Systems Test IV Study Guide

2017-12-05

Intro to Computer Systems Security

Information Risks

The Principle of Least Privilege

- Confidentiality
    - Access to information should only be **given** to *those who need it*
- Integrity
    - Information should only be **mutated** by *those who need to*
- Availability
    - Information should only be **accessible** when *those who need it should need it*

Privacy vs. Confidentiality

- Privacy
    - Applies to a single person
    - How a person is identified
    - The minimum amount of data required to run the business
- Confidentiality
    - Applies to data
    - An extension of privacy
    - What procedures are put in place to only authorize those who need the it

Physical Risks

- Disk failure
    - Affects availability (up-time), but also security
- Power failure/surge
    - Can cause physical damage to a computer
- Physical theft
    - Confidentiality problems, up-time loss
    - May have integrity problems

People Risks

- External
    - Steal secrets
    - Reveal company plans
    - Destroy/delete
- Internal
    - Downsizing
    - Angry at management

Can be accidental, or malicious.

Other Risks

- Denial of service attacks
- Erasing key components
- Compromising file content
- Extracting passwords
- Extracting financial information

Top Security Risks

- Unpatched Client-side software
    - CVE's found all the time, and not patching them can leave security holes open.
    - Flash, Quicktime, Java
- Internet-facing web sites
    - Turn trusted sites malicious
    - Use similar names domains to phish users

Types of Cyber Attacks

- Virus', Trojans, Worms
- Botnets
- Web-based attacks
- Stolen devices
- Malicious code
- Malicious insiders
- Phishing/Social engineering
- Denial of service

Exploitation Trends

- Vulnerabilities in *applications* greater than those in the base *OS*.

Levels of exploitations:

1. Applications
2. OS Libraries
3. OS Transport
4. Network

*** PATCH YOUR SHIT***

Why Application Exploits Occur

Security doesn't know development, development doesn't know security.

Web Application Attacks

- Brute force password guessing
    - Microsoft SQL, FTP, and SSH servers are popular targets
    - The access that is gained if valid login is found is huge (admin)
- Web Application attacks
    - SQL Injection, XSS, PHP File, PHP Email, Just PHP in general ***help***
    - Automated tools to target specific vulnerabilities and exploits.

Zero Day Vulnerability

- Flaw in software code is discovered and exploited before a patch is ready.
- Vulnerabilities are oftern found in popular 3rd party addons (Flash), or software suits (Office)
- One of the **most** significant threats
- Can go for as high as **$1,500,000** each on the black market

Case Study - Stuxnet (2005 - 2010)

- A malicious computer worm discovered by *Kaspersky Labs*.
- Used by an unknown actor (**cough** *The CIA* **cough**) to target SCADA systems, damaging Iran's nuclear program.
- Used an unprecedented **four** zero-day Windows exploits.

Implications to Design

- Hardware
    - Router/Modem
    - Firewall
    - Physical Space
    - Printers, Output
- Software
    - Operating Systems
    - Patching
    - Updates
    - Third-party software
- Development
    - Programming Environment
    - Database
    - Third-party tools

- Testing
    - Exposure
    - Privacy of data (obfuscation)
    - Third-party users/testers
- Installation
    - Data migration
    - Patch levels
    - Third-party software
    - Open Ports/Exposure of web sites
- Ops
    - Maintaining patch level
    - Up to date AV Software
    - New hardware/software solutions
    - Backups, Recovery
- Project Management
    - Protecting Assets
    - Communication of details and privacy
    - External personnel attacks
    - Internal personnel attacks
- Network
    - Open Ports
    - Firewall
    - Availability
- Physical Space
    - Printers, confidentiality
    - Protection of hardware
    - Climate control
    - Location
    - Transportation
    - Backups
    - Disaster Recovery
- Application design
    - Exposure
    - Authentication
    - Authorization
    - Database Vulnerabilities
    - SSL
    - SQL
    - Interfaces to external systems
    - Timeouts

Enterprise-Wide Web Application Security

Web application security testing must be applied in all phases of the Application life-cycle, by all constituencies throughout the enterprise.

- Developers
    - Must have clear cut security requirements during Development, and QA
    - Need to have automated testing during Development
    - Utilize secure coding standards
- Quality Assurance
    - Must test for functionality, but also security
    - Must test environments for potentials flaws and insecurities
    - Must provide detailed security flaw reports
    - Require automated testing that integrate into the environment
- Security
    - Just continually test application in real world environment to asses impact of changing code.
    - Must look for all levels of web vulnerabilities
        - Platform
        - Information
        - Application
- Auditors, Risk Compliance
    - Define regulatory requirements during the definition phase
    - Asses applications once they are in Production
    - Must act a resource for what is, and isn't acceptable

Important Terms

- Reverse Directory Traversal
    - Access to restricted directories by guessing the URL path
- Path Truncation
    - Removing file names from the URL to access the directory
- Cooking Manipulation
    - Reading and/or changing cookie values
- Directory Enumerations
    - Accessing hidden files, directories on a web site
- Parameter Manipulation
    - Changing parameter data between client, server
- XSS (Cross-site Scripting)
    - Injecting client side scripts into web applications

- SQL Injection
    - Entering malicious SQL Code in a field with the intent of it being executed by the database
- Buffer Overflow
    - Overflow the buffer to write data outside the bounds the application is suppose to use

Secure Coding

From the Beginning

Security is not something that can be added as an afterthought

> You must identify the nature of the threats to your software, and include secure coding practices throughout development.

SDLC

Software flaws can be found at any stage:

- Not identifying security requirements up front
- Creating logic errors is conceptual designs
- Using poor coding practices
- Deploying improperly
- Adding flaws during maintenance, updating

What, Why is Secure Coding

The practice of writing programs that are resistant to attack by malicious targets.

An insecure program can provide access for a malicious actor to take control of a computer, resulting in:

- Denial of Service Attacks
- Compromise of Secrets
- Damage to users/corporate systems

Web application attacks are 60% of the total attacks on the internet.

Secure Coding Practices

1. Valid input
    - Validate input from all untrusted (aka ***USERS***) sources
    - Proper input validation can eliminate the majority of vulnerabilities
    - Be suspicious of most external data sources
2. Heed compiler warnings

    - Compile code using the *highest* warning level, eliminate warnings by changing the code
    - Use static, dynamic analysis tools to detect and eliminate further vulnerabilities

3. Architect, design for security policies

    - Create a software architecture, design your software to implement and enforce security policies

        - If system requires different privileges, consider dividing the system

4. KISS

    - Keep it simple, *shithead*
    - Complex designs increase likelihood of errors

5. Default: **DENY**

    - Whitelist, not blacklist
    - Protection scheme identifies *who* has access, not *who* does **not**

6. Principle of lest privilege

    - Every process should execute with the lowest level privileges necessary

7. Sanitize data

    - Clean data of possible command injection attacks

8. Practice defence
9. Effective testing techniques

    - Fuzz testing, pen testing, code audits should all be incorporated

10. Adopt a secure coding standard

    - Develop and/or adopt a secure coding standard for your language and platform

Types of Security Vulnerabilities

Buffer Overflows

- When an application writes data past the end/beginning of a buffer
- If the input data is not truncated, it will overwrite other data in memory
- Input is stored temporarily in one of two places:

    - The memory stack
    - The heap (reserved memory, `malloc`)

- Attempt to enter a string longer than the legal filename
- Use a datablock larger than what's asks for

Unvalidated Input

- Can be used in a number of ways:

    - Buffer overflows
    - Format string vulnerabilities
    - URL commands
    - Code insertion
    - Social engineering

- There should be an input validation routine for the application
- Encode data to a common character set
- Fail should reject input
- Validate **all** client provided data (including third-party data)
- Validate redirect data
- Validate data types (or just use a real language :wink:)
- Validate data range
- Validate data length
- Validate against a whitelist of character if possible

Race Conditions

- Occurs when changes to the order of events changes behaviour
- If a *specific* order must be required, it is a bug
- Files being altered during read/write operations

Interprocess Communication

- Separate processes sharing information
- Shared Memory/Sockets
- Always assume the other end of the channel is hostile

Insecure File Operations

- Do not assume ownership, location, or attributes of a file
- Lock files while they are being used
- Don't read/write from a public directory
- *Principle of Least Permissions* a directory
- Check if a file exists before writing
- Verify that a file is the file you expect (Checksum)

Access Control Problems

- Improper use of authentication, authorization, permissions, certificates

Secure Storage, Encriptions

- ***ENCRYPT EVERYTHING***

Injection Attacks

- The most common is SQL injection, takes advantage of SQL syntax to execute arbitrary commands
- XSS is used to inject code into a website (Fake logins, keys, etc)

Database Security

Designing Security for Files

- Physical table security
    - Protect data from failure or data loss
    - Secure from unauthorized use
- Security is achieved primarily by implementing controls on each file
- Types of security:
    - File backups
    - User access
- Techniques for file restoration:
    - Periodically make a backup copy of a file
    - Store a copy of each change to a file in a log/audit trail
    - Store a copy of each row before/after it's changed
- *Encrypt* the data in the file
- Require users to *identify* themselves
- Prohibit direct manipulation of the file, use a *working copy*

Security Mechanisms in DBMSs

- Different degrees of access
    - DBMS must must have access controls at various levels
        - Table
        - Column
        - Row
        - Elements
- Different access modes
    - Select
    - Insert
    - Update
    - Delete
- Different types of access control
    - Name-dependent: name of object
    - Data-dependent: value of object
    - context-dependent: depends on time/user/location/etc.
- Dynamic authorization, change while db is operational
- Auditing
    - Security related events should be reported in *journals*, *audit trails*, *system logs*
- Flow control
    - Check the destination of output through authorization access

- No back doors
    - Access must be through the DBMS
- Reasonable performace
    - Security should not increase execution times significantly

Integrity Mechanisms in DBMSs

- Well-formed transactions
    - Updates may only occur via transactions
- Authenticate users
    - Updates may only occur via authorized users
- Least privilege
    - Minimum update rights for a task
- Separation of duties
    - No user should be able to corrupt data on their own
- Continuity of operations
    - Run in a disaster (redundancy)
- Reconstruction of events
    - More audit trails
- Reality checks
    - ???
- Easy of safe use
    - Easy to use, fault-free
- Delegation of authority
    - DBMS should assign privileges according to policies

DB Security Guidelines Overview

- Economy of mechanisms
    - Mechanisms should be as simple as possible
- Efficiency
    - Mechanisms should be efficient
- Linearity of cost
    - Operations costs == actual use of mechanisms

- Privilege separation
    - Layered mechanisms, multiple passwords

SQL Server Security Best Practices

Authentication Mode

- Windows Authentication
    - Specific Windows user and groups are trusted to log int o SQL Server
    - No passwords are passed across the network
- SQL Server Authentication
    - SQL Login created for the user
    - Passed across the network
- SQL Server has two modes during database setup
    - Windows Authentication
    - Mixed Mode Authentication

Authentication Mode Best Practices

- Always use *Windows* Authentication if possible
- Use *Mixed Mode* for legacy applications, non-Windows users
- Change the `sa` password. Use a strong password, change regularly
- Do **not** manage SQL Server using `sa`; assign `sysadmin` to a known user or group.
- Rename `sa` to prevent name collisions

Password Policy Best Practices

- Mandate a **strong** password policy, include *expiration*, *complexity* policies
- If using SQL Logins, ensure 2008 runs on Windows server 2008
- Outfit applications to change SQL login passwords
- Set `MUST_CHANGE` for new logins

Administrator Privileges Best Practices

- Use admin only when needed
- **Minimize** the number of admins
- Avoid dependency on default admins Windows group

Database Ownership

- Members of `sysadmin` are *DBOs* in every database
- There is a database role `db_owner` in every database. Same privileges as `dbo`
- Have distinct owners, not all dbs should be owned by `sa`
- **Minimize** owners for each database
- Confer trust selectively

Schema Best Practices

- Group like objects into a schema
- Have distinct owner for schemas
- Not all schemas should be owned by `dbo`
- **Minimize** owners for each schema

Object Authorization Best Practices

- Manage permissions via *db roles*/*windows groups*
- Do **not** enable `guest` access

Data Encryption Best Practices

- Encrypt
- Use *symmetric* keys
- *Password protect* keys
- Backup keys