SECURE BY DESIGN

By Harippriya Sivapatham, IAM

Most organizations have a well-oiled machine with the sole purpose of building functional software. However, the increasing concerns and business risks associated with insecure software has increased the awareness for integrating security into the development process**. Secure Development Life Cycle (SDL)** is gaining a lot of traction in the industry and also within Micro Focus.

Security has always been an afterthought and SDLC intends to change that. Architects and developers play a vital role in the success of SDLC. It is very efficient and easy to rollout SDL if products are already built with security in mind. This article is a concise list of various **Security Principles** that would help to build a secure product.

Secure Design

Security Architecture is one component of the product architecture. It addresses where and how the security of the components must be handled. Some generic high level security principles to consider while designing are:

✓ Modularize and secure

Modularize your system and secure access to every module. This loose coupling helps with reuse of modules and also introduces multiple layers to security.

✓ Separation of Duties

It is a classic security method to prevent fraud and from giving too much power to one individual. For example, an administrator should not be able to change the audit settings for himself.

✓ Implement CSRF prevention

Cross Site Request Forgery is one of the top 10 OWASP vulnerabilities. An infrastructure must be put in place to support random CSRF tokens to be managed and validated for each request.

✓ Reuse existing security components

Do not implement your own security mechanism like encryption algorithms, cryptography, etc.

✓ Defense in Depth

Avoid single point of failure. Support multiple levels of security, so that it is not easy to exploit a system.

Secure Development

Validate Input

- Input validation mitigates some of the Top 10 Vulnerabilities listed by OWASP:
    - Injection attacks – Command Injection, LDAP Injection, SQL Injection, etc.
    - Cross Site Scripting (XSS)
    - Directory traversal attacks
    - Un-validated forwards and redirects
- Input validations must be performed on the server..
- Use whitelists -- Validate against a “whitelist” of allowed input.
  For example, prevent Directory traversal attacks by allowing only accepted paths or filenames, instead of checking for unaccepted character sequences like “../../”.

Sanitize Output

- Sanitization of data removes harmful characters and then encodes the data.
- Sanitize on the server side and encode all data returned by the server.
- Prevents Cross Site Scripting (XSS) attacks.

Secure sensitive data

- Do not disclose session IDs, passwords, actual file path, etc. in error messages, debug logs or URLs.
- Do not display debug or error stack traces on the UI.
- Customize error pages to display generic messages.
- Remove unnecessary response headers with information about OS, application framework, web server, etc.

Secure Memory

- C++ : Guard against buffer overflow
- Java: Close resources explicitly. Also, do not use String variables to store passwords. These could be obtained by doing a java memory dump. Use byte arrays instead.

Secure Files

- Save uploaded files in a location outside of the web context.
- Turn off execution privileges on the file upload directories
- Do not accept filenames for well-known files. Use IDs instead.
- Do not send absolute file path to client.

Miscellaneous

- Fix compiler warnings
- Run static code analyzers and fix vulnerabilities reported by them.
- Do not hard code passwords.

Secure Deployment

- Configure to use higher TLS version, stringent ciphers, better key size, etc.
- Run all processes with least privilege possible.
- Properly secure passwords and crypto keys.

Summary

This article is a designed to be a quick reference for the security principles to be considered during various phases of a product life cycle. It is a collection of insights and best practices gathered from various sources.

The best place to start learning more about security is OWASP – Open Web Application Security Project. It list the top security vulnerabilities and provides various projects to help mitigate them.