Computer Intrusion Detection

Lecture 3

Threats

Xiangyang Li

Outline

Some Terminology

Some Attack Examples

Common Attacks

Terminology of Attack

Attack: Intrusion & Misuse

Intrusion

From non-user to "user"

External

Misuse/Abuse

User abuses privileges

Insider

Attack

Attempts to exploit a vulnerability

External or internal

More Terminology

Active. Requires action on the part of the person or system to gather information.

Passive. Relies on information gathered without any action (e.g. without sending any packets).

Fingerprinting. Determining the Operating System of a machine by investigating packets/responses from the machine.

Stateful. For example, a system (intrusion detection, firewall, etc) is stateful if it retains information of the state of TCP sessions.

An Attack Model

Reconnaissance:

Determine which IP addresses are on the net.
Find out what the operating system is, what applications are running.
Find a vulnerability.

Assault:

Attack the vulnerable system.

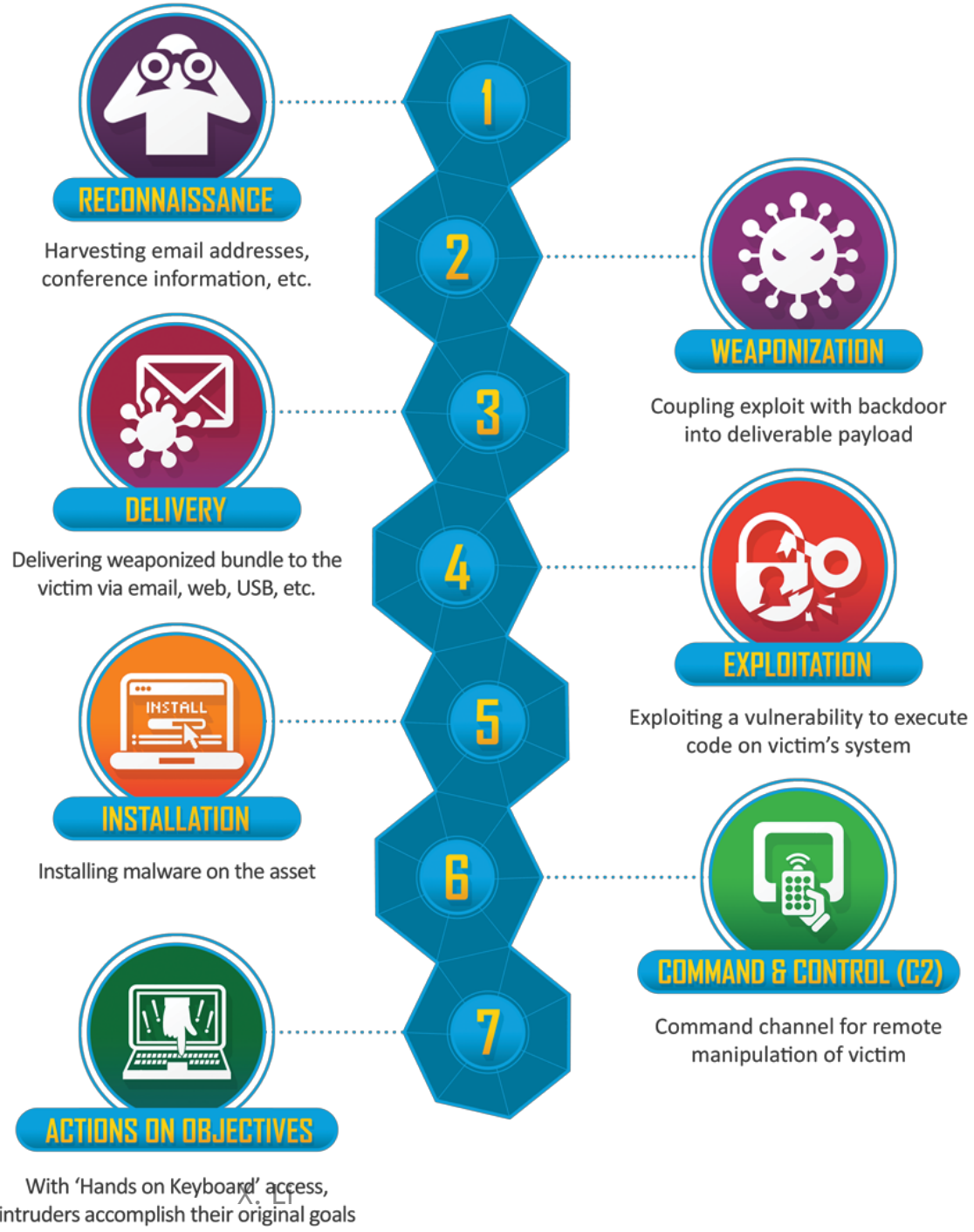
Cover tracks:

Erase logs, hide evidence of the attack.

Dig in:

Install trojans, rootkits, etc.

Cyber Kill Chain (Lockheed Martin)



Another Attack Model

- Flood the computer with requests until it goes down.
  - This may have no precursors, such as scans.
  - The attacker need not be determinable from the packets.
  - The attacker may gain nothing directly; the purpose is to cause damage.

Outline

Some Terminology

Some Attack Examples

Common Attacks

SANS Top List

- 2010 Top 25 Software Errors
  - CWE-79: Failure to Preserve Web Page Structure ('Cross-site Scripting')
  - CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
  - CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
  - CWE-352: Cross-Site Request Forgery (CSRF)
  - CWE-285: Improper Access Control (Authorization)

ITU-T X.1500 series: structured cybersecurity information exchange techniques

- X.1500 Overview of cybersecurity information exchange
- X.1520 Common vulnerabilities and exposures (CVE)
- X.1521 Common vulnerability scoring system (CVSS)
- X.1524 Common weakness enumeration (CWE)
- X.1525 Common weakness scoring system (CWSS)
- X.1526 Language for open definition of vulnerabilities and for assessment of a system state
- X.1528 Common platform enumeration (CPE)
  - X.1528.1 CPE naming /.2 CPE name matching /.3 CPE dictionary /.4 CPE applicability language
- X.1541 Incident object description exchange format
- X.1544 Common attack pattern enumeration and classification (CAPEC)
- X.1546 Malware attribute enumeration and characterization (MAEC)
- X.1570 Discovery mechanisms in the exchange of cybersecurity information
- X.1580 Real-time inter-network defence
- X.1581 Transport of real-time inter-network defence messages
- X.1582 Transport protocols supporting cybersecurity information exchange

https://www.itu.int/rec/T-REC-X/en

SYNOPSYS

Joe Jarzombek, *Software Supply Chain Management*, Presentation

IBM

What is Stuxnet?

- Worm discovered in June 2010 by AV firm VirusBlokAda
- Spreads over the network and removable storage (USB)
- Targets industrial control systems and can modify code on programmable logic controllers (PLCs) that drive industrial processes

Adopted from a IBM Presentation. The opinion expressed is solely by IBM.

IBM

Stuxnet highlights

- Uses ~8 different methods to propagate
- Includes exploits for 4 previously unpatched vulnerabilities
- Includes components signed with stolen digital certificates
- Injects itself into multiple processes, using global mutexes and an RPC server to coordinate and communicate
- Can function as part of a peer-to-peer network
- Jumps the 'air gap' by infecting USB drives and Siemens SIMATIC Step7 files
- Modifies PLC code on Siemens PCS7 system.
  (http://support.automation.siemens.com/WW/view/en/43876783)

IBM

Stuxnet's Targets

- **SCADA** – Supervisory Control and Data Acquisition
- **Siemens SIMATIC PCS7** Process Control System
- **SIMATIC WinCC** operator control and monitoring system
  - –Attacks system through hardcoded default SQL server password
- **SIMATIC Step7** engineering system
  - –Modifies PLC code blocks
  - –Hides modification of code blocks
- **SIMATIC PLCs** (programmable logic controllers)
  - –Specifically, those with 6ES7-315-2 and 6ES7-417 series CPUs
- PLC infection will not occur if the system doesn't match the designated target specification
- **http://support.automation.siemens.com/WW/view/en/43876783**

How Stuxnet Spreads

- Removable storage devices (USB flash drives)
  - `autorun.inf`
  - .LNK vulnerability, unpatched at the time of discovery
- Over the network
  - Network shares
  - Printer Spooler vulnerability – unpatched at the time of discovery
  - NetPathCanonicalize vulnerability – what Conficker/Downadup uses, fixed in 2008
  - Default password in WinCC SQL database server
- Infecting Step7 project files
  - These could spread over USB, e-mail, etc

Outline

Some Terminology

Some Attack Examples

Common Attacks

- Common Network Attacks
- Common Host Attacks
- Malicious Logic
- Covert Channels

Common Network Attack Types

- Denial of Service (DOS)
- Probes and Network Mapping
- Fingerprinting
- Gaining access and TCP Hijacking

Partially based on D. Marchette's book with materials courtesy of D. Marchette

DOS: Land Attack

- A single packet attack.
- A packet is sent with both the source and destination IP address set to the target machine.
- Locks up the machine (only works on older systems).
- Note: This, like most attacks, requires the attacker to carefully craft packets.

Table 4.1 Land attack signature.

Protocol	Specifics	Effect
TCP	SYN packet with same source and destination	Locks up system

Example:

06:49:55.47 10.10.2.23.139 > 10.10.2.23.139: S

Filter: ip[12:4] = ip[16:4]

Comment:

General filter to detect any IP packet with equal source and destination. Note that we cannot say the more natural "src host == dst host."

DOS: Ping of Death

- Another single packet attack.
- A ping (ICMP echo request) is sent to the target machine with an illegally long payload (greater than 64K).
- Locks up the machine (only works on older systems).
- Note: some versions of ping allowed one to do this.

DOS: Teardrop

- Another single packet attack.
- A UDP packet is sent fragmented, with the fragments overlapping.
- Locks up the machine (only works on older systems).

Table 4.7 Teardrop signature.

Protocol	Specifics	Effect
UDP	Fragmented packet, fragments overlap	Locks up system

Example:

07:21:33.21 172.16.123.37.23453 > 10.10.2.23.53: udp (frag 1213:350@0+)

07:21:33.21 172.16.123.37.23453 > 10.10.2.23.53: (frag 1213:300@350)

350@300

Filter: udp and (ip[6:1] & 0x20 != 0)

Comment:

This only detects fragmented UDP packets.

One must then check to see whether the fragments overlap.

DOS: UDP Storm

- Another single packet attack, in principle.
- Was the “attack-du-jour” for a while in 1999.
- A UDP packet is sent with:
  - Source IP victim1.
  - Destination IP victim2.
  - Source port 7 (echo).
  - Destination port 19 (chargen).
- Causes the two victim machines to “attack” each other.

Table 4.8 UDP storm signature.

Protocol	Specifics	Effect
UDP	Source Port: 7 Destination Port: 19	Both hosts lock up

Example:

11:23:17.42 10.10.2.34.7 > 10.10.2.37.19

Filter: udp and (src port 7) and (dst port 19)

Comment:

Other ports can be used, as long as they both respond to packets.

DOS: SYN Flood

- Send a lot of SYN packets to the target.
- These start half-open connections.
- If enough connections are started before they start to time out, the connection table can fill up.
- The machine crashes, or cannot service legitimate connections.

DOS: Process Table

- Similar to the SYN flood. Initiate a lot of TCP connections.
- Requires an application that will accept the connections.
- Each connection forks a process. If enough connections are forked the process table fills up.

DOS: Targa3

- Send a lot of malformed packets:
  - Invalid fragmentation, protocol, packet size, or IP header values;
  - Invalid options;
  - Invalid TCP segments;
  - Invalid routing flags.
- These either crash the system, or use up resources while dealing with the strange packets.

DOS: Smurf

- Send a lot of ping packets (echo requests) to an intermediary network with the victim as source IP.
- These all generate echo replies, which all get sent to the victim.
- Using broadcast (255) greatly magnifies the effect (assuming the intermediary network doesn't ignore broadcasts).

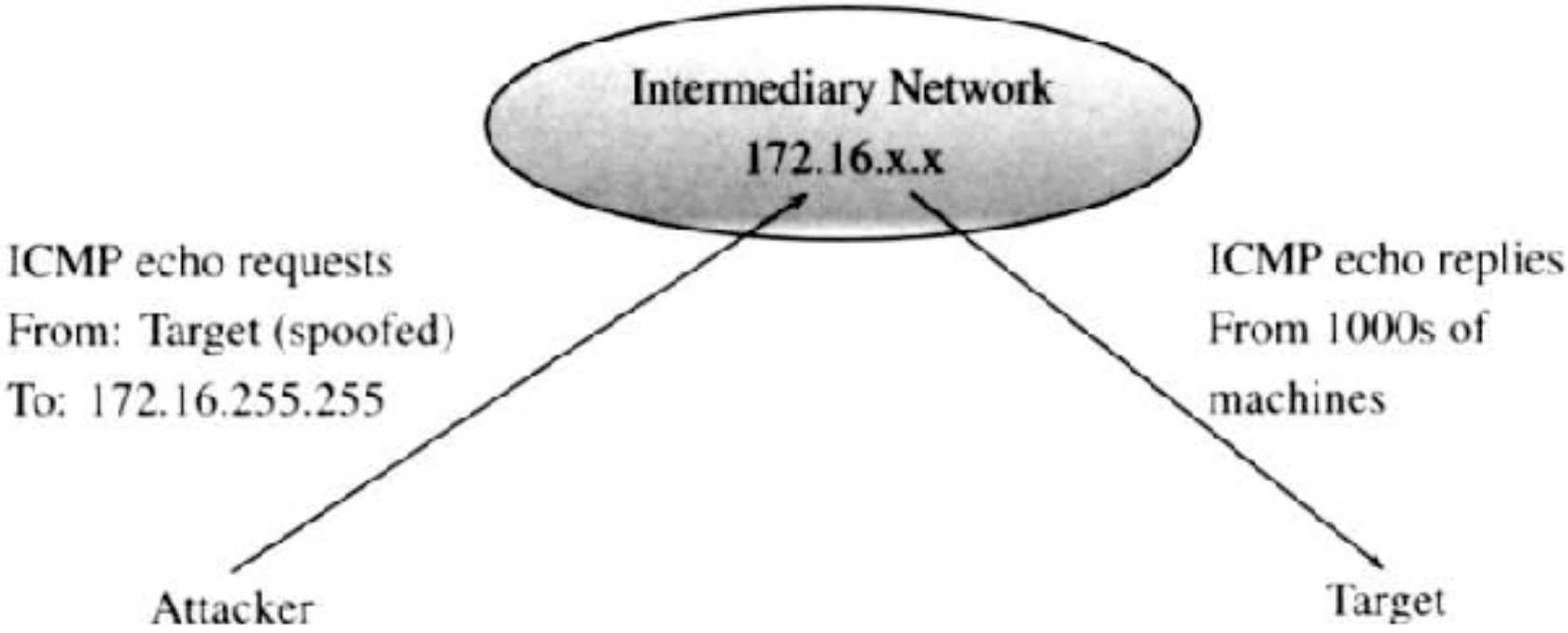


Fig. 4.1 A Smurf attack.

DOS: Syslogd

- There are innumerable attacks that utilize a bug in an application.
- Old versions of the syslogd daemon on Solaris would crash if sent a packet from a source with no DNS entry.
- Who would let a syslog packet in from outside the firewall anyway?

DoS Types

- There are two generic types of denial of service attacks:
  - Ones in which the attacker takes an active (and detectable) part.
  - Spoofing attacks, where the attacker does not show up on the packets sent to the victim.

“Old Attacks”

These take advantage of errors (bugs) in the programming of the operating systems.

This illustrates the basic idea of many attacks: find an error and exploit it.

Code reuse and poor programming habits can make some old attacks viable again.

Network Mapping

- How do you find a victim to attack?
- Scan/map the network
  - ping 10.10.255.255.
  - If this works, every machine on 10.10.xxx.xxx will respond.
  - Many organizations don't allow broadcast packets through their firewalls.
  - But there can be a problem if it works.
- You can do this the hard way.
  - ping one by one

Stealthy Network Mapping

USE PACKETS THAT PEOPLE EXPECT TO SEE, AND WILL IGNORE (E.G. ECHO REQUESTS).

USE PACKETS THAT ARE NOT NORMALLY LOGGED. (E.G. TCP SYN/ACK)

RANDOMIZING THE IPS SCANNED.

SLOW SCANS.

USING MULTIPLE SOURCE IPS.

USING DIFFERENT PACKETS FOR THE SCAN (MIXING ICMP, UDP, ETC.).

ONLY SCAN A SUBSET OF THE NETWORK.

Inverse Mapping

- Uses the Internet error mechanism to the attacker's advantage, e.g.
  - Send a TCP packet with the RESET flag set.
  - If the packet reaches its destination, the packet is dropped.
  - If the final router cannot deliver the packet, it sends back a "host unreachable" (ICMP) error.
- The packets that don't generate a response went to machines that exist.

Port Scan

- The attacker is looking for specific services.
  - Brute force: scan all 65,536 ports (UDP and TCP).
  - Be selective: scan for a few services such as ssh, telnet, ftp, web, etc.
  - Be very selective: scan for a single service.
  - Sometimes this is done by a program, such as a trojan or worm.
- Then you can simply look at the response for additional information.

telnet mycomputer.com 25

one gets an answer like

220 mycomputer.com ESMTP Sendmail 8.9.3/8.9.3; Sat, 8 Jan 2000 12:04:01 -0500

with help you get:

```
214-This is Sendmail version 8.9.3
214-Topics:
214-HELO EHLO MAIL RCPT DATA
214-RSET NOOP QUIT HELP VRFY
214-EXPN VERB ETRN DSN
214-For more info use "HELP <topic>".
214-To report bugs in the implementation send email to
214-sendmail-bugs@sendmail.org.
214-For local information send email to Postmaster at
214-    your site.
214 End of HELP info
```

Table 4.9 An example "bad ports" list.

Port Number	Protocol	Service	Comment
<20	TCP,UDP		Low-numbered ports
23	TCP,UDP	Telnet	
25	TCP,UDP	SMTP Email	Many vulnerabilities
53	TCP	DNS	Zone transfer
79	TCP,UDP	Finger	Vulnerabilities
111	TCP,UDP	sunrpc	Vulnerabilities
143	TCP,UDP	IMAP	Vulnerabilities
666	TCP,UDP	Doom	Networked game

Active Fingerprinting

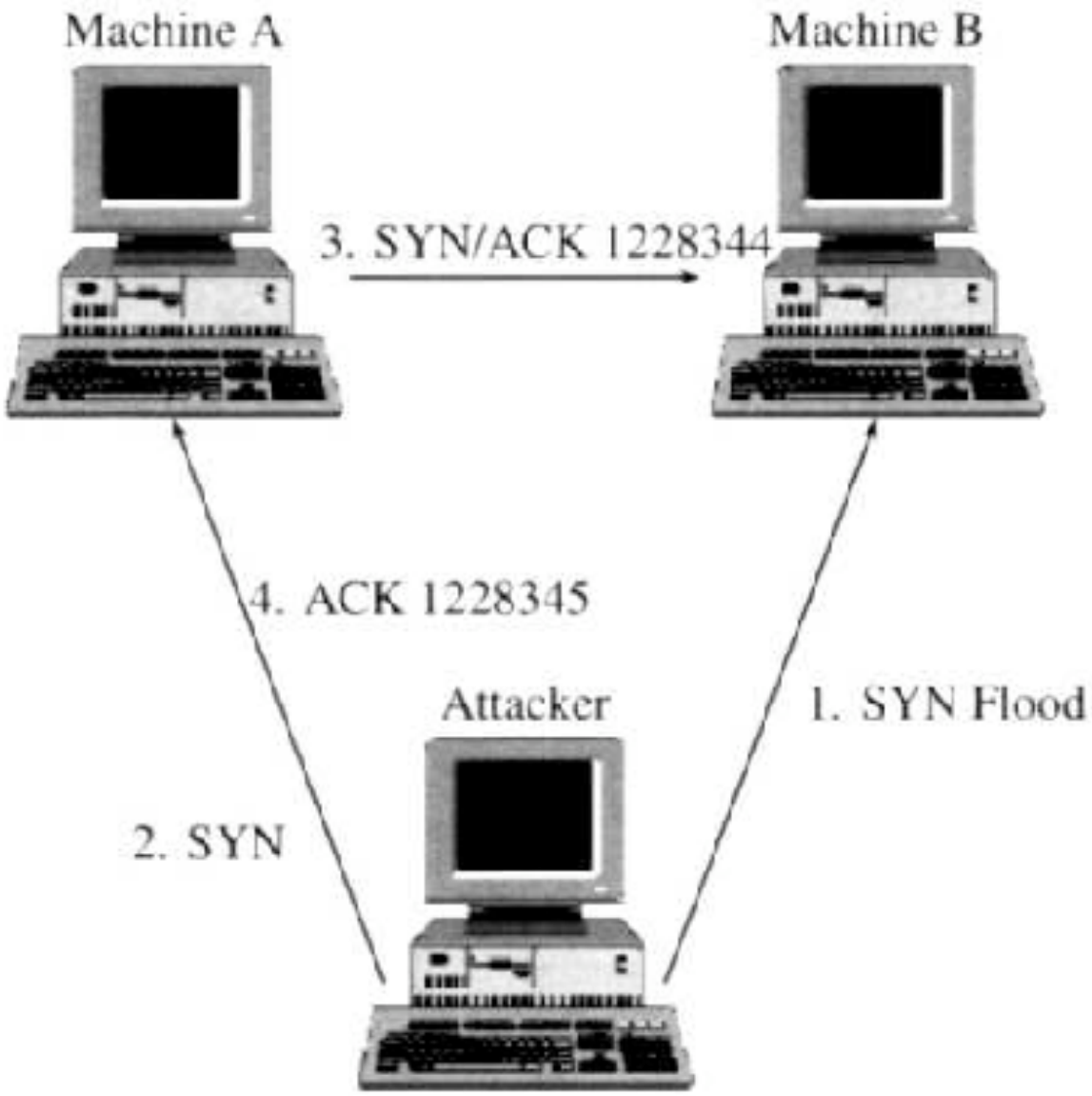
- Implementations are free to make choices about how to react to certain types of packets (e.g. improperly formed ones).
- The idea is to send a series of crafted packets, and see what the response is.
- Based on the responses to the packets, a determination can be made as to the operating system of the target.

Passive Fingerprinting

- Operating systems also make choices when they send packets out.
  - Initial time-to-live value.
  - Source port range.
  - Sequence number generation algorithm.
  - Options.
- A statistical problem.
  - range of ports
  - TTL

TCP Hijacking

- Made famous by Kevin Mitnick.
- Uses TCP sessions to allow an attacker to take over a session between two computers.
- Relies on sequence number algorithms that are easy to predict. For example: use the next number in the sequence, or add a constant to the previous sequence number.


Machine A
Machine B
3. SYN/ACK 1228344
4. ACK 1228345
Attacker
1. SYN Flood
2. SYN

What Happen?

- Setup: Machine A and machine B have a trust relationship, allowing machine B to log into machine A without requiring a password.
- Attacker must first determine the sequence number algorithm machine A uses.
- This can be accomplished by sending a bunch of SYN packets to A and see what the responding sequence numbers look like.
- Attacker SYN floods machine B to make sure it doesn't respond to anything from A.
- Attacker sends a SYN packet to A spoofed to appear to be from B.
- A responds to B with a SYN/ACK.
- Attacker then sends an ACK packet with the correct (inferred) acknowledgment number. A thinks this is part of the session and proceeds as if it were talking to B.

Stop Hijacking

DON'T BE TRUSTING.

USE A HARD TO PREDICT SEQUENCE NUMBER GENERATOR.

BLOCK ALL ACCESS TO ATTACKERS AT THE FIREWALL (IF IT SYN FLOODS, BLOCK IT).

Common Host Attack Types

- Denial of Service
- Remote to User
- User to Root
- Rootkits

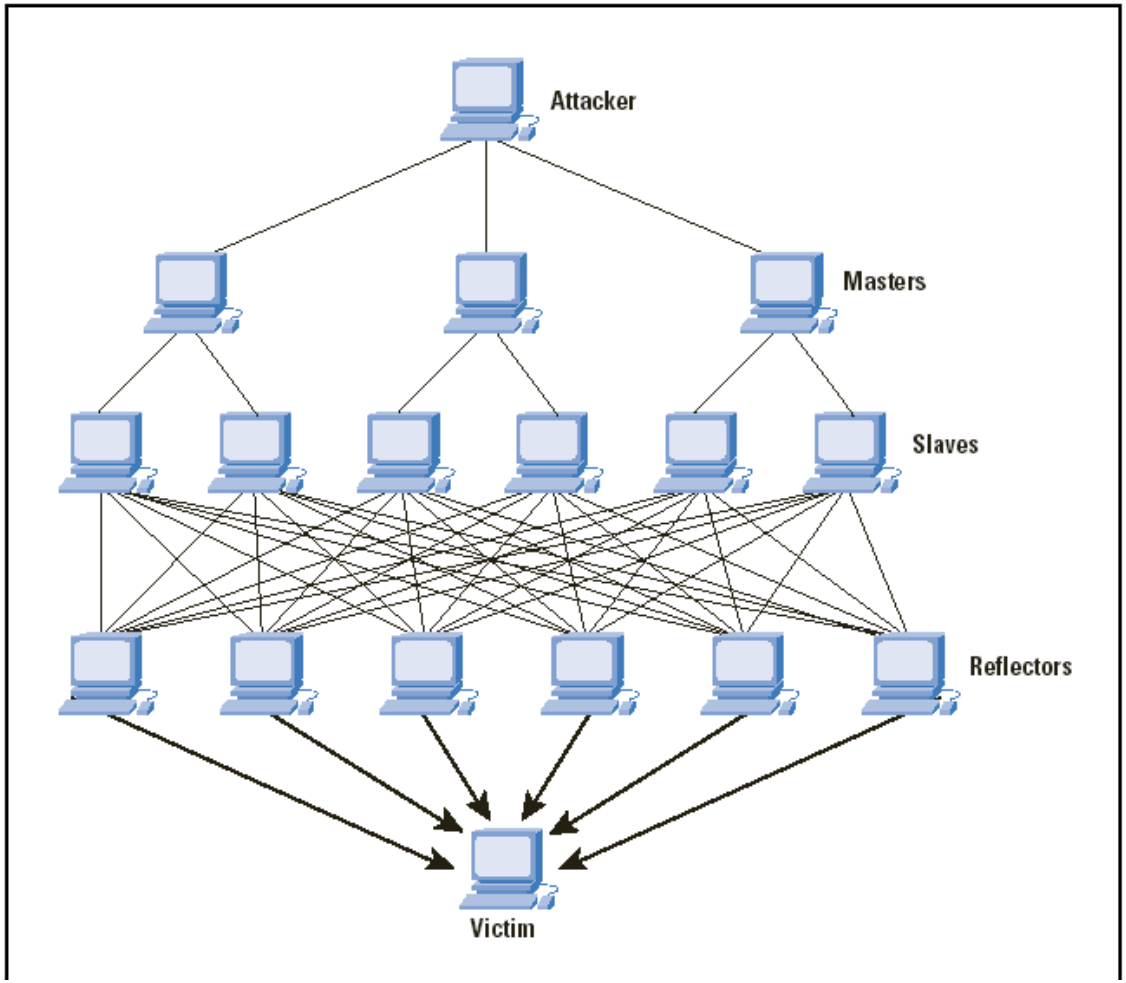
Host vs Network Attacks

- Host attacks can come over the network.
- If the attack can be discovered by looking at headers alone, it is a network attack.
- If it requires investigation of the packet content or of system logs, then it is a host attack.

Denial of Service

- Denial of service comes in three types:
  - Denial of use of a machine.
  - Denial of use of an application.
  - Denial of use of data.
- Distributed DoS can be very sophisticated:
  - Distributed Reflector DoS (DRDoS)

Figure 5: A DRDoS Attack



From Cisco's Internet Protocol Journal, 4(7)

Web Server Attacks

- Attack web servers by sending too many requests, strange requests, or using cgi-scripts that should have been disabled.
  - A typical such attack is sending a request consisting of several thousand '/'s.
  - Another is the distributed denial of service attacks, that send huge numbers of (bogus) requests, bogging down (or bringing down) the server.
  - rashIIS: Microsoft Windows NT IIS Web server can crash by a malformed GET request.

Mailbomb

- The idea is to send thousands of emails.
- The emails fill up the disk, and cause a huge waste of time deleting them.
- These are often sent via an automatic process.
- Note that this is different than spam, in the sense that there is no particular desire to have the mail read, responded to, or even necessarily received.

DoSNuke

- DoSNuke is a Denial of Service attack that sends Out Of Band data (MSG_OOB) to port 139 (NetBIOS), crashing the NT victim (bluescreens the machine).
- The packets being sent by the attacking machines are flagged "urg" because of the MSG_OOB flag. As a result, the target is weighed down.

Resource Hogging

```
#!/bin/csh
cd /tmp
while(true)
mkdir foo
cd foo
cp -r ~/* .
end
```

- Resource hogs are programs that use up the resources of the machine. Resource hogs can fill up disk space, memory, or CPU cycles.
- They require access to the machine to execute the programs, and can be defeated, to some degree, by quotas.

Malicious Users

- Compare the previous script with:
  - rm -rf *
  - If one is root, replace "*" with "/".
- Other forms of misuse
  - renaming system utilities (mv /bin/ls /bin/dir),
  - filling /tmp with junk,
  - running programs that use up all of the available memory,
  - removing user's accounts
- When any of this is done by (otherwise) authorized users, this is referred to as the "insider threat" or an insider attack.

Remote to User

- "Remote to User" refers to an attacker gaining access to a computer from the outside, obtaining a user account.
- There are several methods for doing this.
  - Social Engineering: This is to simply steal an account. This can be done by looking over someone's shoulder while they are typing, or tricking them into giving you their password.
  - The simplest way to gain access is through an account with no password. For example, many early Unix systems were shipped with an lp account that required no password.

    /etc/passwd contains: lp::9:9:Print Spooler Owner:/var/spool/lp:/bin/sh

Account Guessing

- Knowing something about people can help you guess their passwords.
- One way to get this information is through “dumpster diving”.
- Another is social engineering: make friends with them.
  - Example: kids/spouse/pets names, birthdays, social security number.
  - Sometimes just trying a few obvious things will work.
- How to detect and defend against this?

Cracking Password

- Steal the password file and run crack on it.
  - Some cgi scripts will let you do this through a web server, e.g. phf file to provide password file.
  - Any password can be cracked, given time.
  - Sufficiently good passwords can make this impractical

Farmer and Venema's Study in 1993

- They tried to obtain password files from 656 hosts.
- They succeeded for 24 of them.
- They tried to crack the passwords.
- They got:
  - 5 root passwords.
  - 259 passwords total.
  - Access to 80% (19) of the 24 machines.
- Password crackers are better now.

Defense

- Use strong passwords.
- Shadow the passwords (only root has access to the actual passwords).
- Don’t store the passwords (one-way hash).
- Use encrypted channels to access remotely.
- Check your cgi-scripts. Don’t use any unless you know what they do.
- Trust no one.

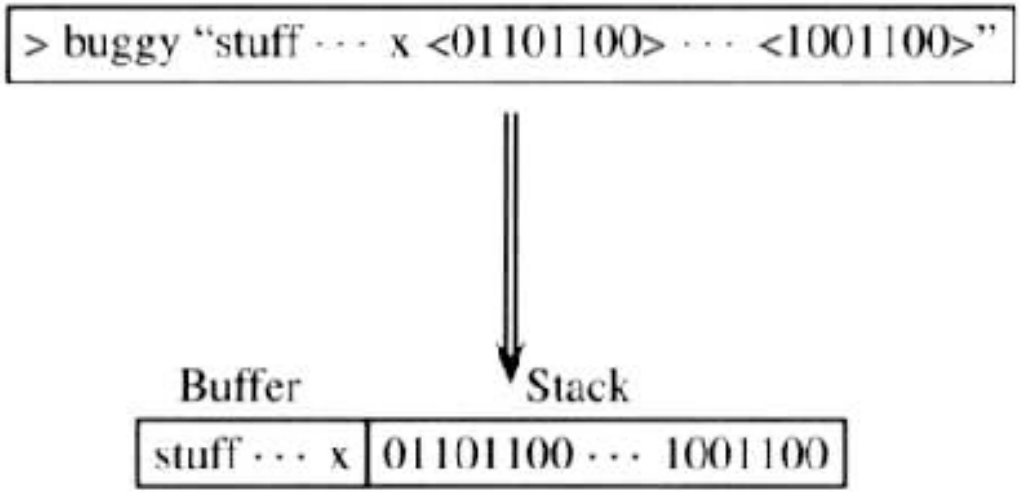
FTP Write

- Writing a machine name in a .rhosts file makes that machine trusted for rlogin.
- Writing "++" means you trust everyone.
- Some misconfigured FTP servers allow one to write in the home directory.
- Other methods are possible, such as sending an email attachment called ".rhosts" and hope the person reading the message is really naive.

Buffer Overflows

- This is the most common method for gaining user access remotely.
- It relies on a program with a bug, that allows stuff to be put on the execution stack.
  - stack based (function)
  - heap based (dynamic data)
- This happens through writing more bytes into a buffer than the buffer can hold.

finger daemon that uses gets() function



Defense

Don’t write code that doesn’t check for out-of-bounds or use “gets” and similar library routines.

Keep patches up to date.

There is a system that puts special code that monitors for inappropriate changes of permission on the stack to avoid buffer overflows.

Other Remote to User Attacks

VULNERABILITIES IN AUTHENTICATION/ AUTHORIZATION – NETWORK PRINTING FACILITY OF SILICON GRAPHICS

TROJANS – SCREEN SAVER, XLOCK, ETC.

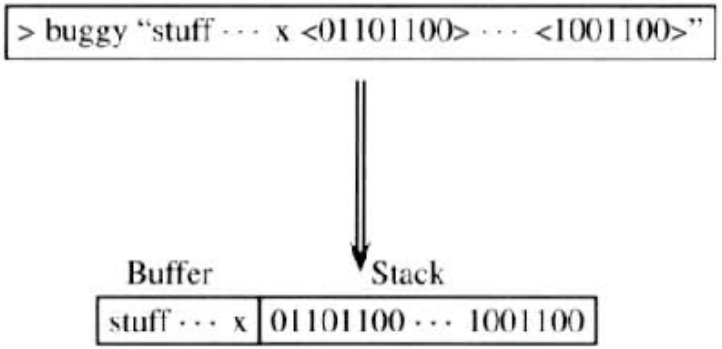
User to Root

- The next goal is to upgrade the privilege of the access.
- This means getting root permission, having an existing account.
- Some can be done through password guessing, social engineering, etc.
- Some can succeed by exploiting the special features of operating systems.

Buffer Overflow

- Many programs run as root, and if the attacker can get one of them to overflow, root access can be obtained.
- A host-based ID system can try to detect these attacks via three basic methods:
  - Check for very long arguments passed to programs.
  - Check for unusual program behavior.
  - Watch for root access that was not the result of a legal method.

The same attack, but this time the “buggy” program has root access.



Race Conditions

- The idea of this attack is to trick the system into letting you write to a root file, by switching files on it in mid-open.
- For example:
  - 1. Create a temporary file.
  - 2. Open the temporary file to write.
  - 3. Between the permission check and the open, switch the file.
  - 4. Write to the file.
- This is called a time-of-check-to-time-of-use flaw.

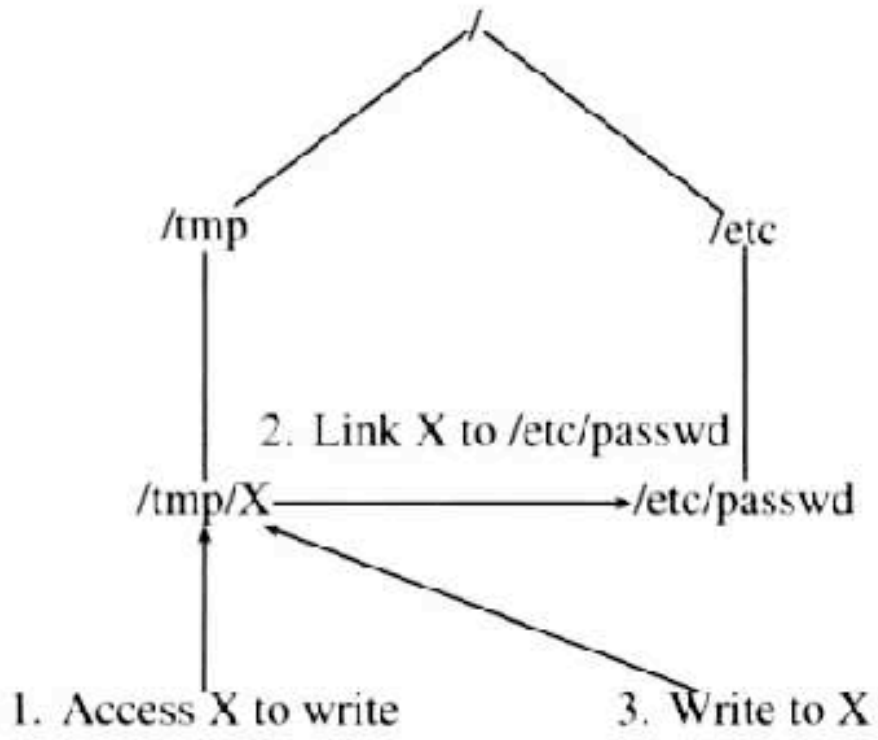


Fig. 5.2 An example of a race condition allowing write access to a password file. After opening the file /tmp/X for writing. but before writing any data, the file is deleted and replaced with a link to /etc/passwd. Subsequent writes then end up in this file.

Defense and Detection

VULNERABILITY ANALYSIS OF THE SOURCE CODE

A DYNAMIC APPROACH THAT WATCHES THE RUN-TIME ENVIRONMENT FOR POTENTIAL TOCT-TOU FLAWS

A FILE INTEGRITY CHECKER (E.G. TRIPWIRE) AND TOGETHER WITH A MONITOR SUCH AS LSOF TO SEE WHO IS OPENING VARIOUS FILES

Cover Up Tracks

- The next thing to do is cover up.
- Erase any evidence from the log files. Root permission is needed for this.
- Hide any files transferred in. (Look for files or directories named "...").
- Change "ls" and "ps" so that they won't report any suspicious files or programs. (ie: install trojans for these).
- Change "tripwire" or other security programs so that they won't report any changes.

Prepare for Revisit

- Fix the vulnerability that was used so no other attacker can use it and mess things up.
- Add a trojan that allows access. This may be a new program that opens a port, or a re-coding of a familiar one such as telnet, rlogin, ssh, etc.
- “rootkit” – A collection of programs that hides the activities of an attacker. This may include programs designed to give the attacker root permission on the machine, change log files to eliminate evidence of the attacks, and install trojan copies of system programs.

Defense and Detection

Keep security programs (such as tripwire) on removable medium and check regularly.

Watch for unusual network activity and new open ports (this is how host-based and network-based ID can work together).

Use multiple security programs to make it harder for the attacker to disable them all.

Consider using a remote log server.

Malicious Logic

- Defining malicious logic
  - Set of instructions that cause site security policy to be violated
- Types
  - Trojan horses
  - Computer viruses and worms
  - Other types
- Defenses
  - Properties of malicious logic
  - Trust

Based on materials from Matt Bishop: Computer Security: Art and Science

Example

- Shell script on a UNIX system:

```
cp /bin/sh /tmp/.xyzzy
chmod u+s,o+x /tmp/.xyzzy
rm ./ls
ls $*
```

- Place in program called “ls” and trick someone into executing it
- You now have a setuid-to-*them* shell!

Trojan Horse

Program with an *overt* purpose (known to user) and a *covert* purpose (unknown to user)

- Often called a Trojan
- Named by Dan Edwards in Anderson Report

Example: previous script is Trojan horse

- Overt purpose: list files in directory
- Covert purpose: create setuid shell

Example: NetBus

- Designed for Windows NT system
- Victim uploads and installs this
  - Usually disguised as a game program, or in one
- Acts as a server, accepting and executing commands for remote administrator
  - This includes intercepting keystrokes and mouse motions and sending them to attacker
  - Also allows attacker to upload, download files

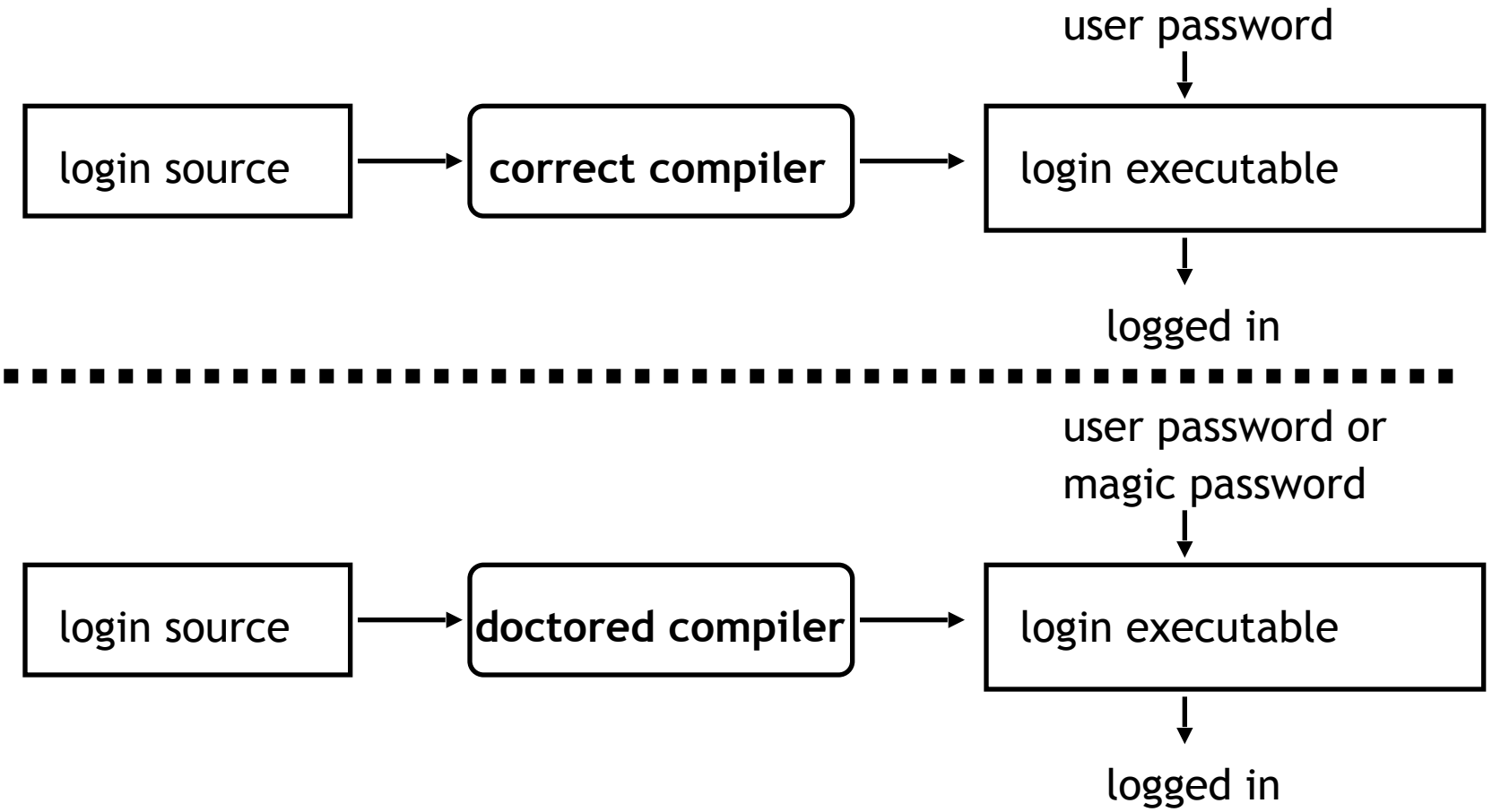
Replicating Trojan Horse

- Trojan horse that makes copies of itself
  - Also called *propagating Trojan horse*
  - Early version of *animal* game used this to delete copies of itself
- Hard to detect
  - 1976: Karger and Schell suggested modifying compiler to include Trojan horse that copied itself into specific programs including later version of the compiler
  - 1980s: Thompson implements this

Thompson's Compiler

- Modify the compiler so that when it compiles *login* , *login* accepts the user's correct password or a fixed password (the same one for all users)
- Then modify the compiler again, so when it compiles a new version of the compiler, the extra code to do the first step is automatically inserted
- Recompile the compiler
- Delete the source containing the modification and put the undoctored source back

The Login Program



Computer Virus

- Program that inserts itself into one or more files and performs some action
  - *Insertion phase* is inserting itself into file
  - *Execution phase* is performing some (possibly null) action
- Insertion phase *must* be present
  - Need not always be executed
  - Lehigh virus inserted itself into boot file only if boot file not infected

Trojan Horse Or Not?

- Yes
  - Overt action = infected program's actions
  - Covert action = virus' actions (infect, execute)
- No
  - Overt purpose = virus' actions (infect, execute)
  - Covert purpose = none
- Semantic, philosophical differences
  - Defenses against Trojan horse also inhibit computer viruses

History

Programmers for Apple II wrote some

Not called viruses; very experimental

Fred Cohen

Graduate student who described them

Teacher (Adleman) named it “computer virus”

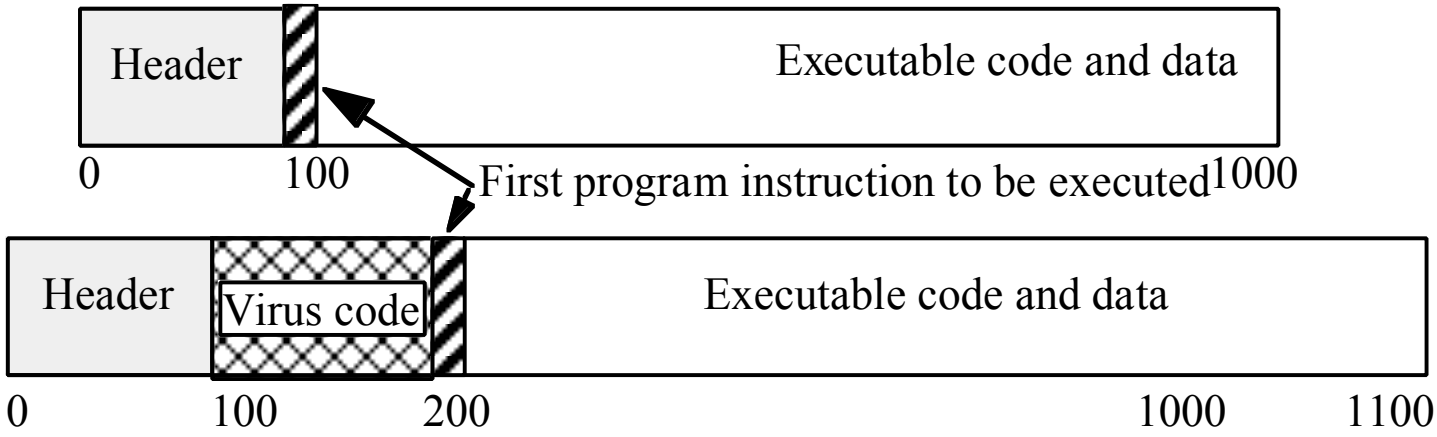
Tested idea on UNIX systems and UNIVAC 1108 system

Types of Viruses

- Boot sector infectors
- Executable infectors
- Multipartite viruses
- TSR viruses
- Stealth viruses
- Encrypted viruses
- Polymorphic viruses
- Macro viruses

Boot Sector Infectors

- A virus that inserts itself into the boot sector of a disk
  - Section of disk containing code
  - Executed when system first "sees" the disk
    - Including at boot time …
- Example: Brain virus
  - Moves disk interrupt vector from 13H to 6DH
  - Sets new interrupt vector to invoke Brain virus
  - When new floppy seen, check for 1234H at location 4
    - If not there, copies itself onto disk after saving original boot block



Executable Infectors

- A virus that infects executable programs
  - Can infect either .EXE or .COM on PCs
  - May prepend itself (as shown) or put itself anywhere, fixing up binary so it is executed at some point

Multipartite Viruses

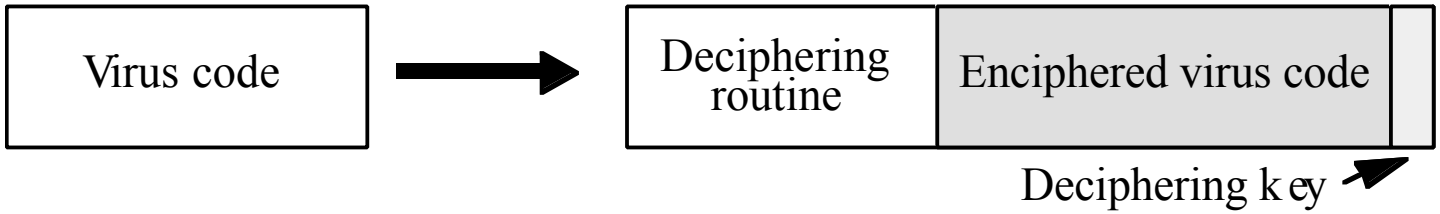
- A virus that can infect either boot sectors or executables
- Typically, two parts
  - One part boot sector infector
  - Other part executable infector

TSR Viruses

- A virus that stays active in memory after the application (or bootstrapping, or disk mounting) is completed
  - TSR is “Terminate and Stay Resident”
- Examples: Brain, Jerusalem viruses
  - Stay in memory after program or disk mount is completed

Stealth Viruses

- A virus that conceals infection of files
- Example: IDF virus modifies DOS service interrupt handler as follows:
  - Request for file length: return length of *uninfected* file
  - Request to open file: temporarily disinfect file, and reinfect on closing
  - Request to load file for execution: load infected file



Encrypted Viruses

- A virus that is enciphered except for a small deciphering routine
  - Detecting virus by signature now much harder as most of virus is enciphered

Polymorphic Viruses

- A virus that changes its form each time it inserts itself into another program
- Idea is to prevent signature detection by changing the "signature" or instructions used for deciphering routine
- At instruction level: substitute instructions
- At algorithm level: different algorithms to achieve the same purpose
- Toolkits to make these exist (Mutation Engine, Trident Polymorphic Engine)

Example

- These are different instructions (with different bit patterns) but have the same effect:
  - add 0 to register
  - subtract 0 from register
  - xor 0 with register
  - no-op
- Polymorphic virus would pick randomly from among these instructions

Macro Viruses

- A virus composed of a sequence of instructions that are interpreted rather than executed directly
- Can infect either executables (Duff's shell virus) or data files (Highland's Lotus 1-2-3 spreadsheet virus)
- Independent of machine architecture
  - But their effects may be machine dependent

Example

- Melissa
  - Infected Microsoft Word 97 and Word 98 documents
    - Windows and Macintosh systems
  - Invoked when program opens infected file
  - Installs itself as “open” macro and copies itself into Normal template
    - This way, infects any files that are opened in future
  - Invokes mail program, sends itself to everyone in user’s address book

Computer Worms

- A program that copies itself from one computer to another
- Origins: distributed computations
  - Schoch and Hupp: animations, broadcast messages
  - Segment: part of program copied onto workstation
  - Segment processes data, communicates with worm's controller
  - Any activity on workstation caused segment to shut down

Example: Internet Worm of 1988

- Targeted Berkeley, Sun UNIX systems
  - Used virus-like attack to inject instructions into running program and run them
  - To recover, had to disconnect system from Internet and reboot
  - To prevent re-infection, several critical programs had to be patched, recompiled, and reinstalled
- Analysts had to disassemble it to uncover function
- Disabled several thousand systems in 6 or so hours

Rabbits, Bacteria

- A program that absorbs all of some class of resources
- Example: for UNIX system, shell commands:

```
while true
do
    mkdir x
    chdir x
done
```

- Exhausts either disk space or file allocation table (inode) space

Logic Bombs

- A program that performs an action that violates the site security policy when some external event occurs
- Example: program that deletes company's payroll records when one particular record is deleted
  - The "particular record" is usually that of the person writing the logic bomb
  - Idea is if (when) he or she is fired, and the payroll record deleted, the company loses *all* those records

Defenses against Malicious Logic

DISTINGUISH BETWEEN DATA, INSTRUCTIONS

LIMIT OBJECTS ACCESSIBLE TO PROCESSES

INHIBIT SHARING

DETECT ALTERING OF FILES

DETECT ACTIONS BEYOND SPECIFICATIONS

ANALYZE STATISTICAL CHARACTERISTICS

Data vs. Instructions

- Malicious logic is both
  - Virus: written to program (data); then executes (instructions)
- Approach: treat "data" and "instructions" as separate types, and require certifying authority to approve conversion
  - Keys are assumption that certifying authority will *not* make mistakes and assumption that tools, supporting infrastructure used in certifying process are not corrupt

Example: LOCK

- Logical Coprocessor Kernel
  - Designed to be certified at TCSEC A1 level
- Compiled programs are type “data”
  - Sequence of specific, auditable events required to change type to “executable”
- Cannot modify “executable” objects
  - So viruses can’t insert themselves into programs (no infection phase)

Limiting Accessibility

Basis: a user (unknowingly) executes malicious logic, which then executes with all that user's privileges

Limiting accessibility of objects should limit spread of malicious logic and effects of its actions

Approach draws on mechanisms for confinement

Reducing Protection Domain

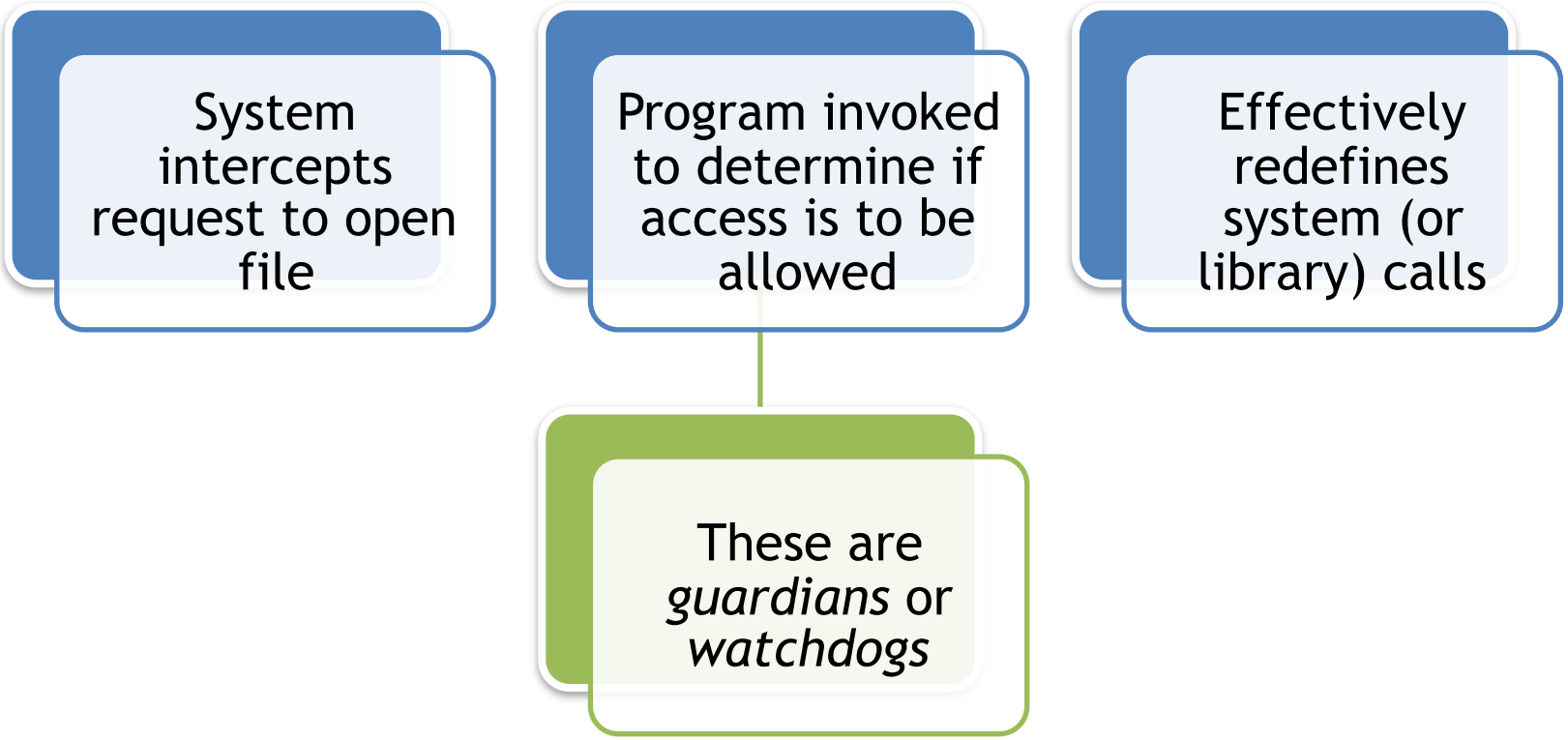
Application of principle of least privilege

Basic idea: remove rights from process so it can only perform its function

Warning: if that function requires it to write, it can write anything

But you can make sure it writes only to those objects you expect

Guardians, Watchdogs



Sandboxing

- Sandboxes, virtual machines also restrict rights
  - Modify program by inserting instructions to cause traps when violation of policy
  - Replace dynamic load libraries with instrumented routines

Example: Race Conditions

- Occur when successive system calls operate on object
  - Both calls identify object by name
  - Rebind name to different object between calls
- Sandbox: instrument calls:
  - Unique identifier (inode) saved on first call
  - On second call, inode of named file compared to that of first call
    - If they differ, potential attack underway …

Inhibit Sharing

- Use separation implicit in integrity policies
- Example: LOCK keeps single copy of shared procedure in memory
  - Master directory associates unique owner with each procedure, and with each user a list of other users the first trusts
  - Before executing any procedure, system checks that user executing procedure trusts procedure owner

Multilevel Policies

- Put programs at the lowest security level, all subjects at higher levels
  - By *-property, nothing can write to those programs
  - By ss-property, anything can read (and execute) those programs
- Example: DG/UX system
  - All executables in “virus protection region” below user and administrative regions

Detect Alteration of Files

- Compute manipulation detection code (MDC) to generate signature block for each file, and save it
- Later, recompute MDC and compare to stored MDC
  - If different, file has changed
- Example: tripwire
  - Signature consists of file attributes, cryptographic checksums chosen from among MD4, MD5, HAVAL, SHS, CRC-16, CRC-32, etc.)

Assumptions

- Files do not contain malicious logic when original signature block generated
- Pozzo & Grey: implement Biba's model on LOCUS to make assumption explicit
  - Credibility ratings assign trustworthiness numbers from 0 (untrusted) to *n* (signed, fully trusted)
  - Subjects have risk levels
    - Subjects can execute programs with credibility ratings ≥ risk level
    - If credibility rating < risk level, must use special command to run program

Antivirus Programs

- Look for specific sequences of bytes (called "virus signature" in file
  - If found, warn user and/or disinfect file
- Each agent must look for known set of viruses
- Cannot deal with viruses not yet analyzed
  - Due in part to undecidability of whether a generic program is a virus

N-Version Programming

- Implement several different versions of algorithm
- Run them concurrently
  - Check intermediate results periodically
  - If disagreement, majority wins
- Assumptions
  - Majority of programs not infected
  - Underlying operating system secure
  - Different algorithms with enough equal intermediate results may be infeasible
    - Especially for malicious logic, where you would check file accesses

Detecting Statistical Changes

- Example: application had 3 programmers working on it, but statistical analysis shows code from a fourth person—may be from a Trojan horse or virus!
- Other attributes: more conditionals than in original; look for identical sequences of bytes not common to any library routine; increases in file size, frequency of writing to executables, etc.
  - Denning: use intrusion detection system to detect these

Covert Channels

- Storage channel
  - Uses attribute of shared resource
- Timing channel
  - Uses temporal/ordering relationship of access to shared resource
- Noise in covert channel
  - Noiseless: Resource only available to sender/receiver
  - Noisy: Other subjects can affect resource

Covert Channel Examples

- Process A cannot communicate with B. They share the same file system. A creates a file named *send*. B deletes *send*. Then A transmits a bit by creating a file named either *0bit* or *1bit*. B receives this by deleting it. This continues until A creates a file called *end*.
- In KVM/370 system, the sending VM could relinquish the CPU immediately to indicate "0", and use its full quantum otherwise. The second VM could deduce whether the first sends a "1" or "0" by how quickly it gets the CPU.

CSC in TCP Header(Flags)

- 6-bit field flags and 64 combinations[1]
- 35 invalid combinations used to hide information

Valid TCP flag combination (29 types in total)

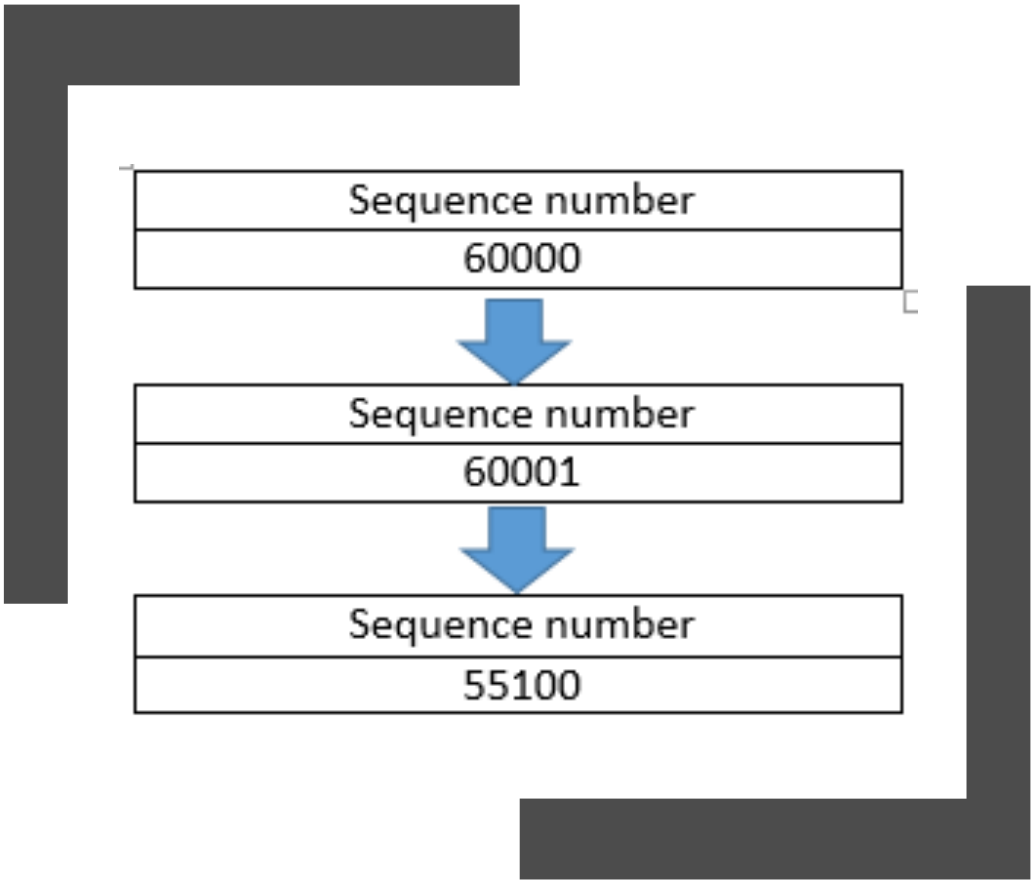
URG	ACK	PSH	RST	SYN	FIN
0	0	0	0	1	0

URG	ACK	PSH	RST	SYN	FIN
1	1	1	1	1	1

Invalid TCP flag combination (35 types in total)

(Song, MSSI Capstone/2014)

CSC in TCP Header(Sequence No.)



- Sequence should change in some order (increment by 1 each time in a specific session)
- CSC can use abnormal sequence changes in multiple packets to convey secret information[2].

A CTC Channel

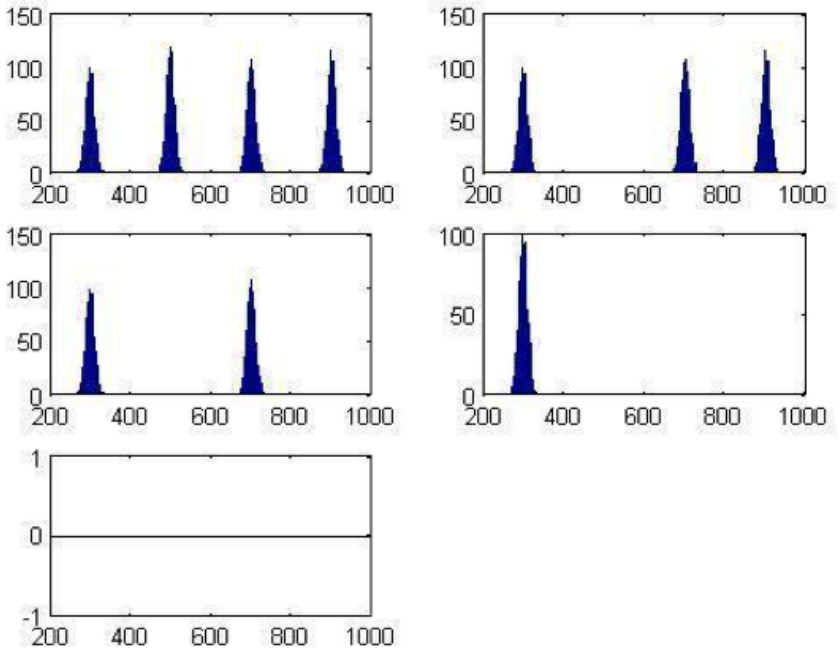
1

2

3

4

Covert timing channels use packet inter-arrival times, not header or payload embedded information, to encode covert messages.



EN.600.654 Computer Intrusion Detection, Term Project

(Wang, Song, Chen, 650.654/2014 project)

Modeling Covert Channels

Noninterference

Bell-LaPadula approach

All shared resources modeled as subjects/objects

Information Flow analysis

Again model all shared resources

Covert Channel Mitigation

- Can covert channels be eliminated?
  - Eliminate shared resource?
- *Severely* limit flexibility in using resource
  - Otherwise we get the halting problem
  - Example: Assign fixed time for use of resource
    - *Closes timing channel*
- Not always realistic
  - *Do we really need to close every channel?*

Covert Channel Analysis

- Solution: Accept covert channel
  - But analyze the capacity
    - *How many bits/second can be "leaked"*
- Allows cost/benefit tradeoff
  - Risk exists
  - Limits known
- Example: Assume data time-critical
  - Ship location classified until next commercial satellite flies overhead