Reward Gateway Secure Development Policy

Classification – Confidential

September 2018 – Version 1.3

265 Tottenham Court Road
London
W1T 7RQ
UK

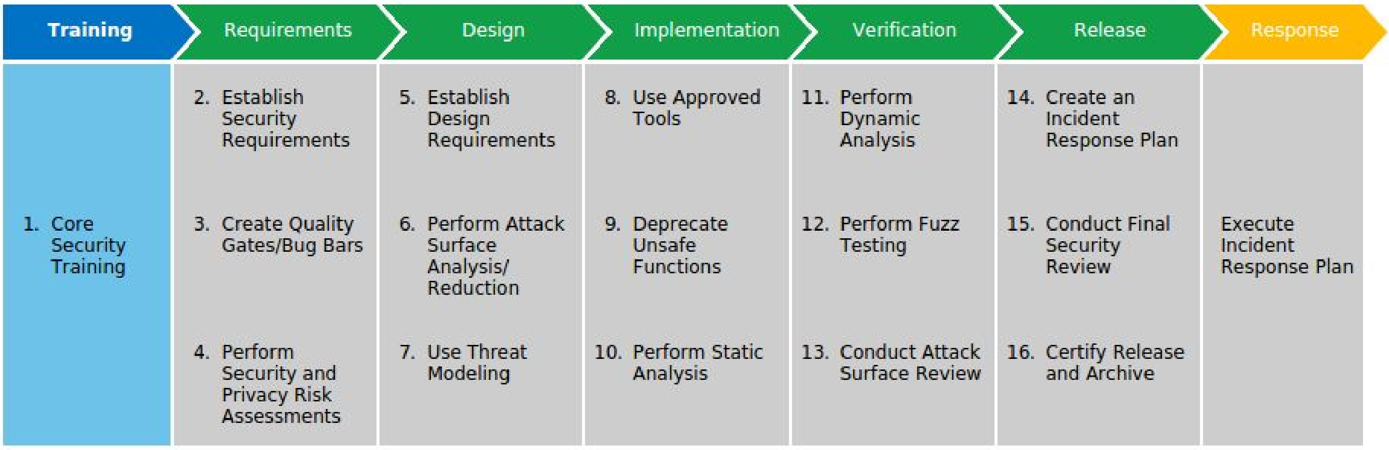
Table of contents

Document purpose

This document represents the general guidelines on secure software development at Reward Gateway.

Secure development overview

Reward Gateway has adopted elements of Microsoft's Secure Development Life Cycle for Agile. These are as follows:



Training	Requirements	Design	Implementation	Verification	Release	Response
	2. Establish Security Requirements	5. Establish Design Requirements	8. Use Approved Tools	11. Perform Dynamic Analysis	14. Create an Incident Response Plan	
1. Core Security Training	3. Create Quality Gates/Bug Bars	6. Perform Attack Surface Analysis/ Reduction	9. Deprecate Unsafe Functions	12. Perform Fuzz Testing	15. Conduct Final Security Review	Execute Incident Response Plan
	4. Perform Security and Privacy Risk Assessments	7. Use Threat Modeling	10. Perform Static Analysis	13. Conduct Attack Surface Review	16. Certify Release and Archive	

Practice details

Core security training

This practice is a prerequisite for all subsequent steps and implementing a Secure Development Life Cycle. Foundational concepts for building better software include secure design, secure coding, security testing, and best practices surrounding privacy.

When? – Before sprints start.

How? – Training is performed on an on-going basis.

Establish security requirements

Security and privacy analysis includes defining minimum security and privacy criteria for a project and creating the relevant work items.

When? – This should be done towards the start of the project and identified through the user stories.

How? – A common set of Security Requirements should be drafted.

Create quality gates / bug bars

Defining minimum acceptable levels of security and privacy quality at the start helps a team understand risks associated with security issues, as well as identifying and fixing security bugs during development. This applies the standards throughout the entire project.

When? – This should be identified before a project is started.

How? – Reward Gateway have chosen to set this bug bar at zero for all new projects. No project will be shipped if a readily identifiable security bug exists.

Perform security and privacy risk assessments

Examining software design based on costs and regulatory requirements helps a team identify which portions of a project will require threat modeling and security design reviews before release of a feature, product, or service.

When? – This should be done as part of the project.

How? – Reward Gateway already consider Security and Privacy to be the highest operational priority. Any new feature should obey the existing Privacy Policy or require an amendment to it. Product Managers should readily identify the situations that this is required prior to a project being started. Further training will be required and given in this area.

Establish design requirements

Addressing security and privacy concerns early helps minimise the risk of disruption. Validating all design specifications against a functional specification involves accurate and complete design specifications, including minimal cryptographic design requirements and a specification review.

When? – This is an ongoing exercise throughout the project.

How? – This is done throughout the project on the user stories as the Acceptance Criteria provided by the Product Manager, or identified during Sprint Planning. Cryptographic algorithms should be adopted based on Reward Gateway's Acceptable Encryption Policy.

Perform attack surface analysis/reduction

Reducing the opportunities for attackers to exploit a potential weak spot or vulnerability requires thoroughly analysing overall attack surface. This includes disabling or restricting access to system services, applying the principle of least privilege, and employing layered defenses wherever possible.

When? – This is an ongoing exercise but any formal requirements should be identified at the start.

How? – For most Reward Gateway projects, this step is not required as the attack surface is known to be the web-facing elements on the frontend and in Reward Manager. A common list of attack surfaces will be produced. If a new API is being introduced or used, it should first be evaluated for weaknesses.

Use threat modeling

Applying a structured approach to threat scenarios during design helps a team more effectively and less expensively identify security vulnerabilities, determine risks from those threats, and establish appropriate mitigations.

When? – Every sprint.

How? – This step will require training from the Information Security Manager. A common list of attack surfaces will be produced.

Use approved tools

Publishing a list of approved tools and associated security checks (such as compiler/linker options and warnings) helps automate and enforce security practices easily and at a low cost. Keeping the list regularly updated means the latest tool versions are used and allows inclusion of new security analysis functionality and protections.

When? – Every sprint.

How? – Reward Gateway do not have an approved set of tools for development but do require an environment controlled by Puppet to be used. This environment has been extensively tested and hardened with the relevant configuration settings in place to match the live environment.

Deprecate unsafe functions

Analysing all project functions and APIs, banning those determined to be unsafe, helps reduce potential security bugs with very little engineering cost. Specific actions include using header files, newer compilers, or code scanning tools to check code for functions on the banned list and then replacing them with safer alternatives.

When? – Every sprint.

How? – These are documented in the Code Review checks which are performed prior to work being merged into the development branch. This covers the use of deprecated functions and banned PHP constructs. Third-party libraries are updated based on the output from SensioLabs Security Advisories Checker by the continuous integration server.

Perform static analysis

Analysing the source code prior to compile provides a scalable method of security code review and helps ensure that secure coding policies are being followed.

When? – Every sprint.

How? – PHP is a dynamic language meaning that traditional static analysis cannot be performed. Instead, we use tools provided by the PHP community, in particular, the PHP Quality Assurance Toolchain (used on each developer workstation) and Scrutinizer CI are both an integral part of the daily integration checks. BlackDuck scans the code base daily and alerts on known security vulnerabilities and new security threats.

Perform dynamic analysis

Performing runtime verification, checks software functionality using tools that monitor application behavior for memory corruption, user privilege issues, and other critical security problems.

When? – At sprints across the project.

How? – PHP does not readily suffer from issues such as memory corruption, but user privilege and other security problems may be identified during the test process. These should be created as work items on the project and fixed before deployment. Other issues can be identified by Nessus which is configured to perform web application scan each week. Output is reviewed by the Security Team.

Perform fuzz testing

Inducing program failure by deliberately introducing malformed or random data to an application helps reveal potential security issues prior to release, while requiring modest resource investment.

When? – At sprints across the project.

How? – Once code has entered production, fuzz testing is performed by the Nessus Web Scanner on a weekly basis, but mod_security is installed and should mitigate many attempted attacks before they reach the application-level – this is not a reason to not write secure code! Manual penetration tests are commissioned by Reward Gateway on an annual basis but clients may perform their own test more often.

Conduct attack surface review

Reviewing attack surface measurement upon code completion helps ensure that any design or implementation changes to an application or system have been taken into account, and that any new attack vectors created as a result of the changes have been reviewed and mitigated, including threat models.

When? – At sprints across the project.

How? – For most Reward Gateway projects, this step is not required as the attack surface is known to be the web-facing elements on the frontend and in Reward Manager. These will be evaluated as part of the Code Review process and other tests documented in this page.

Create an Incident Response Plan

Preparing an Incident Response Plan is crucial for helping to address new threats which can emerge over time. It includes identifying appropriate security and emergency contacts, as well as establishing security servicing plans for code inherited from other groups within the organisation and for licensed third-party code.

When? – This should be conducted at some point before the project is deployed.

How? – Reward Gateway has a set of incident response plans as part of the ISO 27001 compliance. These cover the most likely situations, such as data loss and privacy breaches, and provide action plans to enact on these events.

Conduct a Final Security Review

Deliberately reviewing all security activities that were performed helps ensure software release readiness. The Final Security Review (FSR) usually includes examining threat models, tools outputs, and performance against the quality gates and bug bars defined during the Requirements Phase. The FSR results in one of three different outcomes: Passed FSR, Passed FSR with exceptions, FSR with escalation.

When? – Every sprint.

How? – Final Security Review should be conducted by the Team Leader as part of each sprint. Ultimately, the Final Security Review is part of the Code Review performed when work is merged into the main branch. Passing FSR is indicated by the work being approved by the reviewers and merged with comments being used to document any exceptions.

Certify release and archive

Certifying software prior to a release helps ensure security and privacy requirements are met. Archiving all pertinent data is essential for performing post-release servicing tasks and helps lower the long-term costs associated with sustained software engineering.

Archiving should include all specifications, source code, binaries, private symbols, threat models, documentation, and emergency response plans, as well as license and servicing terms for any third-party software.

When? – Every sprint.

How? – Source code should all be kept in the git repositories and follow the versioning flow outlined in the Development Process. Licensing and other documentation should be maintained throughout the project life cycle in Confluence.

Test Data

Production data shall not be used for testing and is held exclusively in the Production and Disaster Recovery environments. Test data in the Staging environment is based on an anonymised copy of the production data generated by scripts controlled by the DevOps Engineers.

Test data in Development environments is generated independently to all of these from scratch.

Revision history

Rev	Date	Author	Description	Approved	Date
1.0	08.06.2015	Will Tracz	Initial draft.	Richard Hurd-Wood	27.08.2015
1.1	08.03.2016	Asen Varsanov	Minor review.	Richard Hurd-Wood	23.03.2016
1.2	01.03.2017	Liam Jones	Rebrand and general clean.	Will Tracz	03.03.2017
1.3	21.09.2018	Asen Varsanov	Review and update.	Will Tracz	24.09.2018