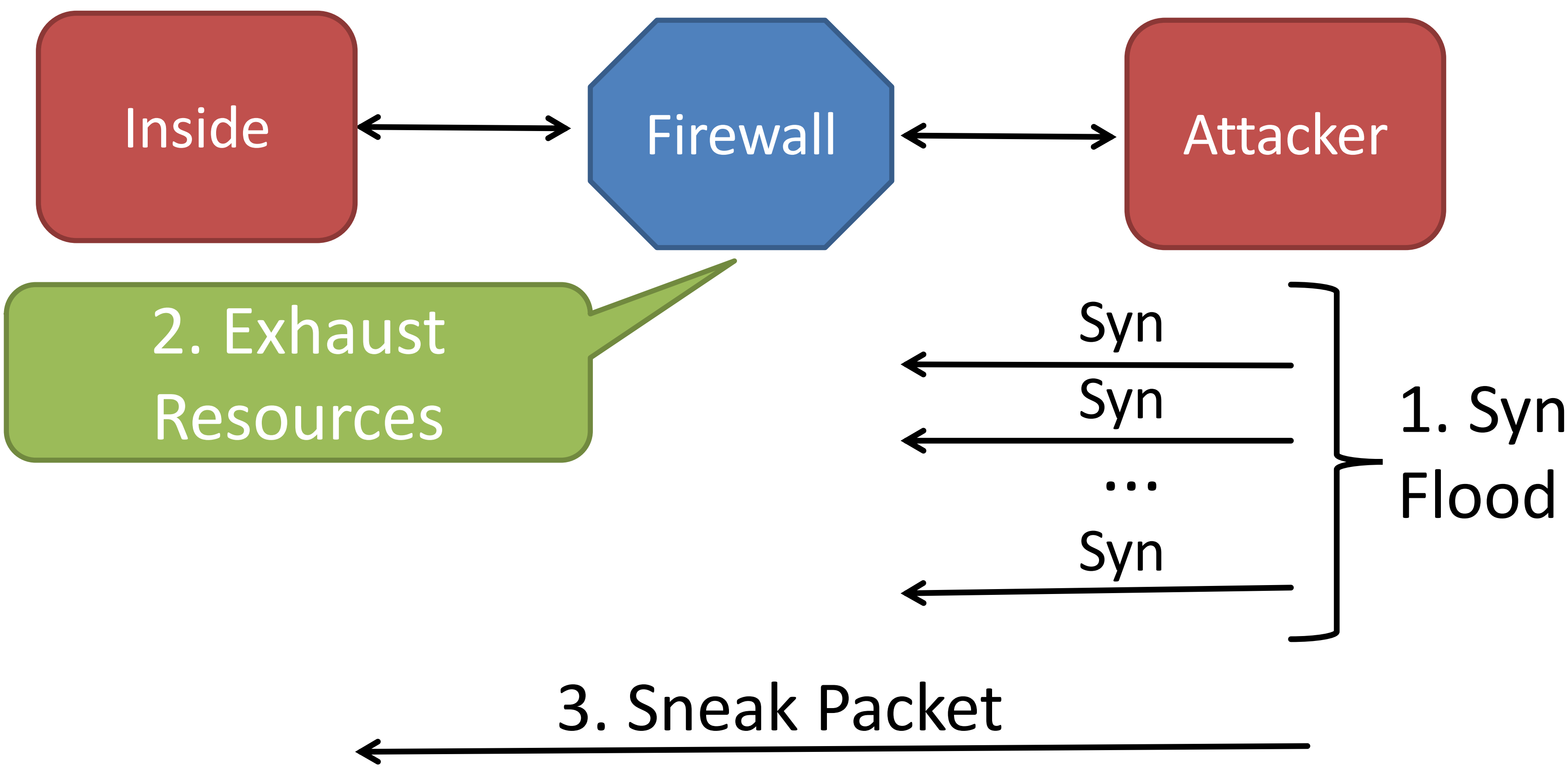
Cons

- Requires attacks are not strongly related to known traffic
- Learning distributions is hard

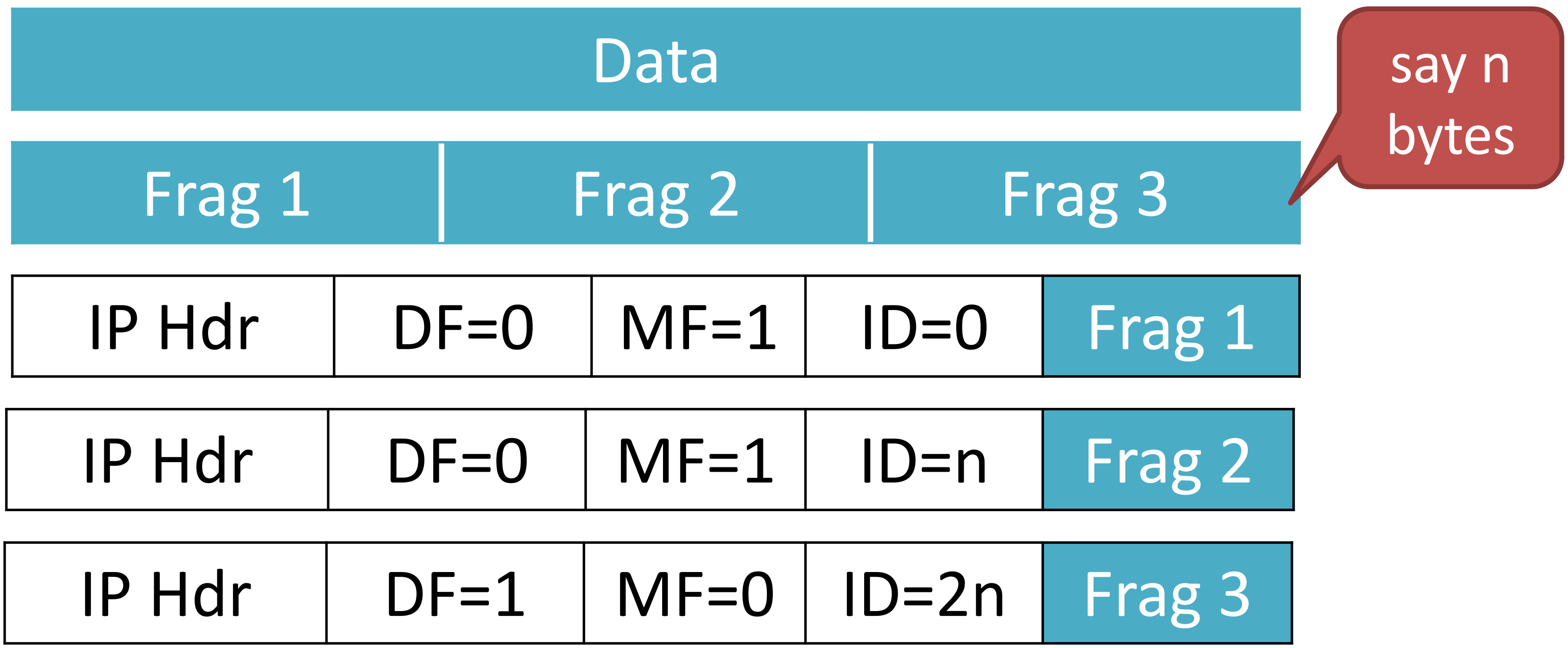
ATTACKS AND EVASION

State Holding Attack

Assume stateful TCP policy



Fragmentation

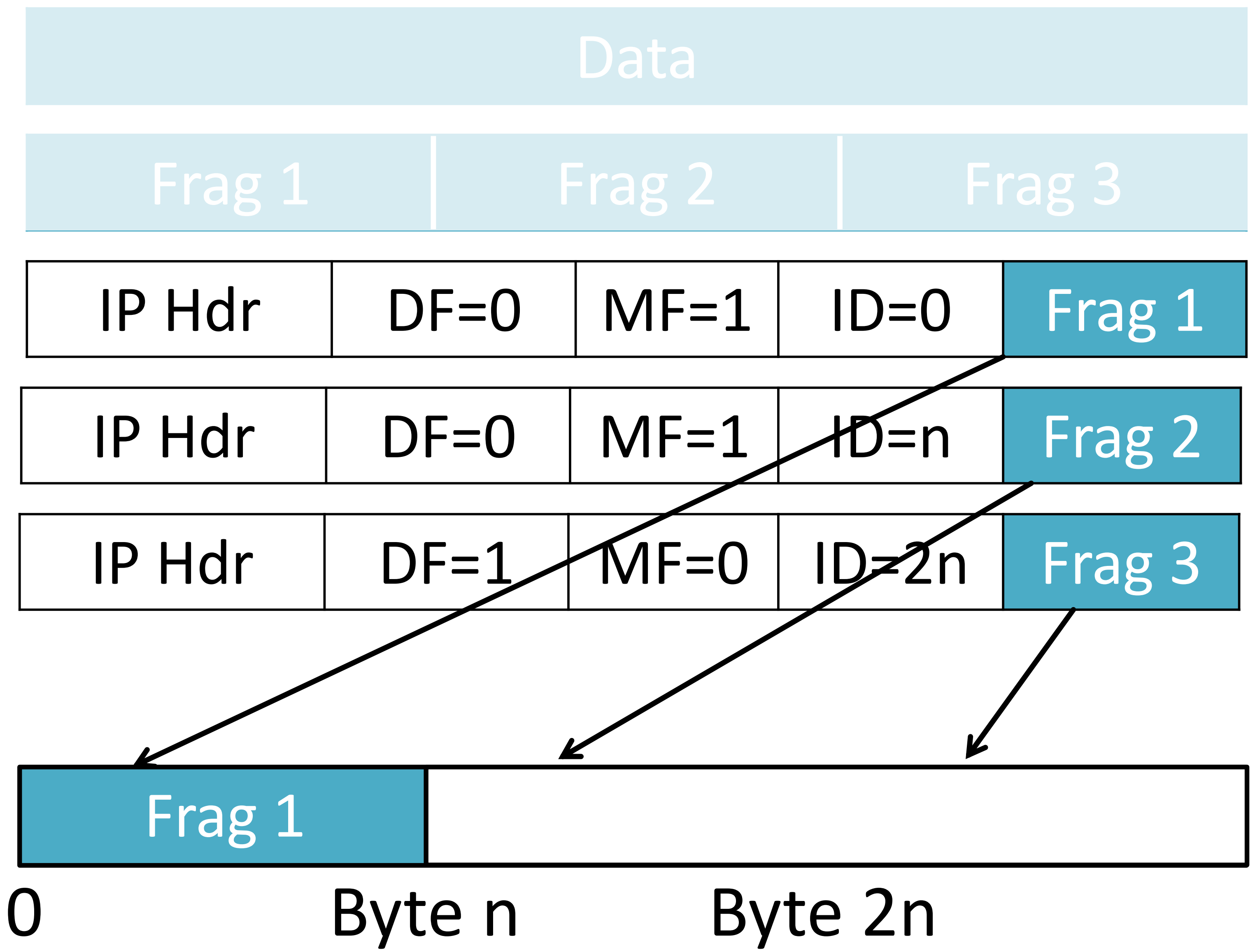


DF : Don’t fragment
(0 = May, 1 = Don’t)
MF: More fragments
(0 = Last, 1 = More)
Frag ID = Octet number

Octet 1 Octet 2 Octet 3 Octet 4

<table>
<tr><td>Ver</td><td>IHL</td><td>TOS</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="3">ID</td><td>0</td><td>D
F</td><td>M
F</td><td>Frag ID</td></tr>
<tr><td colspan="7">...</td></tr>
</table>

Reassembly


Data
Frag 1
Frag 2
Frag 3
IP Hdr
DF=0
MF=1
ID=0
Frag 1
IP Hdr
DF=0
MF=1
ID=n
Frag 2
IP Hdr
DF=1
MF=0
ID=2n
Frag 3
Frag 1
0
Byte n
Byte 2n

Overlapping Fragment Attack

Assume Firewall Policy:

☑ Incoming Port 80 (HTTP)

☒ Incoming Port 22 (SSH)

Packet 1	…	DF=1	MF=1	ID=0	…	1234 (src port)	80 (dst port)	…
Packet 2	…	DF=1	MF=1	ID=2	…	22		…

TCP Hdr (Data!)

Octet 1	Octet 2	Octet 3	Octet 4
1234		22	
Sequence Number			
….			

Bypass policy

TTL attack example

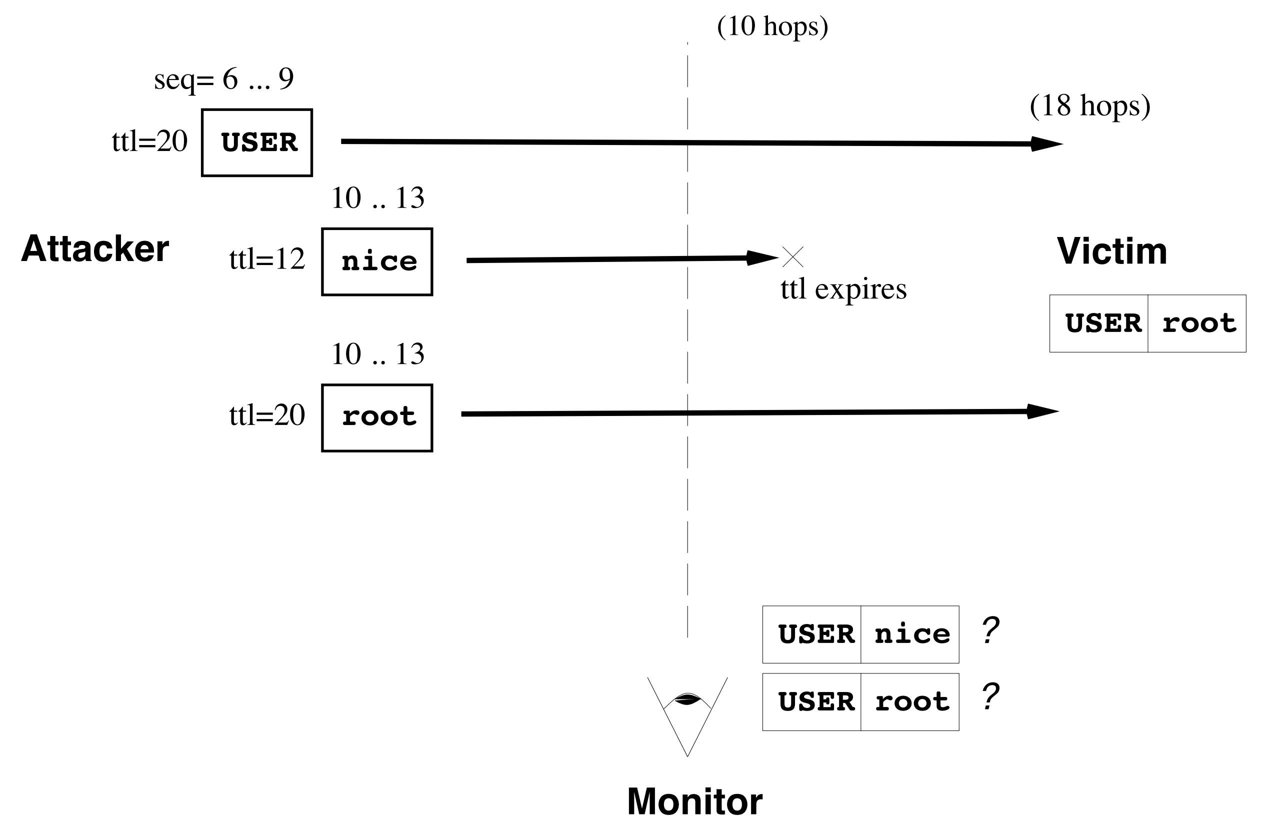
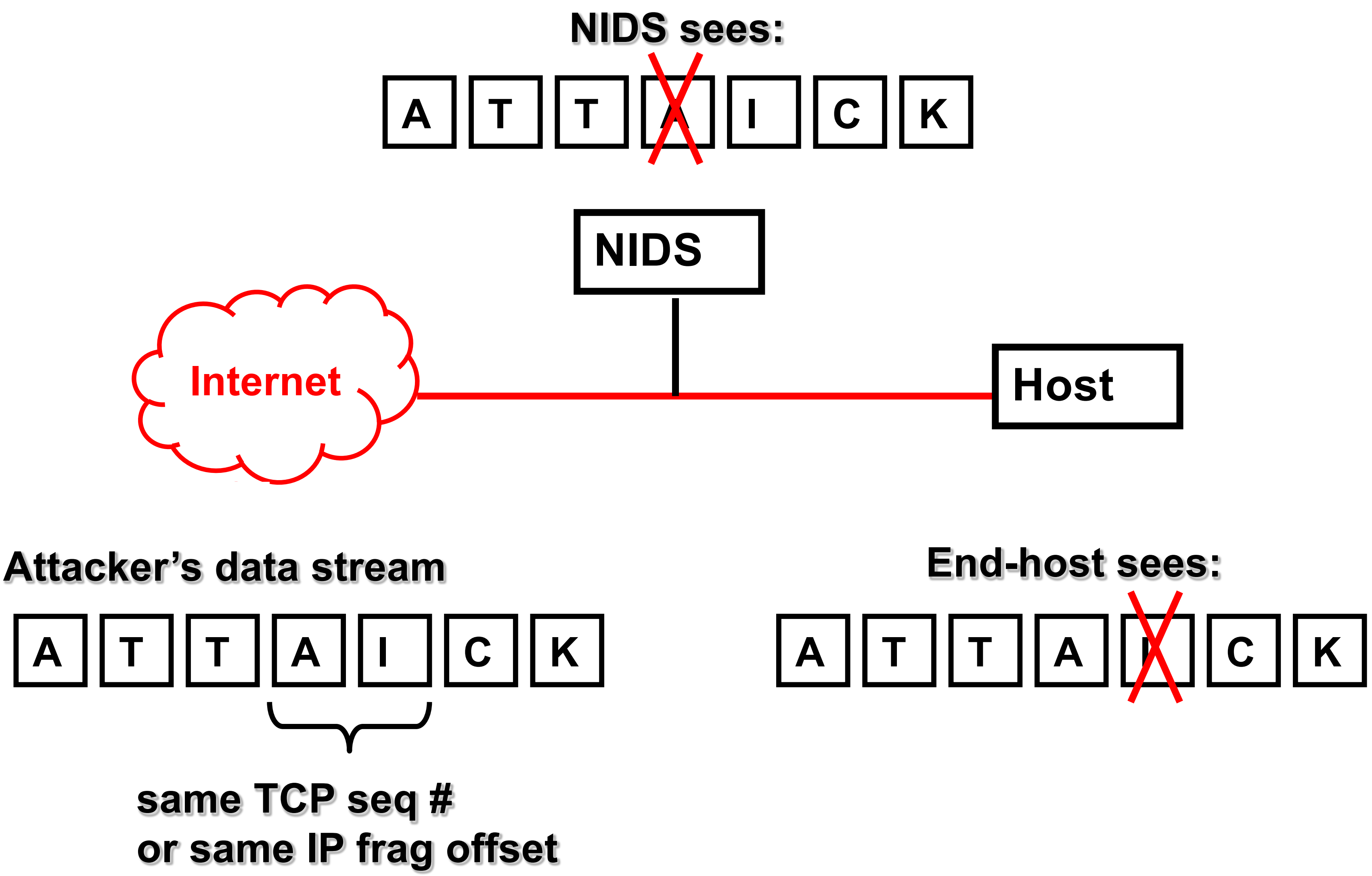


Figure 2: A TTL-based evasion attack on an intrusion detection system

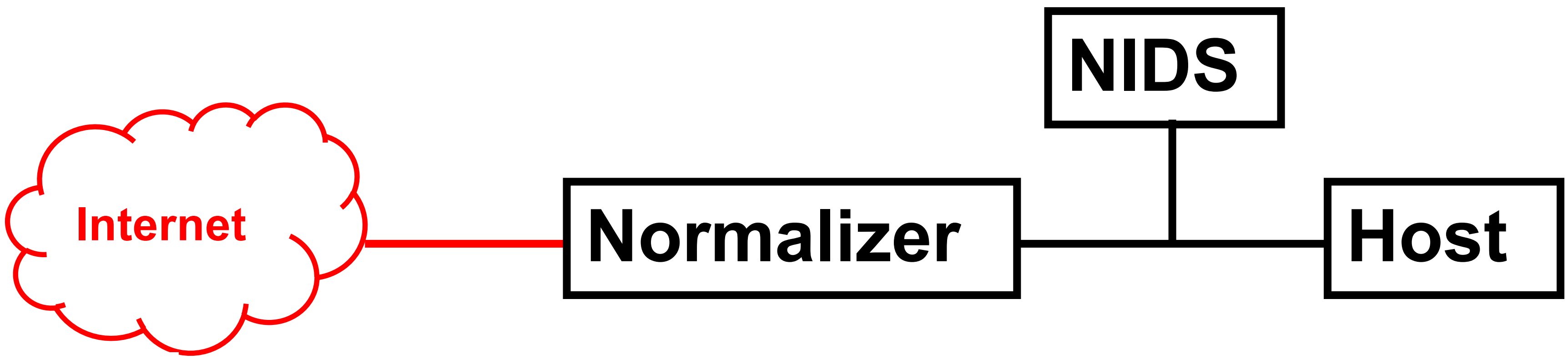
From Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", 1999

Fragmentation overlap attack



Potential Fix: traffic normalizer

Introduce “bump in the wire”: traffic normalizer to evade protocol ambiguities



Key problem

- IDS may not see what hosts see
  - Different packet orders
  - Different packets (TTL)
  - Resolve ambiguities differently (fragments)

Algorithmic complexity attacks

- DoS attack on IDS enables other attacks to remain undetected
- Example: hash tables

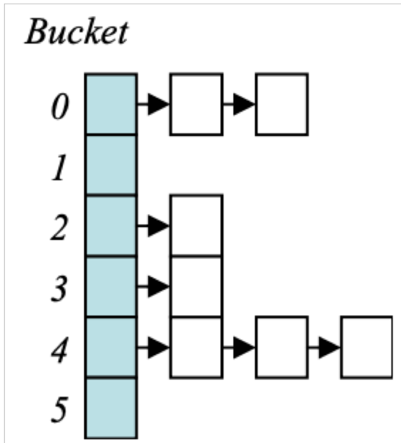


Figure 1: Normal operation of a hash table.

From: Denial of Service via Algorithmic Complexity Attacks

Base Rate, fallacies, and detection systems

DETECTION THEORY

How hard is it to detect anomalies

- Given normal events, how accurate does a detector have to be?

Ω

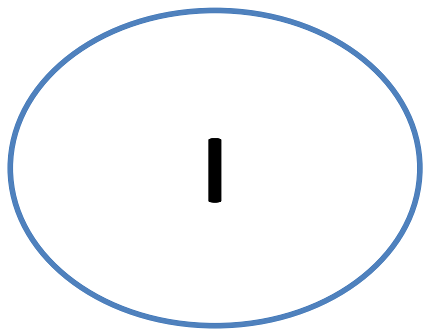
Let Ω be the set of all possible events.

For example:

- Audit records produced on a host
- Network packets seen

Ω

<u>Example</u>: IDS Received 1,000,000 packets.
20 of them corresponded to an intrusion.
The <u>*intrusion rate*</u> Pr[I] is:
Pr[I] = 20/1,000,000 = .00002



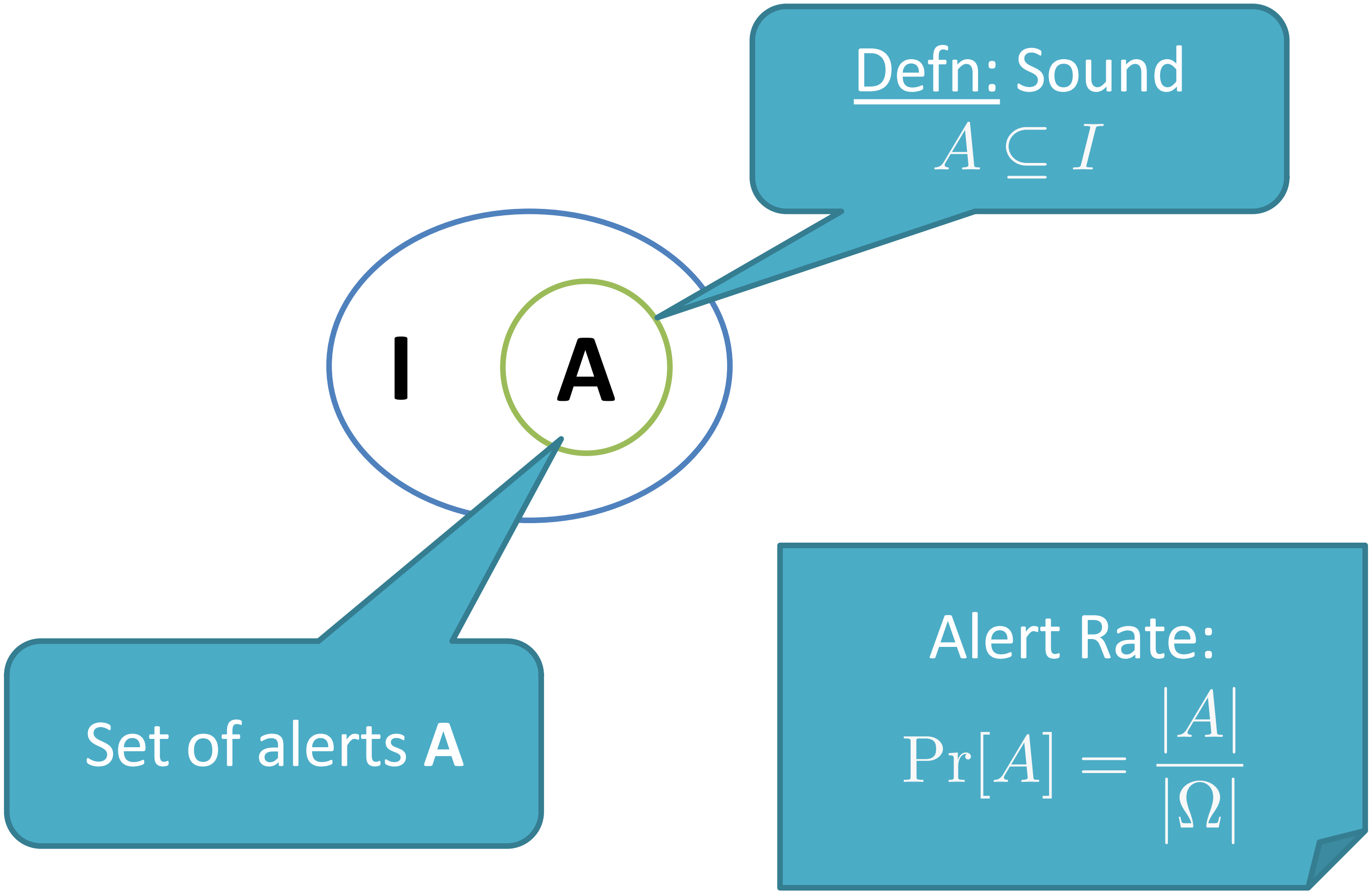
Set of intrusion events **I**

Intrusion Rate:

$$\Pr[I] = \frac{|I|}{|\Omega|}$$

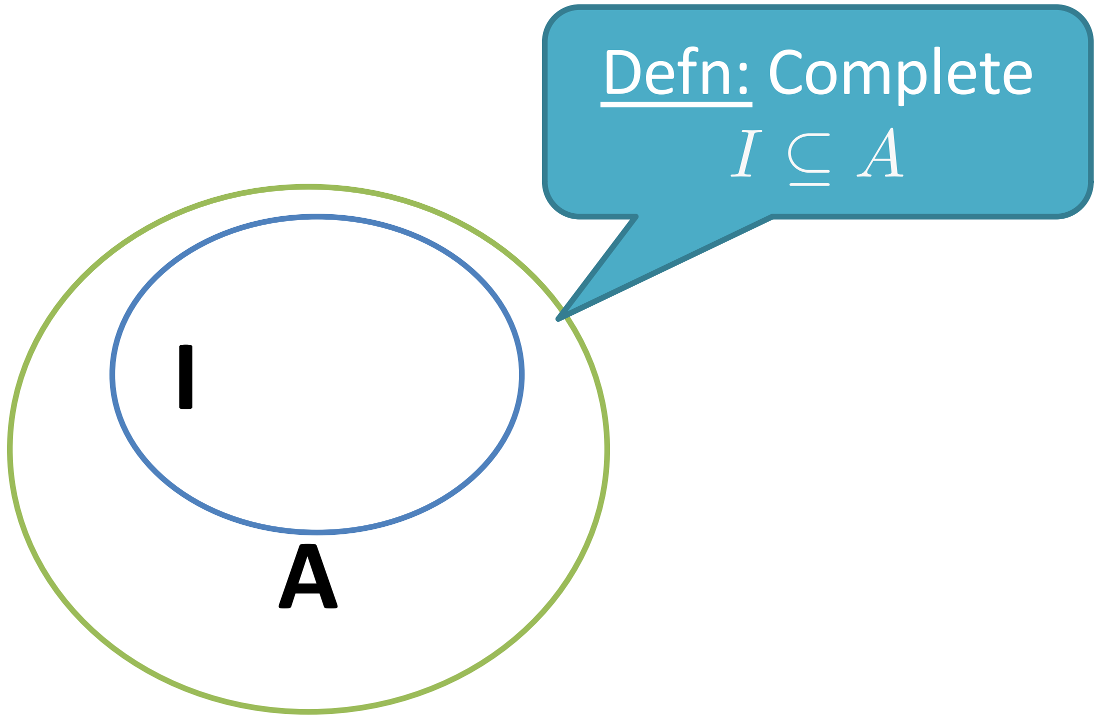
Ω

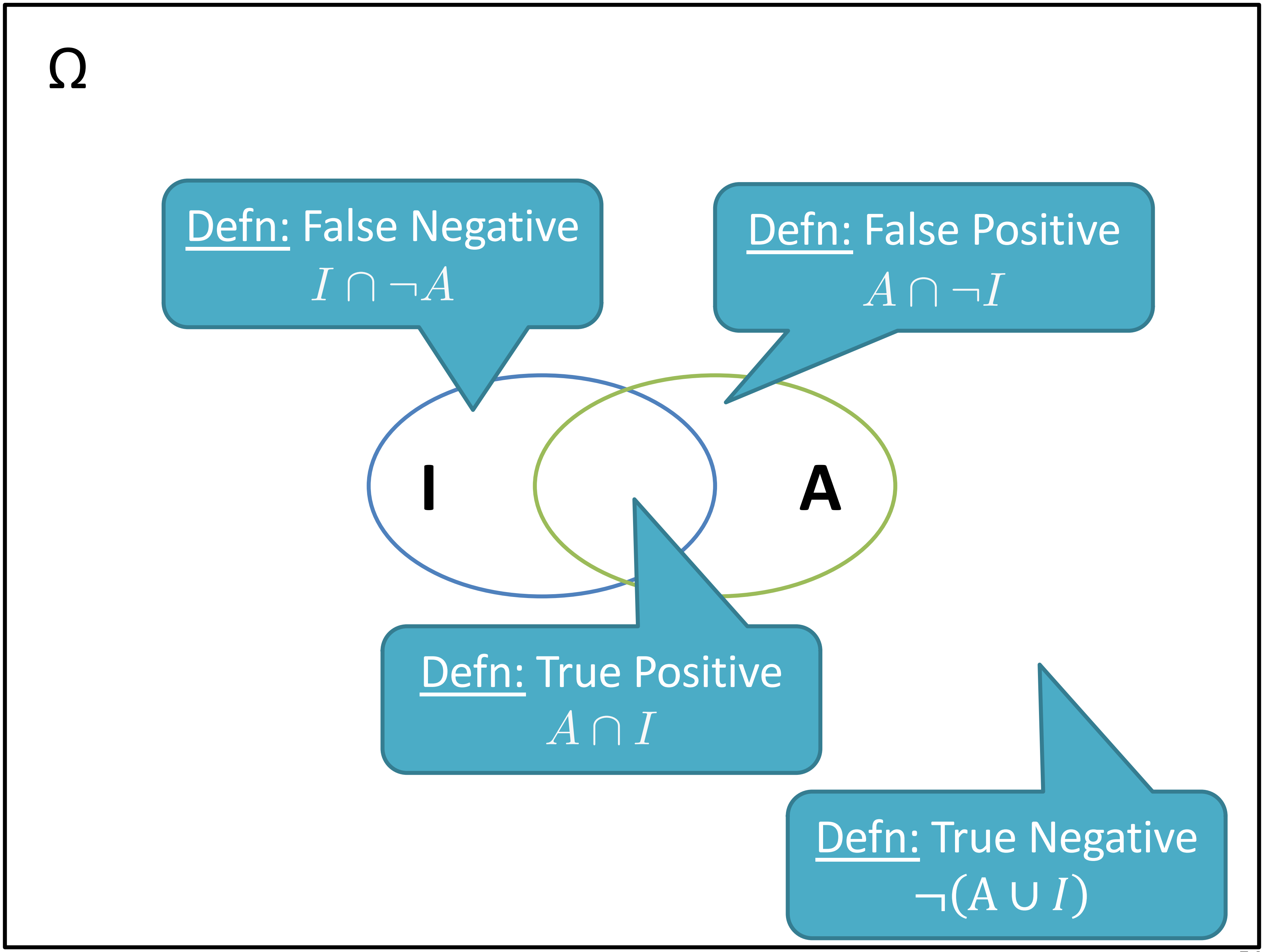
Sound: Alerts are all intrusions



Ω

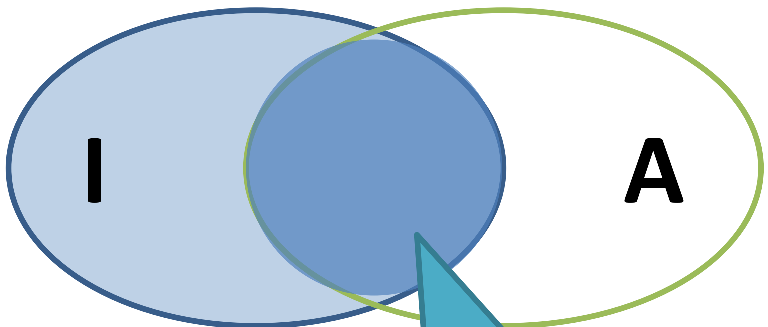
Complete: All intrusions generate alerts




Ω
Defn: False Negative
$I \cap \neg A$
Defn: False Positive
$A \cap \neg I$
I
A
Defn: True Positive
$A \cap I$
Defn: True Negative
$\neg(A \cup I)$

Ω

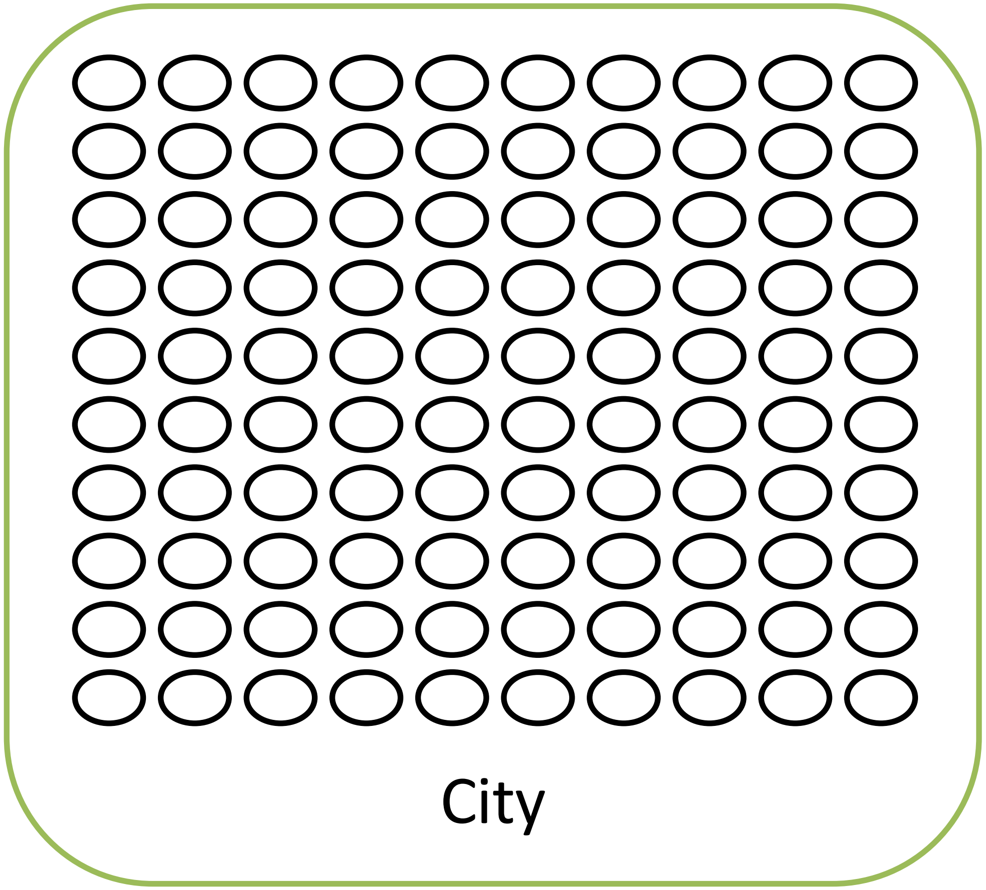
Think of the detection rate as the set of *intrusions raising an alert* normalized by the *set of <u>all</u> intrusions*.



<u>Defn:</u> Detection rate

$$\Pr[A|I] = \frac{\Pr[A \cap I]}{\Pr[I]}$$

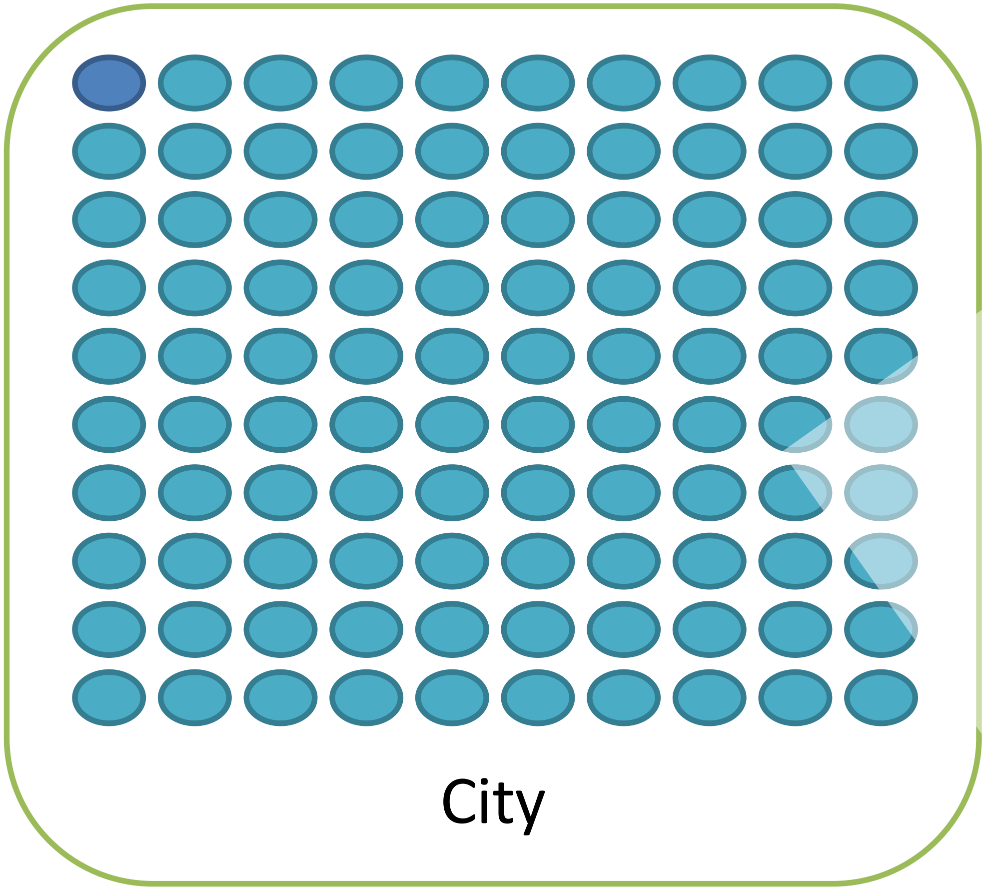
Example



(this times 10)

- 1,000 people in the city
- 1 is a terrorist, and we have their pictures. Thus the ***base rate*** of terrorists is 1/1000
- Suppose we have a new terrorist facial recognition system that is 99% accurate.
  - 99/100 times when someone is a terrorist there is an alarm
  - For every 100 good people, the alarm only goes off once.
- An alarm went off. Is the suspect really a terrorist?

Example

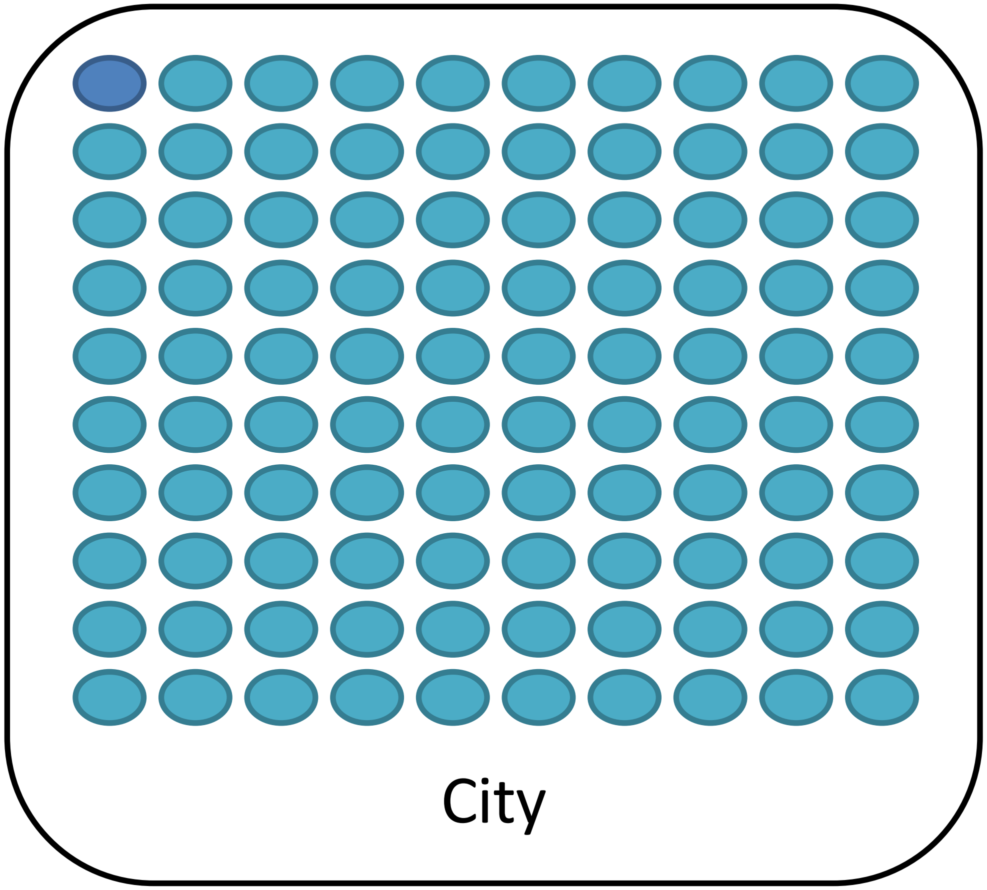


(this times 10)

Answer: The facial recognition system is 99% accurate. That means there is only a 1% chance the guy is not the terrorist.

Wrong!

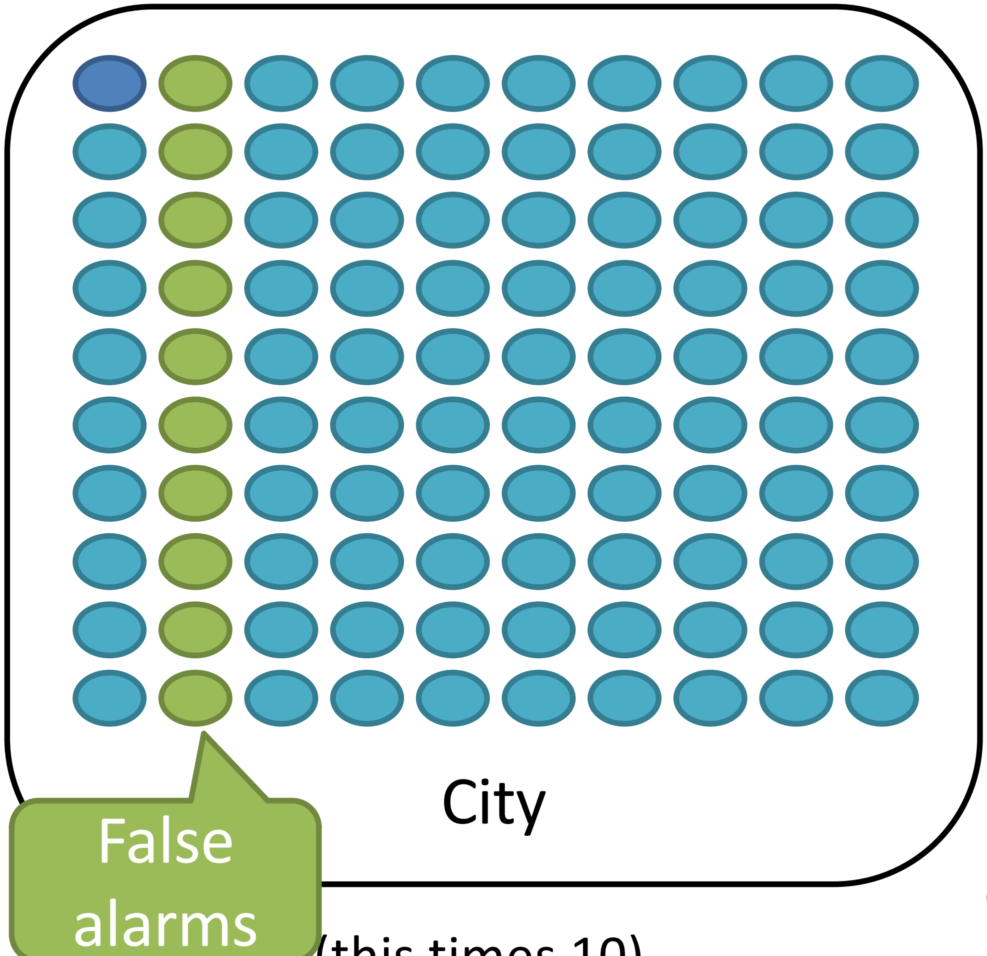
Formalization



(this times 10)

- 1 is terrorists, and we have their pictures. Thus the ***base rate*** of terrorists is 1/1000.
  P[T] = 0.001
- 99/100 times when someone is a terrorist there is an alarm.
  P[A|T] = .99
- For every 100 good guys, the alarm only goes off once.
  P[A | not T] = .01
- Want to know **P[T|A]**

Intuition: Given 999 good guys, we have 999*.01 ≈ 9-10 false alarms



(this times 10)

- 1 is terrorists, and we have their pictures. Thus the ***base rate*** of terrorists is 1/1000. **P[T] = 0.001**
- 99/100 times when someone is a terrorist there is an alarm. **P[A|T] = .99**
- For every 100 good guys, the alarm only goes off once. **P[A | not T] = .01**
- Want to know **P[T|A]**

$$\text{Have: } \Pr[T] = 0.001$$

$$\Pr[A|T] = .99, \Pr[A|\neg T] = .01$$

$$\text{Want to calculate: } \Pr[T|A] = \frac{\Pr[T \cap A]}{\Pr[A]}$$

Unknown

Unknown

Calculating Probabilities

- Probability of an alert and a terrorist is:
  - Probability of an alert given a terrorist * probability of a terrorist
  - **$Pr[A \cap T] = Pr[A|T]*P[T]$**
- Probability of an alert is:
  - Probability of an alert given a terrorist * probability of a terrorist
  - Probability of an alert given not a terrorist * probability not a terrorist
  - **$Pr[A] = Pr[T]* Pr[A|T] + Pr[\neg T] * Pr[A|\neg T]$**

$$\text{Have: } \Pr[T] = 0.001$$

$$\Pr[A|T] = .99, \Pr[A|\neg T] = .01$$

$$\text{Want to calculate: } \Pr[T|A] = \frac{\Pr[T \cap A]}{\Pr[A]}$$

✓

✓

$$= \frac{\Pr[A|T] * \Pr[T]}{\Pr[T] * \Pr[A|T] + \Pr[\neg T] + \Pr[A|\neg T]}$$

Probability alert finds terrorist

$$\text{Have: } \Pr[T] = 0.001$$

$$\Pr[A|T] = .99, \Pr[A|\neg T] = .01$$

$$\text{Want to calculate: } \Pr[T|A] = \frac{\Pr[A|T]*\Pr[T]}{\Pr[T]*\Pr[A|T]+\Pr[\neg T]+\Pr[A|\neg T]}$$

$$\frac{.99 * .001}{.001 * .99 + .999 * .01}$$

$$= 0.\overline{09} \approx \boxed{9\%}$$

With Pr[A|T] = 0.999, still only 50% of alerts are terrorists

Why is anomaly detection hard

For infrequent events: must be very accurate to avoid false positives

Using anomaly detection:
- Easy to learn common and legal events
- Hard to learn rare but legal events

Bottom line: decide how bad are false positives?

Conclusion

- Firewalls
  - Ttypes: Packet filtering, Stateful, and Application
  - Placement and DMZ
- IDS
  - Anomaly vs. policy-based detection
- How can we exploit for evasion?
  - E.g., fragmentation, TCP session reassembly, TTL
- How can we attack the defense infrastructure?
  - E.g., overload, algorithmic complexity
- Detection theory
  - Base rate fallacy