

Lecture 22, May 20, 2026

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

Adware

- Trojan horse that gathers information for marketing purposes and displays advertisements
 - Often selects ads to display based on gathered information
- Believed to have originated with a company announcing it would make its software available for free, because it would pop up a window advertising company
 - Benign as user had to opt in
 - Spread through distribution of program only

Types of Behavior

- *Low severity behavior*: just display ads, don't transmit information
- *Medium severity behavior*: transmits information deemed low risk, such as location information, and may display ads based on this
- *High severity behavior*: transmits personal information, and displays ads tailored to devices, people with those characteristic
 - Typically very aggressive (annoying)
 - Sometimes called *malware*

Getting Adware On a System

- Put on a web site user visits
 - Put it in a banner enticing the user to click on it; this installs the adware
 - Page may require user to install software to view parts of web site; software contains adware
 - If page refreshes automatically, it may direct browser to run an executable
 - Usually browser notifies user via a dialog box that may require a click; on click, program runs and installs adware
 - Some browser plug-ins download, execute files automatically; there may be no indication of this
 - Called *drive-by downloading*

Getting Adware on a System

- Put into software that user downloads
 - Very common with mobile apps
- Problem: app asks for permission to carry out its tasks
 - Some may be unnecessary; often hard for users to minimize permissions set
 - Thus app may have access to camera, microphone, and may be able to make calls without going through dialing interface — and user does not realize this
- Example: survey of 900 Android apps
 - 323 had unnecessary permissions

Economic Motives

- Used to target ads that use is most likely to respond to
- Purveyors get money for every ad displayed or clicked on
 - Web site owners display ads on their sites
 - Developers put adware libraries in their apps
 - Others take apps, modify them to include adware, and put them on unauthorized app stores

Spyware

- Trojan horse that records information about the use of a computer for a third party
 - Usually results in compromise of confidential information like keystrokes, passwords, credit card numbers, etc.
 - Information can be stored for retrieval or sent to third party
- Put on a system the way any other malware gets onto system

Example: Pegasus

- Designed for Apple's iPhone, attacker sends URL to victim who clicks on it, triggering attack that tries to gain control of iPhone
- First sends HTML file exploiting vulnerability in WebKit
 - Basis for Safari and other browsers
- This downloads software to gain control of iPhone
 - Software enciphered with different keys for each download
 - Includes a loader for the next stage
- Loader downloads dynamic load libraries, daemons, other software and installs Pegasus
 - If iPhone has previously been jailbroken, removes all access to the iPhone provided by the earlier break

Example: Response to Pegasus

- Apple developed patches for the vulnerabilities exploited
 - Deployed them in update to iPhone's operating system, iOS
- Discovered when human rights activist received text messages with a suspicious link
 - Sent messages to Citizens Lab
 - Citizens Lab recognized links were associated with a manufacturer of spyware for government surveillance
 - Lookout carried out technical analysis

Ransomware

- Malware inhibiting use of computer, resources until a ransom is paid
 - Ransom is usually monetary, and must be paid through some anonymous mechanism (BitCoin is popular)
- PC CYBORG (1989) altered AUTOEXEC.BAT to count number of times system was booted; on 90th, names of all files on main drive (C:) enciphered and directories hidden
 - User told to send fee to post office box to recover the system
- CryptoLocker (2013) encrypted files and gave victim 100 hours to pay ransom; if not, encryption keys destroyed
 - Used evasive techniques to make tracking more difficult
 - Spread via email as attachments

Example Protocol

Goal: Angie wants to extort money from Xavier

- Angie generates asymmetric key pair; embeds public key in malware
 - She retains private key
- Malware infects Xavier's system
 - Generates symmetric key and uses that to encipher target data
 - Enciphers symmetric key with public key, erases all instances of symmetric key
 - Xavier sees message saying he needs to do something for Angie (usually send money); he does so and includes the encrypted symmetric key
- Angie then deciphers encrypted symmetric key with her private key, returns it to Xavier

Phishing

- Act of impersonating legitimate entity in order to obtain information such as passwords or credit card information without authorization
 - Usually a web site associated with a business
- Usual approach: craft a web site that looks like the legitimate one
 - Send out lots of email trying to persuade people to go to that web site
 - Copy their login, password, and other information for later use
- More vicious attack: fake web site passes data on to real web site, and sends replies back to victim
 - Man-in-the-middle attack

Example

- Heidi banks at MegaBank, with URL of <https://www.megabank.com>
- She receives a letter saying she needs to check her account for possible fraudulent activity
- Email includes link
 - Link is visible as www.megabank.com
 - But link actually connects to <https://www.megabank.crookery.com>
- Attacker records name, password, then give error
 - If very clever, client is redirected to actual bank's home page

Spearphishing

- Phishing attack tailored for particular user
- Used to attack specific (types of) users to obtain information
- Example: some employees of a major cybersecurity company received email called “2011 Recruitment Plan”
 - They opened an attacked spreadsheet
 - This exploited a vulnerability in a supporting program to install a backdoor so attackers could control system remotely
 - Attackers used this as a springboard to compromise other systems in the company’s network, and ultimately stole sensitive information
 - Embarrassment, financial costs of recovery large

Theory of Viruses

- **Theorem:** It is undecidable whether an arbitrary program contains malicious logic

Defenses

- Scanning
- Distinguishing between data, instructions
- Containing
- Specifying behavior
- Limiting sharing
- Statistical analysis

Scanning Defenses

- Malware alters memory contents or disk files
- Compute manipulation detection code (MDC) to generate signature block for data, and save it
- Later, recompute MDC and compare to stored MDC
 - If different, data has changed

Example: *tripwire*

- File system scanner
- Initialization: it computes signature block for each file, saves it
 - Signature consists of file attributes, cryptographic checksums
 - System administrator selects what file attributes go into signature
- Checking file system: run *tripwire*
 - Regenerates file signatures
 - Compares them to stored file signatures and reports any differences

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 \geq risk level
 - If credibility rating $<$ risk level, must use special command to run program

Antivirus Programs

- Look for specific “malware signatures”
 - If found, warn user and/or disinfect data
- At first, static sequences of bits, or patterns; now also includes patterns of behavior
- At first, derived manually; now usually done automatically
 - Manual derivation impractical due to number of malwares

Example: Earlybird

- System for generating worm signatures based on worm increasing network traffic between hosts, and this traffic has many common substrings
- When a packet arrives, its contents hashed and destination port and protocol identifier appended; then check hash table (called *dispersion table*) to see if this content, port, and protocol have been seen
 - If yes, increment counters for source, destination addresses; if both exceed a threshold, content may be worm signature
 - If no, run through a multistage filter that applies 4 different hashes and checks for those hashes in different tables; count of entry with smallest count incremented; if all 4 counters exceed a second threshold, make entry in dispersion table
- Found several worms before antimalware vendors distributed signatures for them

Behavioral Analysis

- Run suspected malware in a confined area, typically a sandbox, that simulates environment it will execute in
- Monitor it for some period of time
- Look for anything considered “bad”; if it occurs, flag this as malware

Example: Panorama

- Loads suspected malware into a Windows system, which is itself loaded into Panorama and run
 - Files belonging to suspect program are marked
- Test engine sends “sensitive” information to trusted application on Windows
- Taint engine monitors flow of information around system
 - So when suspect program and trusted application run, behavior of information can be recorded in taint graphs
- Malware detection engine analyzes taint graphs for suspicious behavior
- Experimentally, Panorama tested against 42 malware samples, 56 benign samples; no false negatives, 3 false positives

Evasion

Malware can try to ensure malicious activity not triggered in analysis environment

- Wait for a (relatively) long time
- Wait for a particular input or external event
- Identify malware is running in constrained environment
 - Check various descriptor tables
 - Run sequence of instructions that generate an exception if not in a virtual machine (in 2010, estimates found 2.13% of malware samples did this)

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
 - Key are assumption that certifying authority will *not* make mistakes and assumption that tools, supporting infrastructure used in certifying process are not corrupt

Example: Duff and UNIX

- Observation: users with execute permission usually have read permission, too
 - So files with “execute” permission have type “executable”; those without it, type “data”
 - Executable files can be altered, but type immediately changed to “data”
 - Implemented by turning off execute permission
 - Certifier can change them back
 - So virus can spread only if run as certifier

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

Karger's Scheme

- Base it on attribute of subject, object
- Interpose a knowledge-based subsystem to determine if requested file access reasonable
 - Sits between kernel and application
- Example: UNIX C compiler
 - Reads from files with names ending in “.c”, “.h”
 - Writes to files with names beginning with “/tmp/ctm” and assembly files with names ending in “.s”
- When subsystem invoked, if C compiler tries to write to “.c” file, request rejected

Lai and Gray

- Implemented modified version of Karger's scheme on UNIX system
 - Allow programs to access (read or write) files named on command line
 - Prevent access to other files
- Two types of processes
 - Trusted: no access checks or restrictions
 - Untrusted: valid access list (VAL) controls access and is initialized to command line arguments plus any temporary files that the process creates

File Access Requests

1. If file on VAL, use effective UID/GID of process to determine if access allowed
2. If access requested is read and file is world-readable, allow access
3. If process creating file, effective UID/GID controls allowing creation
 - Enter file into VAL as NNA (new non-argument); set permissions so no other process can read file
4. Ask user. If yes, effective UID/GID controls allowing access; if no, deny access

Example

- Assembler invoked from compiler

```
as x.s /tmp/ctm2345
```

and creates temp file **/tmp/as1111**

- VAL is

```
x.s /tmp/ctm2345 /tmp/as1111
```

- Now Trojan horse tries to copy x.s to another file
 - On creation, file inaccessible to all except creating user so attacker cannot read it (rule 3)
 - If file created already and assembler tries to write to it, user is asked (rule 4), thereby revealing Trojan horse

Trusted Programs

- No VALs applied here
 - UNIX command interpreters: *cs**h*, *sh*
 - Program that spawn them: *getty*, *login*
 - Programs that access file system recursively: *ar*, *chgrp*, *chown*, *diff*, *du*, *dump*, *find*, *ls*, *restore*, *tar*
 - Programs that often access files not in argument list: *binmail*, *cpp*, *dbx*, *mail*, *make*, *script*, *vi*
 - Various network daemons: *fingerd*, *ftpd*, *sendmail*, *talkd*, *telnetd*, *tftpd*

Guardians, Watchdogs

- System intercepts request to open file
- Program invoked to determine if access is to be allowed
 - These are *guardians* or *watchdogs*
- Effectively redefines system (or library) calls

Trust

- Trust the user to take explicit actions to limit their process' protection domain sufficiently
 - That is, enforce least privilege correctly
- Trust mechanisms to describe programs' expected actions sufficiently for descriptions to be applied, and to handle commands without such descriptions properly
- Trust specific programs and kernel
 - Problem: these are usually the first programs malicious logic attack

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 so that:
 - On first call, unique file identifier (inode) saved
 - On second call, inode of named file compared to that of first call; if they differ, attack may be underway ...

Specifications

- Treat infection, execution phases of malware as errors
- Example
 - Break programs into sequences of non-branching instructions
 - Checksum each sequence, encrypt it, store it
 - When program is run, processor recomputes checksums, and at each branch compares with precomputed value; if they differ, an error has occurred

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: Trusted Solaris system
 - All executables, trusted data stored below user region, so user applications cannot alter them

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

Overview of Vulnerability Analysis

- What is a vulnerability?
- Penetration studies
 - Flaw Hypothesis Methodology
 - Other methodologies
- Vulnerability examples
- Classification schemes
 - RISOS, PA, NRL Taxonomy, Aslam's Model
- Standards
 - CVE, CWE
- Theory of penetration analysis

Definitions

- *Vulnerability, security flaw*: failure of security policies, procedures, and controls that allow a subject to commit an action that violates the security policy
 - Subject is called an *attacker*
 - Using the failure to violate the policy is *exploiting the vulnerability* or *breaking in*

Formal Verification

- Mathematically verifying that a system satisfies certain constraints
- *Preconditions* state assumptions about the system
- *Postconditions* are result of applying system operations to preconditions, inputs
- Required: postconditions satisfy constraints

Penetration Testing

- Testing to verify that a system satisfies certain constraints
- Hypothesis stating system characteristics, environment, and state relevant to vulnerability
- Result is compromised system state
- Apply tests to try to move system from state in hypothesis to compromised system state

Notes

- Penetration testing is a *testing* technique, not a verification technique
 - It can prove the *presence* of vulnerabilities, but not the *absence* of vulnerabilities
- For formal verification to prove absence, proof and preconditions must include *all* external factors
 - Realistically, formal verification proves absence of flaws within a particular program, design, or environment and not the absence of flaws in a computer system (think incorrect configurations, etc.)

Penetration Studies

- Test for evaluating the strengths and effectiveness of all security controls on system
 - Also called *tiger team attack* or *red team attack*
 - Goal: violate site security policy
 - Not a replacement for careful design, implementation, and structured testing
 - Tests system *in toto*, once it is in place
 - Includes procedural, operational controls as well as technological ones

Goals

- Attempt to violate specific constraints in security and/or integrity policy
 - Implies metric for determining success
 - Must be well-defined
- Example: subsystem designed to allow owner to require others to give password before accessing file (i.e., password protect files)
 - Goal: test this control
 - Metric: did testers get access either without a password or by gaining unauthorized access to a password?

Goals

- Find some number of vulnerabilities, or vulnerabilities within a period of time
 - If vulnerabilities categorized and studied, can draw conclusions about care taken in design, implementation, and operation
 - Otherwise, list helpful in closing holes but not more
- Example: vendor gets confidential documents, 30 days later publishes them on web
 - Goal: obtain access to such a file; you have 30 days
 - Alternate goal: gain access to files; no time limit (a Trojan horse would give access for over 30 days)

Layering of Tests

1. External attacker with no knowledge of system
 - Locate system, learn enough to be able to access it
2. External attacker with access to system
 - Can log in, or access network servers
 - Often try to expand level of access
3. Internal attacker with access to system
 - Testers are authorized users with restricted accounts (like ordinary users)
 - Typical goal is to gain unauthorized privileges or information