CSE 490K

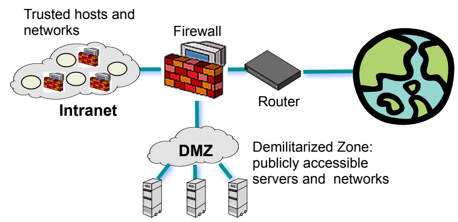
Firewalls and Network Defense

Tadayoshi Kohno

Some slides based on Vitaly Shmatikov's

Firewalls

- Idea: separate local network from the Internet



Castle and Moat Analogy

- More like the moat around a castle than a firewall
  - Restricts access from the outside
  - Restricts outbound connections, too (!!)
    - Important: filter out undesirable activity from internal hosts!

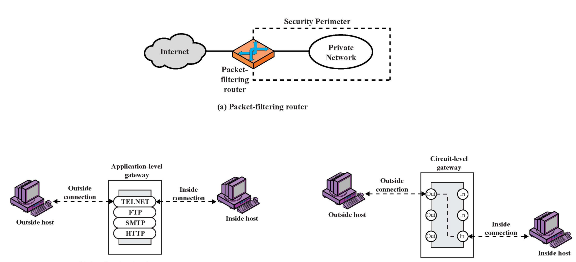
Firewall Locations in the Network

- 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"
  - Microsoft's Internet Connection Firewall (ICF) comes standard with Windows XP

Firewall Types

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

Firewall Types: Illustration



Packet Filtering

- For each packet, firewall decides whether to allow it to proceed
  - Decision must be made on per-packet basis
    - Stateless; cannot examine packet's context (TCP connection, application to which it belongs, etc.)
- To decide, use information available in the packet
  - IP source and destination addresses, ports
  - Protocol identifier (TCP, UDP, ICMP, etc.)
  - TCP flags (SYN, ACK, RST, PSH, FIN)
  - ICMP message type
- Filtering rules are based on pattern-matching

Packet Filtering Examples

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

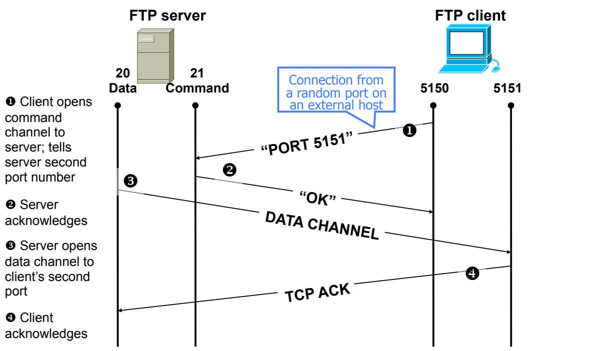
D

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]



Disadvantages of stateless packet filters: How distinguish (3) from attack?

FTP Packet Filter

The following filtering rules allow a user to FTP from any IP address to the FTP server at 172.168.10.12

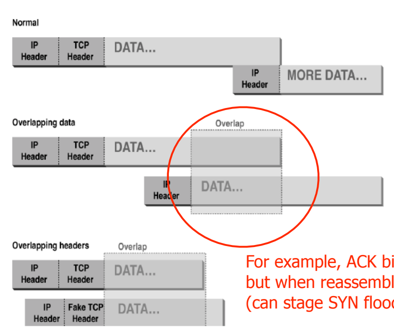
```
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 21
access-list 100 permit tcp any gt 1023 host 172.168.10.12 eq 20
 ! Allows packets from any client to the FTP control and data ports
access-list 101 permit tcp host 172.168.10.12 eq 21 any gt 1023
access-list 101 permit tcp host 172.168.10.12 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
!
```

Anything not explicitly permitted by the access list is denied!

Weaknesses of Packet Filters

- Do not prevent application-specific attacks
  - For example, if there is a buffer overflow in URL decoding routine, firewall will not block an attack string
- No user authentication mechanisms
  - … except (spoofable) address-based authentication
  - Firewalls don't have any upper-level functionality
- Vulnerable to TCP/IP attacks such as spoofing
  - Solution: list of addresses for each interface (packets with internal addresses shouldn't come from outside)
- Security breaches due to misconfiguration

Abnormal Fragmentation

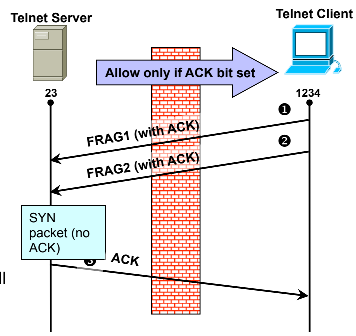


For example, ACK bit is set in both fragments, but when reassembled, SYN bit is set (can stage SYN flooding through firewall)

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



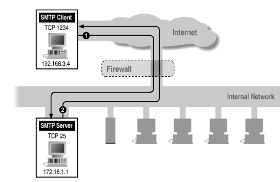
More Fragmentation Attacks

- Split ICMP message into two fragments, the assembled message is too large
  - Buffer overflow, OS crash
- Fragment a URL or FTP "put" command
  - Firewall needs to understand application-specific commands to catch this
- Denial of service (e.g., chargen attacks)
  - "Character generation" debugging tool: connect to a certain port and receive a stream of data
  - If attacker fools it into connecting to itself, CPU locks

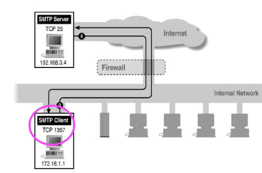
Stateless Filtering Is Not Enough

- In TCP connections, ports with numbers less than 1024 are permanently assigned to servers
  - 20,21 for FTP, 23 for telnet, 25 for SMTP, 80 for HTTP…
- Clients use ports numbered from 1024 to 16383
  - They must be available for clients to receive responses
- What should a firewall do if it sees, say, an incoming request to some client's port 5612?
  - 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: Variable Port Use



Inbound SMTP



Outbound SMTP

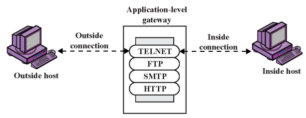
Session Filtering

- 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 incoming traffic to a high-numbered port if there is an established connection to that port
- Hard to filter stateless protocols (UDP) and ICMP
- Typical filter: deny everything that's not allowed
  - Must be careful filtering out service traffic such as ICMP
- Filters can be bypassed with IP tunneling

Example: Connection State Table

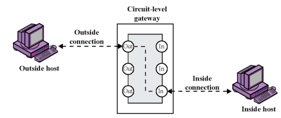
Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	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

Application-Level Gateway



- Splices and relays two application-specific connections
  - Example: Web browser proxy
  - Daemon spawns proxy process when communication is detected
  - Big processing overhead, but can log and audit all activity
- Can support high-level user-to-gateway authentication
  - Log into the proxy server with your name and password
- Simpler filtering rules than for arbitrary TCP/IP traffic
- Each application requires implementing its own proxy

Circuit-Level Gateway



- Splices two TCP connections, relays TCP segments
- Less control over data than application-level gateway
  - Does not examine the contents of TCP segment
- Client's TCP stack must be aware of the gateway
  - Client applications are often adapted to support SOCKS
- Often used when internal users are trusted
  - Application-level proxy on inbound connections, circuit-level proxy on outbound connections (lower overhead)

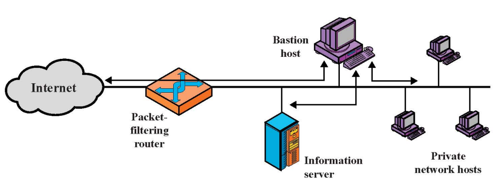
Comparison

	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

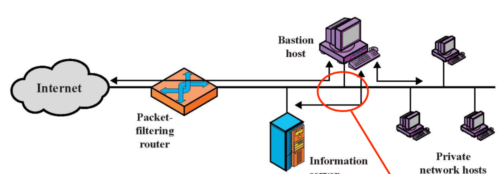
Bastion Host

- 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 (independent of others)
  - 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

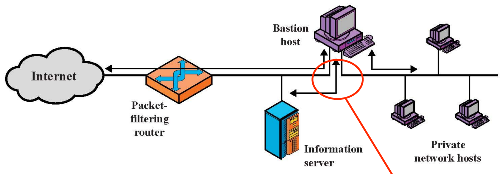


Single-Homed Bastion Host

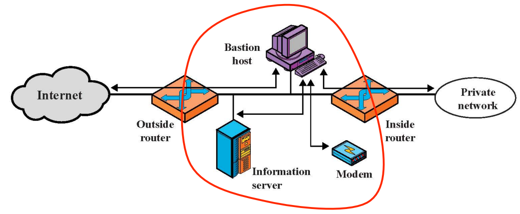


If packet filter is compromised, traffic can flow to internal network

Dual-Homed Bastion Host



Screened Subnet



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 the attacker

General Problems with Firewalls

- Interfere with networked applications
- Don't solve the 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
  - The "wireless access points" hole
- Increasing complexity and potential for misconfiguration

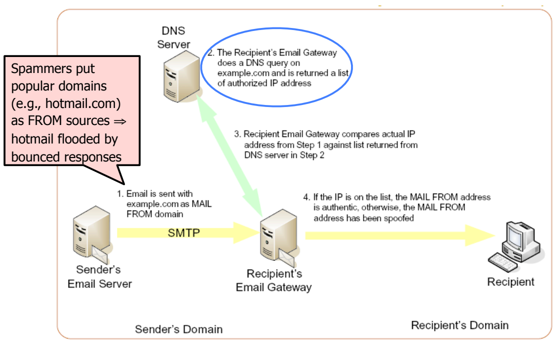
Defending Against Spam

CAN-SPAM Act (passed in 2003)

http://www.ftc.gov/spam

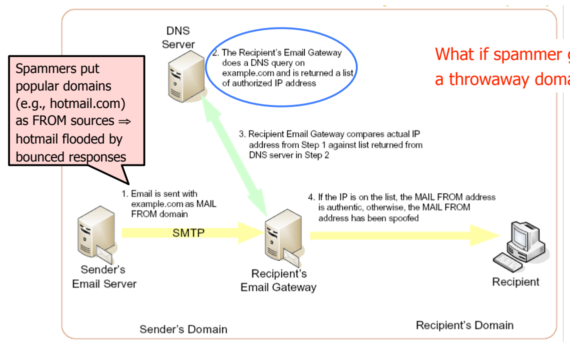
- Legal solution to the problem
  - Bans email harvesting, misleading header information, deceptive subject lines, use of proxies
  - Requires opt-out and identification of advertising
  - Imposes penalties (up to $11K per violation)
- FTC report on effectiveness (Dec 2005)
  - 50 cases pursued in the US
  - No impact on spam originating outside the US (60%)
  - Open relays hosted on botnets make it difficult to collect evidence

SPF (Sender Policy Framework)



Used by AOL and others

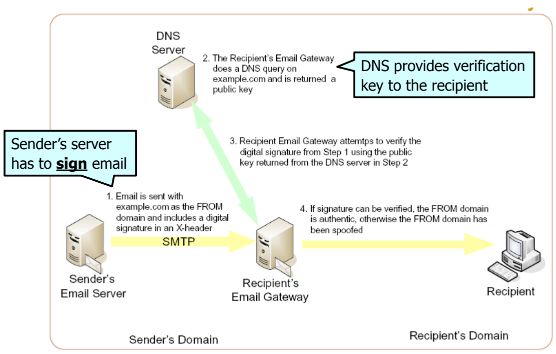
SPF (Sender Policy Framework)



What if spammer gets a throwaway domain?

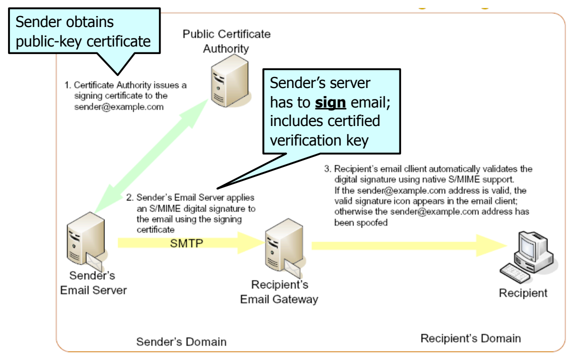
Used by AOL and others

Domain Keys (DKIM)



From Yahoo

S/MIME



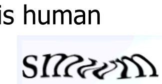
Encapsulating Policies in Email Addresses

- Email addresses encode policies
- (Using crypto)

- Idea: Give different email address to different people. Email address contained MACed encoding of policy for that address
  - Receive once
  - Receive only for a specific window in time

Puzzles and CAPTCHAs

- Generic defenses against spam and DoS
- Basic idea: sender must solve a "puzzle" before his email or connection request is accepted
  - Takes effort to solve, but solution easy to check
  - Sender has to "pay" in computation time
    - Example (Hashcash): find collision in a short hash
- CAPTCHA: prove that the sender is human
  - Solve a "reverse Turing test"
  - Only in application layer (e.g., Web)



- Both are difficult to deploy (why?)

Defending Against DDoS

DDoS Defense Techniques

- Authenticate packet sources
  - Not feasible with current IP (unless IPsec is used)
- Filter incoming traffic on access routers or rate-limit certain traffic types (ICMP and SYN packets)
  - Need to correctly measure normal rates first!
- Puzzles and CAPTCHAS: force clients to do an expensive computation or prove they are human
  - Honest clients can easily do this, but zombies can't
  - Requires modification of TCP/IP stack (not feasible?)

Finding Attack Sources

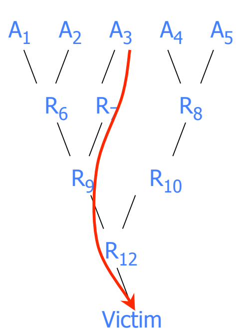
- Note: this will only locate zombies
  - Forensics on zombie machines can help find masters and the attacker who remotely controls them
- Can use existing IP routing infrastructure
  - Link testing (while attack is in progress)
  - Packet logging (for post-mortem path reconstruction)
- …or propose changes to routing infrastructure
  - IP traceback (e.g., via packet marking) and many other proposals
  - Changing routing infrastructure is hard!

Link Testing

- Only works while attack is in progress
- Input debugging
  - Victim reports attack to upstream router
  - Router installs a filter for attack traffic, determines which upstream router originated it
  - Repeat upstream (requires inter-ISP cooperation)
- Controlled flooding
  - Iteratively flood each incoming link of the router; if attack traffic decreases, this must be the guilty link
    - Use a form of DoS to throttle DoS traffic (!!)
  - Need a good network map and router cooperation

IP Traceback Problem

- How to determine the path traversed by attack packets?
- Assumptions:
  - Most routers remain uncompromised
  - Attacker sends many packets
  - Route from attacker to victim remains relatively stable

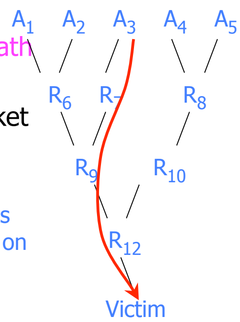


Obvious Solution Doesn't Work

- Obvious solution: have each router on the path add its IP address to packet; victim will read path from the packet
- Problem: requires space in the packet
  - Paths can be long
  - Current IP format provides no extra fields to store path information
  - Changes to packet format are not feasible

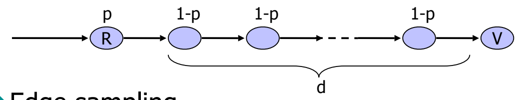
Probabilistic Packet Marking

- DDoS involves many packets on the same path
- With some probability, each router marks packet with router's address
  - Fixed space per packet
  - Large number of packets means that each router on the path will appear in some packet



Node and Edge Sampling

- Node sampling
  - With probability p, router stores its address in packet
  - Router at distance d shows up with probability $p(1-p)^d$



- Edge sampling
  - Packet stores an edge and distance since it was stored
    - More space per packet, but fewer packets to reconstruct path
  - With probability p, router stores the current edge and sets distance to 0, else increments distance by 1

Authenticated Packet Marking

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)
  - Each router shares secret key with victim: too complex

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)
  - Each router shares secret key with victim: too complex

- Time-release keys

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)
  - Each router shares secret key with victim: too complex
- Time-release keys

  - Each router has a sequence of keys

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)
  - Each router shares secret key with victim: too complex
- Time-release keys
  - Each router has a sequence of keys

  - Publishes first key in digital certificate

Authenticated Packet Marking

- Packet markings not authenticated: attacker can forge them and mislead the victim
- Digital signatures are too expensive
- Message authentication codes (MACs)
  - Each router shares secret key with victim: too complex
- Time-release keys
  - Each router has a sequence of keys
  - Publishes first key in digital certificate

  - Changes key periodically

Time-Release Key Chain

- Router creates chain of keys K_0, K_1, … ,K_{N-1}
  - Select a random key K_N
  - Using hash function, let $K_j = hash(K_{j+1})$
- Router publishes K_0 in public certificate
- Keys will be used in order K_1, K_2, …
  - Given K_j, cannot predict K_i for i>j
  - Given K_j, can compute K_0 and check if K_i is a member of a valid key chain

Secure Overlay Services (SOS)

[Keromytis et al.]

- Goal: maintain access in face of DDOS attack
- Idea: move victim site to another location on overlay network
- Separate good from bad/unknown traffic
  - Authenticate human users when their traffic enters the overlay
    - Force them to solve a CAPTCHA (reverse Turing test)
  - Route good traffic to new location through overlay

Intrusion Detection

After All Else Fails

- Intrusion prevention
  - Find buffer overflows and remove them
  - Use firewall to filter out malicious network traffic
- Intrusion detection is what you do after prevention has failed
  - Detect attack in progress
    - Network traffic patterns, suspicious system calls, etc.
  - Discover telltale system modifications

What Should Be Detected?

- Attempted and successful break-ins
- Attacks by legitimate users
  - For example, illegitimate use of root privileges
  - Unauthorized access to resources and data
- Trojan horses
- Viruses and worms
- Denial of service attacks

Where Are IDS Deployed?

- 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 vs. Anomaly

- Password file modified
- Four failed login attempts
- Failed connection attempts on 50 sequential ports
- User who usually logs in around 10am from UW dorm logs in at 4:30am from a Russian IP address
- UDP packet to port 1434
- "DEBUG" in the body of an SMTP message

Misuse vs. Anomaly

◆ Password file modified	Misuse
◆ Four failed login attempts	
◆ Failed connection attempts on 50 sequential ports	
◆ User who usually logs in around 10am from UW dorm logs in at 4:30am from a Russian IP address	
◆ UDP packet to port 1434	
◆ "DEBUG" in the body of an SMTP message	

Misuse vs. Anomaly

◆ Password file modified	Misuse
◆ Four failed login attempts	Anomaly
◆ Failed connection attempts on 50 sequential ports	
◆ User who usually logs in around 10am from UW dorm logs in at 4:30am from a Russian IP address	
◆ UDP packet to port 1434	
◆ "DEBUG" in the body of an SMTP message	

Misuse vs. Anomaly

◆ Password file modified	Misuse
◆ Four failed login attempts	Anomaly
◆ Failed connection attempts on 50 sequential ports	Anomaly
◆ User who usually logs in around 10am from UW dorm logs in at 4:30am from a Russian IP address	
◆ UDP packet to port 1434	
◆ "DEBUG" in the body of an SMTP message	

Misuse vs. Anomaly

◆ Password file modified	Misuse
◆ Four failed login attempts	Anomaly
◆ Failed connection attempts on 50 sequential ports	Anomaly
◆ User who usually logs in around 10am from UW dorm logs in at 4:30am from a Russian IP address	Anomaly
◆ UDP packet to port 1434	
◆ "DEBUG" in the body of an SMTP message	

Misuse vs. Anomaly

◆ Password file modified	Misuse
◆ Four failed login attempts	Anomaly
◆ Failed connection attempts on 50 sequential ports	Anomaly
◆ User who usually logs in around 10am from UW 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	

Misuse vs. Anomaly

Password file modified	Misuse
Four failed login attempts	Anomaly
Failed connection attempts on 50 sequential ports	Anomaly
User who usually logs in around 10am from UW 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
    - 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?

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 overflow exploits
  - Hard to handle mutations
    - Polymorphic viruses: each copy has a different body
- Big research 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

What's a "Profile?"

- Login and session activity
  - Login and location frequency; last login; password fails; session elapsed time; session output, CPU, I/O
- Command and program execution
  - Execution frequency; program CPU, I/O, other resources (watch for exhaustion); denied executions
- File access activity
  - Read/write/create/delete frequency; records read/written; failed reads, writes, creates, deletes; resource exhaustion
- How to make all this auditing scalable?

Host-Based IDS

Host-Based IDS

- Use OS auditing and monitoring mechanisms to find applications taken over by attacker
  - Log all 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

Host-Based IDS

- Use OS auditing and monitoring mechanisms to find applications taken over by attacker
  - Log all 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

Host-Based IDS

- Use OS auditing and monitoring mechanisms to find applications taken over by attacker
  - Log all 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

Host-Based IDS

- Use OS auditing and monitoring mechanisms to find applications taken over by attacker
  - Log all 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

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

Host-Based Anomaly Detection

- Compute statistics of certain system activities
- 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

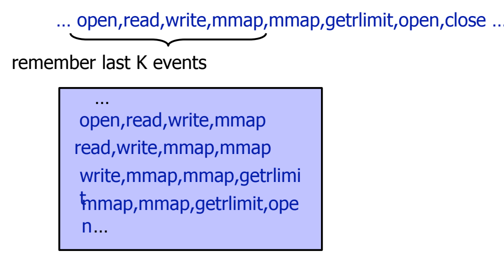
Big problem: most unpredictable user is the most important

Tripwire

- File integrity checker
  - Records hashes of critical files and binaries
    - Recorded hashes must be in read-only memory (why?)
  - Periodically checks that files have not been modified, verifies sizes, dates, permission
- Good for detecting rootkits
- Can be subverted by a clever rootkit
  - Install 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?

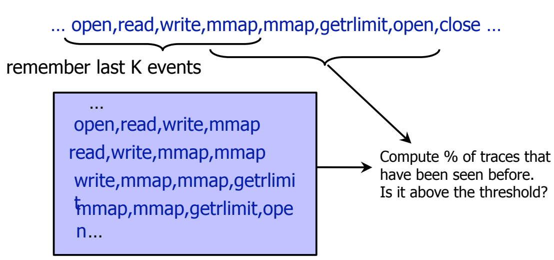
"Self-Immunology" Approach [Forrest]

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



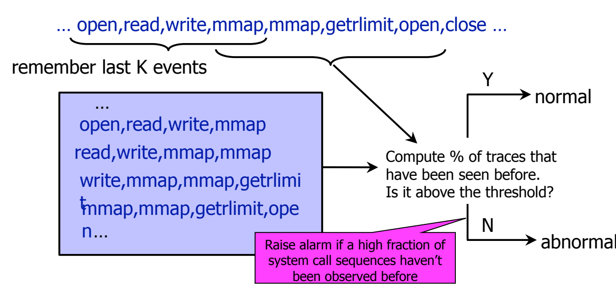
"Self-Immunology" Approach [Forrest]

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



"Self-Immunology" Approach [Forrest]

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



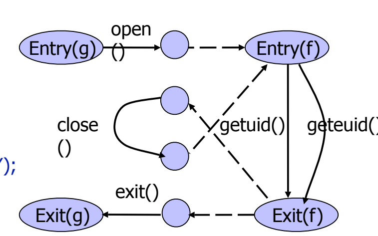
Better System Call Monitoring

[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

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)

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)

- Watch for protocol violations, unusual connection patterns, attack strings in packet payloads
  - Check packets against rule sets

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)
- Watch for protocol violations, unusual connection patterns, attack strings in packet payloads
  - Check packets against rule sets

- Con: can't inspect encrypted traffic (IPSec, VPNs)

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)
- Watch for protocol violations, unusual connection patterns, attack strings in packet payloads
  - Check packets against rule sets
- Con: can't inspect encrypted traffic (IPSec, VPNs)

- Con: not all attacks arrive from the network

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)
- Watch for protocol violations, unusual connection patterns, attack strings in packet payloads
  - Check packets against rule sets
- Con: can't inspect encrypted traffic (IPSec, VPNs)
- Con: not all attacks arrive from the network

- Con: record and process huge amount of traffic

U. of Toronto, 2004 (from David Lie)

- Date: Fri, 19 Mar 2004
- Quote from email:
  - "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' "

Popular NIDS

- Snort (popular open-source tool)
  - Large rule sets for known vulnerabilities
    - **2007-03-22:** Microsoft Windows Server Service Controller is prone to a buffer overflow vulnerability that may allow an attacker to take complete control of the target host.
    - **2007-03-08:** The HP Mercury LoadRunner agent suffers from a programming error that may allow a remote attacker to cause a stack-based buffer overflow condition to occur.
- Bro (from Vern Paxson at LBL)
  - Separates data collection and security decisions
    - Event Engine distills the packet stream into high-level events describing what's happening on the network
    - Policy Script Interpeter uses a script defining the network's security policy to decide what to do in response

Irony and NIDS

Sourcefire Snort Remote Buffer Overflow

- **Notification Type:** IBM Internet Security Systems Protection Advisory
- **Notification Date:** Feb 19, 2007
- **Description:** Snort IDS and Sourcefire Intrusion Sensor IDS/IPS are vulnerable to a stack-based buffer overflow, which can result in remote code execution.

… patched since then (phew!)

Port Scanning

- Many vulnerabilities are OS specific
  - Bugs in specific implementations
  - Oversights in 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

Scanning Defense

- Scan suppression: block traffic from addresses that previously produced too many failed connection attempts
  - Goal: detect port scans from attacker-controlled hosts
  - Requires network filtering and maintaining state
  - Can be subverted by slow scanning; does not work very well if the origin of 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

Attacking and Evading NIDS

- Overload NIDS with huge data streams, then attempt the intrusion
  - Bro solution: watchdog timer
    - Check that all packets are processed by Bro within T seconds; if not, terminate Bro, use tcpdump to log all subsequent traffic
- Use encryption to hide packet contents
- Split malicious data into multiple packets
  - NIDS does not have full TCP state and does not always understand every command of receiving application
  - Simple example: send "ROB<DEL><BS><BS>OT", receiving application may reassemble to "ROOT"