License Dashboard Secure Coding Guidelines

PHX055

Contents

1. Document Control

1.1 Purpose of this guide

This document establishes the Secure Application Development Code standards and guidelines, this document establishes the minimum practices to ensure secure code is developed and implemented in all License Dashboard systems.

1.2 Scope

This document applies to all members of staff that perform development of systems owned by the License Dashboard team of Phoenix Software Ltd.

2. Top 10 Critical Security Risks

Everyone involved in the software development lifecycle (SDLC) from design through to operation must consider the security of data as an integral part of their role. The security of our systems should be constantly challenged, questions asked and discussed, the risk understood, and a plan put in place to fix or mitigate.

No system exhibits perfect security, but though open challenge and discussion security risks can be identified and reduced to a tolerable level.

Some practical steps include the following:

- regularly ask the question, "Is there a security implication here?"
- encourage developers to change their working practices to include security and support them with training
- make supportive security tools available
- encourage senior team members to lead by example
- share an understanding of how individuals directly affect security
- hold workshops where you think like an attacker and try to break your systems
- challenge your team's security practices

Source: NCSC Secure development and deployment guidance

The top 10 most common vulnerabilities for web applications are:

1. Injection
2. Broken authentication
3. Sensitive data exposure
4. XML external entities (XXE)
5. Broken access control

6. Security misconfigurations
7. Cross-site scripting (XSS)
8. Insecure deserialization
9. Using components with known vulnerabilities
10. Insufficient logging and monitoring

Source: OWASP Top Ten 2017

The top 10 most common vulnerabilities for desktop applications are:

1. Injections
2. Broken Authentication & Session Management
3. Sensitive Data Exposure
4. Improper Cryptography Usage
5. Improper Authorization
6. Security Misconfiguration
7. Insecure Communication
8. Poor Code Quality
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Source: OWASP Desktop App Security Top Ten 2021

3. Policy

The adherence to and use of the Secure Code Guideline document is a requirement for all software development on License Dashboard information technology systems

All code developers shall verify that their code is in compliance with the most recent and approved coding standards and guidelines.

Only validated code shall be implemented into the company's production environment.

A review and validation ensure that code exhibits fundamental security properties to include correctness, predictability, and attack tolerance.

3.1 Security in the software development lifecycle

It is paramount that security requirements are identified throughout the development lifecycle.

Application Code developers shall:

- security implications are identified during requirement gathering and refinement

- ensure code meets the level of confidence that software is free from exploitable code vulnerabilities, regardless of whether they are already designed into the software or inserted later in its life cycle
- ensure code provides predictable execution or justifiable confidence and that the software, when executed, will provide security functionality as intended
- never trust incoming data to the system, apply checks to this data
- never rely on the client to store sensitive data no matter how trivial
- disable error messages that return implementation information to the user
- applications must validate input to ensure it is well-formed and meaningful
- personal data is only available to the relevant people in accordance to the Data Protection Act
- credit card information and credentials are all encrypted
- all web applications consider the items detailed in OWASP Top 10 – 2017
- all desktop applications consider the items detailed in OWASP Top 10 Desktop Application Security Risks – 2012
- ensure all new development with security implications has automated test coverage
- ensure a peer code review is undertaken for all changes

3.2 Security of the development environment

Access to the License Dashboard DevOps organization is restricted to members of the development team and certain key internal stakeholders. Access is further restricted via DevOps group membership to project administrators, project team members, external developers, stakeholders, and readers.

Who has access and their level of access is periodically re-evaluated.

3.3 Transfer of customer data via the internet

Automated transfer of customer data for the SAM managed service will adhere to the following guidelines:

- All communication is encrypted using HTTPS/TLS
- Data is compressed and if necessary encrypted using 256-bit AES encryption
- The encrypted package is uploaded using an encrypted connection to an SFTP or FTPS server using credentials unique to the customer. Each customer has their own folder on the SFTP/FTPS server.
- The encrypted package is automatically moved off the SFTP/FTPS server to an internal file store that is not externally accessible, only then does the package get decrypted.

3.4 Secure source code repositories

All source code is stored in Microsoft DevOps in a collection of Git repositories, which are restricted to members of the development team and certain key stakeholders. Access to source code is controlled by the Development Manager and Lead Software Developer. The source code repository allows for:

- Granular permissions to be defined - https://msdn.microsoft.com/en-us/library/ms252587.aspx;
- Version control of source code
- Feature branches and code review via pull requests
- Auditing and Reporting
- Build and test pipelines which are triggered on code change and schedule
- Release pipelines which copy tested build packages to the release archive

3.5 Code reviews

Git has a mechanism to enforce code reviews through feature branches and pull requests. Pull requests are a mechanism to allow other team members to review the changes, make comments where they believe further changes should be made (e.g., incorrect functionality, code maintainability, code style etc) and when satisfied, approve the request which allows the merge to take place. Code can only be merged into the main development branch through a pull request and each pull request requires at least one approver. The creator of the pull request can't approve their own request. We have a checklist of items to review as part of the pull request including:

- Does the code meet our coding style guidelines?
- Is the code well commented?
- Is the code maintainable?
- Are new third-party dependencies from a trusted source, are well-used/popular and have a justification why we are using them?
- Analyse the code for potential security implications – are they mitigated?
- Has the code followed OWASP recommended practices?
    - Introduction - OWASP Cheat Sheet Series
    - OWASP Top Ten Web Application Security Risks | OWASP

3.6 Static code analysis

Static code analysis at compile time is performed by Microsoft Roslyn analyzers for C# projects or equivalent for other languages which are added to the project being built. The analyzers check the project for various design, naming, performance, maintainability, reliability, and security issues. All our projects use a common analyzer configuration file which sets all security issues to cause a build error which prevents the build pipeline progressing further. This configuration file is under source control so changes can be audited and monitored.

3.7 Static vulnerability scanning

We use Snyk to perform vulnerability scanning of all 3rd party libraries used by our solutions and alerts if any vulnerability is discovered in a library we are using. Snyk also performs security static analysis of the codebase. Scans are performed weekly and email alerts are sent after each scan. Snyk allows potential vulnerabilities to be assessed and ignored with mandatory reason if not deemed a risk.

3.8 Automated tests

The automated test suites include both functional and security tests. The security tests are designed to ensure any security aspects e.g., authentication, role-based access controls are correctly implemented and haven't regressed. A failure of any automated test causes a build error which prevents the build pipeline progressing further.

Version Control

Author	Version	Date	Description
Andy Livesey	1.0	26/08/2021	Branched from PHX041 Secure Coding Guideline
Andy Livesey	2.0	21/07/2022	Amendments following annual review
Andy Livesey	2.0	06/07/2023	Annual Review – no changes

Document Approval

Name	Version	Date	Position
Andy Livesey	1.0	26/08/2021	Development Manager
Clare Metcalfe	2.0	30/09/2022	Operations Director

Signed:  Clare Metcalfe, Operations Director

Dated: 30/09/2022