Security +
Software Security

Author: Joseph Lee
Email: joseph@ripplesoftware.ca
Mobile: 778-725-3206

Software Security Concepts

Mail Gateway
- An **application proxy** for an email server that adds security features such as **filtering to reduce the risk** associated with email
- Mail-gateways can include **DLP capabilities** to look for **exfiltration** of classified documents

CGI Common Gateway Interface
- Any program that runs on a web server
- CGI defines a standard way in which information may be passed to and from the browser and server
- CGI defines a standard way in which information may be passed to and from the browser and server
- Any program or script that can process information according to the CGI specification can, in theory, be used to code a CGI script

Application Whitelisting / Blacklisting
- Setting a list of explicit allowed / disallowed applications

Trusted Operating System
- Defines an operating system that meets a set list of requirements. These requirements primarily focus on authentication and authorization
- Commonly uses MAC (Mandatory Access Control) model of access control

- **Evaluation Security Level (EAL1 - EAL7)**
  - A numerical grade assigned upon completion of a **Common Criteria** security evaluation
  - Conventional commercial operating systems are usually EAL4
- **TOE Target of Evaluation**
  - Acronym for the hardware / software system that is subjected to testing
- **PP Protection Profile**
  - Set of formal security targets for TOE’s to be specified to
- **Hardened Configurations**
  - Are available from the software vendor after the Common Criteria certification process has been completed

RTOS Real Time Operating System
- An operating system that reacts to input within a specific time

- If the system does not react within a set time, the system can report errors and shut down a process such as manufacturing process

Patching
- Applying software updates to fix software bugs and /or vulnerabilities

Compiled Code / Language
- Code that has been optimized into machine code
- Provides optimized performance when the program operates, but needs to be compiled in preparation of being run

Runtime Code / Language (AKA Interpreted or Scripted language)
- Code that is compiled at runtime and so is easily readable and modified by users

Hybrid Compiled / Runtime Language (JIT compiled language)
- Combines the advantages and drawbacks of both types of languages
- Allows adaptive optimization, such as dynamic recompilation and micro-architecture-specific speedups
- Some languages have JIT compilers such as:
  - Java -> JVM
  - Jython, JRuby, Groovy, Tamarin also have JIT compilers

Testing Paradigms
- **Black-box testing**
  - Starting penetration test without any knowledge of the internal network system
- **White-box testing**
  - Starting penetration testing with full knowledge of the target network
- **Gray-box testing**
  - Limited or partial knowledge of the target network

Non-intrusive testing
- Essentially is **vulnerability scanning** and **considered passive**, but may be illegal to perform on a someone else's server/network

Intrusive testing
- Essentially is penetration testing (requires permission and detailed contracts of what is allowed)

Input validation
- The practice of checking that data is valid before using it.
- Input handling is required for website forms and other website data coming in from web-applications.
- Some example types of validation include:
  - Validate characters
  - Boundary or range checking
  - Variable type-checking

    - Filtering html or SQL code from input
    - Otherwise filtering specific characters such as escape characters.
    - Client-side and Server-side validation

Data-encoding
- The process of encoding data into a specific encoding format for specific uses

HTML-entities
- Some characters should be encoded if they are being used in HTML

Race-conditions
- Can occur when multiple applications, services, or threads attempt to access the same resource at the same time
- Since the data may be changed and the condition of one event may depend on the state of the data at a certain point
- Loading same options into multiple web-pages and then allowing those options to be reserved by customers may result in race-conditions, where two customers have booked the same item / root / seat

Error Handling and **Exception Handling**
- By gracefully handling errors, software can increase availability and also protect remote access to resources by malicious actors
- Logging errors and exception make it possible to find them and update software to fix those errors

SDK Software Development Kit
- Third party libraries that can provide code reuse and prevent the need to develop large modules of an application.

Code obfuscation
- By writing your software in a style that is difficult for others to read, you can prevent some degree of ability to understand the code.
- Not a good security measure
- Often used in malware / viruses

Code Quality Testing
- **Static code analysis**
    - Examines the code without executing it
- **Dynamic analysis**
    - Checks the code as it is running
- **Fuzzing**
    - A testing process to send random data to an application to try to make it crash
- **Stress testing**
    - Attempts to **simulate high loads** in a live environment
- **Sandbox**
    - Isolated area for testing software

- **Model verification**
  - Checks that the software meets its intended purpose and returns consistent and accurate results
- **Dead code**
  - Code that never gets executed or used
  - This deal code should be removed in production releases
- **Regression testing**
  - Re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change

SDLC Development Life Cycles Models

- **Waterfall**
  - Multiple stages of software development:
    - Requirements
    - Design
    - Implementation
    - Verification
    - Maintenance
- **Agile**
  - Uses a set of principles started by cross-functional teams that use iterative lifecycles of development to work towards a finished product
  - Stress interaction
  - Creating a working application
  - Collaboration / feedback from customer
  - **Agile stages** cycle through the following stages:
    - Plan
    - Create
    - Verify
    - Package
    - Release
    - Configure
    - Monitor

Secure Development Operations

- Software development process that includes extensive communication between software developers
- **Security automation**
  - Automated tests and notifications to check software
  - Scheduled or constantly active
- **Continuous integration / version control**
  - Merging software changes into a central repository
  - Tracks previous versions of software as it is updated to allow rebasing to older versions
  - Software allows the features to be developed and integrated into
- **Baselining**
  - Integrating individual feature branches into the main branch and testing for bugs regularly
- **Immutable systems**

    - Cannot be changed
    - An immutable instance of a software application cannot be directly altered and so it stays secure against altering the code to do malicious things
- **Production environment**
    - Live environment for the software in use by end-users
- **Controlled environment**
    - Controlled environment is the development environment
- **Infrastructure as code**
    - Scripts that can deploy VMs configured to run the application / service
- **Change management**
    - Helps ensure that developers not make unauthorized changes and that changes do not cause unintended outages
    - Accountability and documentation for auditing are available with change management
    - **Snapshots** and **software tools** can allow a **revert to known state** or **rollback to known configuration** which can undo changes if problems occur during updates
- **Dead code**
    - Is code in the application that is never used in the function of the application
    - This dead code is left over from earlier development stages of the application
- **Code reuse**
    - Code reuse adds security especially when using software libraries from popular languages
    - The downside is that in the case of zero-day attacks, attackers can predict which libraries and modules you may be using in your application
- **Code quality**
    - Code should be debugged and undergo proper testing to ensure it is functional and optimized for performance to lower the burden on servers
- **Memory management**
    - Proper coding standards such as type-checking and garbage-collection improves the security if your software by increasing availability and reducing the change of buffer-overflows
- **Static code analysis**
    - Is a debugging tool that reads source code but does not run the code
- **Dynamic analysis**
    - Running and testing portions of code for logic, inputs, interface, memory management, performance, etc.
- **Fuzzing**
    - Entering unexpected data into the application to test it’s reaction / stability
    - Using random data is called **monkey fuzzing**
- **Stress testing**
    - An aggressive form of testing the performance of an application under high-use stress such as a web-server under high number of page-loads, or with large files
- **Model verification**

    - The term **model** defines how defines how developers expect the application to work
    - Model verification is to user-end test the application to verify that it functions as expected
- **Staging**
    - Staging moves the code from development stage to servers that **mimic production servers**
    - Staging is an intermediary stage between development and production
    - Stress testing can occur at this stage
- **Production**
    - The web-application is installed onto a live server and online

Database Schema

- **Normalization**
    - Database normalization refers to organizing the tables and columns to reduce redundant data and improve overall database performance
    - This also provides a standardized database template in order to standardize knowledge and expectations of database functionality
    - **First Normal Form**
        - Each row in the table is unique and identified with primary key
        - Related data is contained in a separate table
        - None of the columns include repeating groups
    - **Second Normal Form**
        - Have a **composite primary key** where two or more columns make up the primary key
        - NF2 databases meet NF1 database form rules
        - Each column dependent on the entire composite primary key
    - **Third Normal Form**
        - NF3 databases meet the NF2 form rules and also therefore NF1 rules
        - All columns that are not primary keys are only dependent on the primary keys None of the non primary key columns are dependent on each other
        - This reduces redundancy in the data
- **Stored Procedures**
    - Are hardcoded into the database server application.
    - These stored procedures can prevent database injection attacks because they are static and cannot be changed
- **Injection attacks**
    - Attacks that attempt to inject SQL code into HTTPS POST or GET requests. By attempting to have SQL code submitted with the data, the attacker will try to alter the database or reveal the contents of the database