


Project / Doc ID #	TASK JEEVES / TS-SEC-SDL
<p style="text-align: center;">SOFTWARE WARRIORS: SECURITY DEVELOPMENT LIFECYCLE (SDL)</p> 	
Classification/Distribution	Software Warriors Proprietary, Restricted Distribution
Release/Revision Date	2020-05-29
Project Name	Task Jeeves
Project Summary	Basic task manager web application
Team Members	<p>Darlene Agbayani Julian Kim Craig Opie Joseph Paragas</p> <p>Information Computer Science University of Hawaii at Manoa</p>
Distribution Authorizations, Important Notices, and Notes	
<p style="text-align: center;"><u>This document contains Software Warriors proprietary information. Distribution of this document, or any of the information contained herein, is restricted without written consent from Software Warriors Management.</u></p>	



List of Revisions

Revision	Date	Revised By	Revision Description
Origin	2020-05-29	Craig Opie	Document Created



Table of Contents

INTRODUCTION	4
PRIVACY	4
Privacy By Design	4
Privacy By Default	5
Privacy In Deployment	5
SECURITY	5
Secure By Design	5
Secure By Default	6
Secure In Deployment	6
THE SECURE SOFTWARE DEVELOPMENT LIFE CYCLE MODEL	7
Education and Awareness	7
Project Inception	8
Cost Analysis	11
Establish Best Practices For Design	12
Risk Analysis	14
Creating Documentation And Tools For Users That Address Privacy And Security	16
Establish Best Practices For Development	17
Privacy and Security Testing	18
Security Push	19
Public Release Privacy Review	21
Response Planning	22
Final Security Review	23



Software Release	27
Post Release	27

Introduction

Our team name is the “Software Warriors”. Our [GitHub Page](#) contains information about each member and our projects. Our first project is “Task Jeeves”, a web application that creates a task list specific to each user. Task Jeeves will require users to authenticate using username and password and then will authorize users to view applicable data that they own. Each task will have a title, details, and due date which can be edited by the user. The user will be able to mark a task as completed to remove the task from the active list. Development tools used for Task Jeeves includes GitHub for version management and agile project management, IntelliJ IDEA using ESLint enhanced to meet the AirBnB coding standards, and will be written in JavaScript.

Our Security Development Lifecycle (SDL) is based on the Microsoft SDL which is an industry-leading software security assurance process. Combining a holistic and practical approach, the SDL introduces security and privacy early and throughout all phases of the development process. The following documentation provides an in-depth description of our SDL methodology and requirements.

How We Handle Privacy

Privacy By Design

Minimize data collection and sensitivity. Collect the minimum amount of data that is required for a particular purpose, and use the least sensitive form of that data.

Provide notice and consent. Provide appropriate notice about data that is collected, stored, or shared so that users can make informed decisions about their personal information.

Enable user policy and control. Enable parents to manage privacy settings for their children and enterprises to manage privacy settings for their employees.



Protect the storage and transfer of data. Encrypt personally identifiable information in transfer, limit access to stored data, and ensure that data usage complies with uses communicated to the customer.

Privacy By Default

Ship with conservative default settings. Obtain appropriate consent before collecting or transferring any data. To prevent unauthorized access, protect personal data stored in access control lists.

Privacy In Deployment

Publish deployment guides. Disclose privacy mechanisms to enterprise customers so that they can establish internal privacy policies and maintain their customers' and employees' privacy.

How We Handle Security

Secure By Design

Secure architecture, design, and structure. Developers consider security issues part of the basic architectural design of software development. They review detailed designs for possible security issues and design and develop mitigations for all threats.

Threat modeling and mitigation. Threat models are created, and threat mitigations are present in all design and functional specifications.

Elimination of vulnerabilities. No known security vulnerabilities that would present a significant risk to anticipated use of the software remain in the code after review. This review includes the use of analysis and testing tools to eliminate classes of vulnerabilities.

Improvements in security. Less secure legacy protocols and code are deprecated, and, where possible, users are provided with secure alternatives that are consistent with industry standards.



Secure By Default

Least privilege. All components run with the fewest possible permissions.

Defense in depth. Components do not rely on a single threat mitigation solution that leaves customers exposed if it fails.

Conservative default settings. The development team is aware of the attack surface for the product and minimizes it in the default configuration.

Avoidance of risky default changes. Applications do not make any default changes to the operating system or security settings that reduce security for the host computer. In some cases, such as for security products, it is acceptable for a software program to strengthen (increase) security settings for the host computer. The most common violations of this principle are games that either open up firewall ports without informing the user or instruct users to do so without consideration of possible risks.

Less commonly used services off by default. If fewer than 80 percent of a program's users use a feature, that feature should not be activated by default. Measuring 80 percent usage in a product is often difficult, because programs are designed for many different personas. It can be useful to consider whether a feature addresses a core/primary use scenario for all personas. If it does, the feature is sometimes referred to as a P1 feature.

Secure In Deployment

Deployment guides. Prescriptive deployment guides outline how to deploy each feature of a program securely, including providing users with information that enables them to assess the security risk of activating non-default options (and thereby increasing the attack surface).

Analysis and management tools. Security analysis and management tools enable administrators to determine and configure the optimal security level for a software release.

Patch deployment tools. Deployment tools are provided to aid in patch deployment.



The Secure Software Development Life Cycle Model

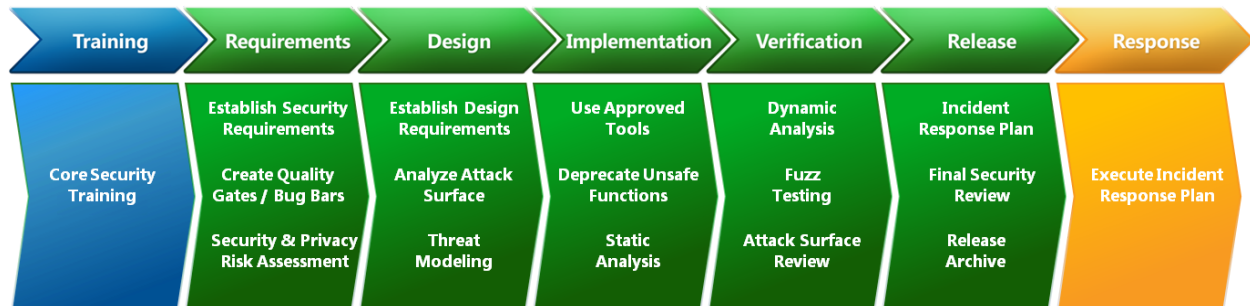


Figure 1: Basic Outline and Model

Education and Awareness

All members of software development teams should receive appropriate training to stay informed about security basics and recent trends in security and privacy. Individuals who develop software programs should attend at least one security training class each year. Security training can help ensure software is created with security and privacy in mind, and can also help development teams stay current on security issues. Project team members are strongly encouraged to seek additional security and privacy education that is appropriate to their needs and/or products.

A number of key knowledge concepts are important to successful software security. These concepts can be broadly categorized as either basic or advanced security knowledge.

Security Requirements

All developers, testers, and program managers must complete at least one security training class each year. Individuals who have not taken a class in the basics of security design, development, and testing must do so.



At least 80% of the project team staff, who work on products or services must be in compliance with the standards listed above before their product or service is released. Relevant managers must also be in compliance with these standards.

Project teams are strongly encouraged to plan security training early in the development process, so that training can be completed as early as possible and have a maximum positive effect on the project's security.

Project Inception

The need to consider security and privacy at a foundational level is a fundamental tenet of system development. The best opportunity to build trusted software is during the initial planning stages of a new release or a new version, because development teams can identify key objects and integrate security and privacy, which minimizes disruption to plans and schedules.

Privacy Requirements

Identify the privacy advisor who will serve as your team's first point of contact for privacy support and additional resources.

Identify who on the team is responsible for privacy for a project. This person is typically called the privacy lead or, sometimes, the privacy champion.

Define and document the project's privacy bug bar (see the preceding "Security Requirements" section).

Security Requirements

Develop and answer a short questionnaire to verify whether your development team is subject to Security Development Lifecycle (SDL) policies. The questionnaire has two possible outcomes:

1. If the project is subject to SDL policies, it must be assigned a security advisor who serves as the point of contact for its Final Security Review (FSR). It is in the project team's interest to register promptly and establish the security requirements for which they will be held accountable. The team will also be asked some



technical questions as part of a security risk assessment to help the security advisor identify potential security risks.

2. If the project is not subject to SDL policies, it is not necessary to assign a security advisor and the release will be classified as exempt from SDL security requirements.

Identify the team or individual that is responsible for tracking and managing security for the product. This team or individual does not have sole responsibility for ensuring that a software release is secure, but the team or individual is responsible for coordinating and communicating the status of any security issues. In smaller product groups, a single program manager might take this role.

Ensure that bug reporting tools can track security bugs and that a database can be queried dynamically for all security bugs at any time. The purpose of this query is to examine unfixed security bugs in the FSR. The project's bug tracking system must accommodate the bug bar ranking value recorded with each bug.

Define and document a project's security bug bar. This set of criteria establishes a minimum level of quality. Defining it at the start of a project improves understanding of risks associated with security issues and enables teams to identify and fix security bugs during development. The project team must negotiate a bug bar approved by the security advisor with project-specific clarifications and (as appropriate) more stringent security requirements specified by the security advisor. The bug bar must never be relaxed, though, even as a project's release date nears.

Cost Analysis

Before you invest time in design and implementation, it is important to understand the costs and requirements involved in handling data with privacy considerations. Privacy risks increase development and support costs, so improving security and privacy can be consistent with business needs.



Privacy Requirements

Complete the "Initial Assessment" section of "SDL Privacy Questionnaire". An initial assessment is a quick way to determine a project's Privacy Impact Rating and estimate how much work is necessary to comply with Privacy Guidelines for Developing Software Products and Services.

The Privacy Impact Rating (P1, P2, or P3) measures the sensitivity of the data your software will process from a privacy point of view:

P1 – High Privacy Risk: The feature, product, or service stores or transfers Personally Identifiable Information (PII) or error reports; monitors the user with an ongoing transfer of anonymous data; changes settings or file type associations; or installs software.

P2 – Moderate Privacy Risk: The sole behavior that affects privacy in the feature, product, or service, is a one-time user-initiated anonymous data transfer (e.g., the user clicks on a link and goes out to a website).

P3 – Low Privacy Risk: No behaviors exist within the feature, product, or service that affect privacy. No anonymous or personal data is transferred, no PII is stored on the machine, no settings are changed on the user's behalf, and no software is installed.

Product teams must complete only the work that is relevant to their Privacy Impact Rating. Complete the initial assessment early in the product planning/requirements phase, before you write detailed specifications or code.

Security Requirements

A security risk assessment (SRA) is a mandatory exercise to identify functional aspects of the software that might require deep security review. Given that program features and intended functionality might be different from project to project, it is wise to start with a simple SRA and expand it as necessary to meet the project scope.

Such assessments must include the following information:

1. What portions of the project will require threat models before release.
2. What portions of the project will require security design reviews before release.



3. What portions of the project will require penetration testing (pen testing) by a mutually agreed upon group that is external to the project team. Any portion of the project that requires pen testing must resolve issues identified during open testing before it is approved for release.
4. Any additional testing or analysis requirements the security advisor deems necessary to mitigate security risks.
5. Clarification of the specific scope of fuzz testing requirements. ("Privacy and Security Testing" discusses fuzz testing.)

Establish Best Practices For Design

The best time to influence a project's trustworthy design is early in its lifecycle. Functional specifications may need to describe security features or privacy features that will be directly exposed to users, such as requiring user authentication to access specific data or user consent before use of a high-risk privacy feature. Design specifications should describe how to implement these features, as well as how to implement all functionality as secure features. We define secure features as features whose functionality is well engineered with respect to security, such as rigorously validating all data before processing it or cryptographically robust use of cryptographic APIs. It is important to consider security and privacy concerns carefully and early when you design features, and avoid attempts to add security and privacy near the end of a project's development.

Privacy Requirements

If your project has a privacy impact rating of P1, identify a compliant design based on the concepts, scenarios, and rules in the Privacy Guidelines for Developing Software Products and Services. Definitions of privacy rankings (P1, P2, P3) can be found in Cost Analysis. You can find additional guidance in "SDL Generic Privacy Questionnaire".

Security Requirements

The best time to influence a project's trustworthy design is early in its lifecycle. Functional specifications may need to describe security features or privacy features that will be directly exposed to users, such as requiring user authentication to access specific



data or user consent before use of a high-risk privacy feature. Design specifications should describe how to implement these features, as well as how to implement all functionality as secure features. We define secure features as features whose functionality is well engineered with respect to security, such as rigorously validating all data before processing it or cryptographically robust use of cryptographic APIs. It is important to consider security and privacy concerns carefully and early when you design features, and avoid attempts to add security and privacy near the end of a project's development.

Complete a security design review with a security advisor for any project or portion of a project that requires one. Some low-risk components might not require a detailed security design review.

- Satisfy minimal cryptographic design requirements.
- When developing with managed code, use strong-named assemblies and request minimal permission. When using strong-named assemblies, do not use APTCA (Allow Partially Trusted Caller Attribute) unless the assembly was approved after a security review. Without specific security review and approval, assemblies that use APTCA will generate FxCop errors and will fail to pass a Final Security Review (FSR).
- Be logical and consistent when you make firewall exceptions. Any product or component that requires changes to the host firewall settings must adhere to the requirements that are outlined in "A Policy for Managing Firewall Configurations."

Risk Analysis

During the design phase of development, carefully review security and privacy requirements and expectations to identify security concerns and privacy risks. It is efficient to identify and address these concerns and risks during the design phase. For security concerns, threat modeling is a systematic process that is used to identify threats and vulnerabilities in software. You must complete threat modeling during project design. A team cannot build secure software unless it understands the assets the project is trying to protect, the threats and vulnerabilities introduced by the project, and details of how the project will mitigate those threats. Important risk analysis considerations include the following:



- Threats and vulnerabilities that exist in the project's environment or that result from interaction with other systems. You cannot consider the design phase complete unless you have a threat model or models that include such considerations. Threat models are critical components of the design phase and reference a project's functional and design specifications to describe vulnerabilities and mitigations.
- Code that was created by external development groups in either source or object form. It is very important to evaluate carefully any code from sources external to your team. Failure to do so might cause security vulnerabilities about which the project team is unaware.
- Threat models that include all legacy code if the project is a new release of an existing program. Such code could have been written before much was known about software security, and therefore could contain vulnerabilities.
- A Review of the design of high-risk (P1) privacy projects with a privacy subject matter expert and, if necessary, with appropriate legal counsel conducted as soon as possible in the project. Definitions of privacy rankings (P1, P2, P3) can be found in Cost Analysis.
- A detailed privacy analysis to document your project's key privacy aspects. Important issues to consider include:
 - What personal data is collected
 - What is the compelling customer value proposition and business justification
 - What notice and consent experiences are provided
 - What controls are provided to users and enterprises
 - How is unauthorized access to personal information prevented

Privacy Requirements

If a project has a privacy impact rating of P1:

- Complete the "Detailed Privacy Analysis" section in "SDL Privacy Questionnaire". The questions will be customized to the behaviors specified in the initial assessment.
- Hold a design review with your privacy subject matter expert.

If your project has a privacy impact rating of P2:



- Complete the "Detailed Privacy Analysis" section in "SDL Privacy Questionnaire". The questions will be customized to the behaviors specified in the initial assessment.
- Hold a design review with your privacy subject matter expert only if one or more of these criteria apply:
 - The privacy subject matter expert requests a design review.
 - You want confirmation that the design is compliant.
 - You wish to request an exception.

If your project has a privacy impact rating of P3, there are no privacy requirements during this phase.

Security Requirements

Complete threat models for all functionality identified during the cost analysis phase. Threat models typically must consider the following areas:

- All projects. All code exposed on the attack surface and all code written by or licensed from a third party.
- New projects. All features and functionality.
- Updated versions of existing projects. New features or functionality added in the updated version.

Ensure that all threat models meet minimal threat model quality requirements. All threat models must contain data flow diagrams, assets, vulnerabilities, and mitigation. Threat modeling can be done in a variety of ways using either tools or documentation/specifications to define the approach. For assistance in creating threat models, see "Risk Analysis" in The Security Development Lifecycle book or consult other guidance listed in the following "Resources" section.

Have all threat models and referenced mitigations reviewed and approved by at least one developer, one tester, and one program manager. Ask architects, developers, testers, program managers, and others who understand the software to contribute to threat models and to review them. Solicit broad input and reviews to ensure the threat models are as comprehensive as possible.



Threat model data and associated documentation (functional/design specs) have been stored using the document control system used by the product team.

Creating Documentation And Tools For Users That Address Privacy And Security

Every release of a software program should be secure by design, in its default configuration, and in deployment. However, people use programs differently, and not everyone uses a program in its default configuration. You need to provide users with enough security information so they can make informed decisions about how to deploy a program securely. Because security and usability might conflict, you also need to educate users about the threats that exist as well as the balance between risk and functionality when deciding how to deploy and operate software programs.

It is difficult to discuss specific security documentation needs before development plans and functional specifications stabilize. As soon as the architecture is reasonably stable, the user education team can develop a security documentation plan and schedule. Delivering documentation about how to use a software program securely is just as important as delivering the program itself.

Establish Best Practices For Development

To detect and remove security issues early in the development cycle, it helps to establish, communicate, and follow effective practices for developing secure code. A number of resources, tools, and processes are available to help you accomplish this goal. Investing time and effort to apply effective practices early will help you eliminate security issues and avoid having to respond to them later in the development cycle, or even after release, which is expensive.

Privacy Requirements

Establish and document development best practices for the development team. Communicate any design changes that affect privacy to your team's privacy lead so that they can document and review any changes.



Security Requirements

Build tools. Use the currently required (or later) versions of compilers to compile options for the Win32, Win64, WinCE and Macintosh target platforms, as listed in "SDL Required and Recommended Compilers, Tools, and Options for All Platforms".

- Compile C/C++ code with /GS or approved alternative on other platforms
- Link C/C++ code with /SAFESEH or approved alternative on other platforms
- Link C/C++ code with /NXCOMPAT (for more information, refer to "Appendix F: SDL Requirement: No Executable Pages") or approved alternative on other platforms
- Use MIDL with /robust or approved alternative on other platforms

Code analysis tools. Use the currently required (or later) versions of code analysis tools for either native C and C++ or managed (C#) code that are available for the target platforms, as listed in "Required and Recommended Compilers, Tools, and Options for All Platforms".

Banned Application Programming Interfaces (APIs). New native C and C++ code must not use banned versions of string buffer handling functions. Based on analysis of previous Microsoft Security Response Center (MSRC) cases, avoiding use of banned APIs is one actionable way to remove many vulnerabilities.

No writable shared PE sections. Sections marked as shared in shipping binaries represent a security threat. Use properly secured dynamically created shared memory objects instead. See "SDL Requirement: Shared Section".

Privacy And Security Testing

Security testing addresses two broad areas of concern:

- Confidentiality, integrity, and availability of the software and data processed by the software. This area includes all features and functionality designed to mitigate threats as described in the threat model.
- Freedom from issues that could result in security vulnerabilities. For example, a buffer overrun in code that parses data could be exploited in ways that have security implications.



Begin security testing very soon after the code is written. This testing stage requires one full test pass after the verification stage because potential issues and vulnerabilities might change during development. Security testing is important to the Security Development Lifecycle. As Michael Howard and David LeBlanc write in *Writing Secure Code, Second Edition*: “The designers and the specifications might outline a secure design, the developers might be diligent and write secure code, but it’s the testing process that determines whether the product is secure in the real world.”

Security Requirements

Create and complete security testing plans that address these issues:

- Security features and functionality work as specified. Ensure that all security features and functionality that are designed to mitigate threats perform as expected.
- Security features and functionality cannot be circumvented. If a mitigation can be bypassed, an attacker can try to exploit software weaknesses, rendering security features and functionality useless.
- Ensure general software quality in areas that can result in security vulnerabilities. Validating all data input and parsing code against malformed or unexpected data is a common way attackers try to exploit software. Data fuzzing is a general testing technique that can help prevent such attacks.

Penetration testing. Use the threat models to determine priorities, test, and attack the software as a hacker might. Use existing tools or design new tools if needed.

- Hire third-party security firms as appropriate. Depending on the business goals for your project and availability of resources, consider engaging an external security firm for a security review and/or penetration testing.

Develop and use vulnerability regression tests. If the code has ever had a security vulnerability reported, we strongly suggest that you add regression tests to the test suite for that component to ensure that similar vulnerabilities are not inadvertently re-introduced to the code. Similarly, if there are other products with similar functionality in the market that have suffered publicly reported vulnerabilities, add tests to the test plan to prevent similar vulnerabilities.



Security Push

A security push is a team-wide focus on threat model updates, code review, testing, and thorough documentation review and edit. A security push is not a substitute for a lack of security discipline. Rather, it is an organized effort to uncover changes that might have occurred during development, improve security in any legacy code, and identify and remediate any remaining vulnerabilities. However, it should be noted that it is not possible to build security into software with only a security push.

A security push occurs after a product has entered the verification stage (reached code/feature complete). It usually begins at about the time Beta testing starts. Because the results of the security push might alter the default configuration and behavior of a product, you should perform a final Beta test review after the security push is complete and after all bugs and required changes are resolved.

It is important to note that the goal of a security push is to find bugs, not to fix them. The time to fix bugs is after you complete the security push.

Privacy Requirements

Review and update the SDL Privacy Questionnaire form for any material privacy changes that were made during the implementation and verification stages. Material changes include:

- Changing the style of consent
- Substantively changing the language of a notice
- Collecting different data types
- Exhibiting new behavior

Security Requirements

Review and update threat models. Examine the threat models that were created during the design phase. If circumstances prevent creation of threat models during the design phase, you must develop them in the earliest phase of the security push.



Review all bugs that affect security against the security bug bar. Ensure that all security bugs contain the security bug bar rating.

Push Preparation

A successful push requires planning:

- You should allocate time and resources for the push in your project's schedule, before you begin development. To rush the security push will cause problems or delays during the final security review.
- Your team's security coordinator should determine what resources are required, organize a security push leadership team, and create the needed supporting materials and resources
- The security representative should determine how to communicate security push information to the rest of the team. It is helpful to establish a central intranet location for all information related to the push, including news, schedules, plans, forms and documents, white papers, training schedules, and links. The intranet site should link to internal resources that help the group execute the security push. This site should serve as the primary source of information, answers, and news for employees during the push.
- There must be well-defined criteria to determine when the push is complete.

Your team will need training before the push. At a minimum, this training should help team members understand the intent and logistics of the push itself. Some members might also require updated security training, as well as training in security or analysis techniques that are specific to the software that is undergoing the push. The training should have two components: the push logistics, delivered by a senior member of the team conducting the push, and technical and role-specific security training.

Push Duration

The amount of time, energy, and team-wide focus that a security push requires will differ depending on the status of the code base and the amount of attention the team has given to security earlier in development. A security push will require less time if your team has:

- Rigorously kept all threat models up to date.
- Actively and completely subjected those threat models to penetrations testing.



- Accurately tracked and documented attack surfaces and any changes made to them.
- Completed security code reviews for all high-priority code (see discussion later in this section for details about how priority is assessed).
- Identified and documented development and testing contacts for all code released with the product.
- Rigorously brought all legacy code up to current security standards.
- Validated the security documentation plan.

Public Release Privacy Review

Before any public release (including Alpha and Beta test releases), update the appropriate SDL Generic Privacy Questionnaire for any significant privacy changes that were made during implementation verification. Significant changes include changing the style of consent, substantively changing the language of a notice, collecting different data types, and exhibiting new behavior.

Although privacy requirements must be addressed before any public release of code, security requirements need not be addressed before public release. However, you must complete a final security review before final release.

Privacy Requirements

Review and update the Privacy Companion form.

For a P1 project, your privacy advisor will review your final SDL Privacy Questionnaire, help determine whether a privacy disclosure statement is required, and give final privacy approval for public release.

For a P2 project, you need validation by a privacy advisor if any of the following is true:

- A design review is requested by a privacy advisor.
- You want confirmation that the design is compliant with privacy standards.
- You wish to request an exception.

For a P3 project, there are no additional privacy requirements.



Complete the privacy disclosure.

Draft a privacy disclosure statement as advised by the privacy advisor. If your privacy advisor indicates that a privacy disclosure is waived or covered, you do not need to meet this requirement.

Work with your privacy advisor and legal representatives to create an approved privacy disclosure.

Post the privacy disclosure to the appropriate Web site before each public release.

Response Planning

Any software can be released with unknown security issues or privacy issues, despite best efforts and intentions. Even programs with no known vulnerabilities at the time of release can be subject to new threats that emerge and that might require action. Similarly, privacy advocates might raise privacy concerns after release. You must prepare before release to respond to potential security and privacy incidents. With proper planning, you should be able to address many of the incidents that could occur in the course of normal business operations.

Your team must be prepared for a zero-day exploit of a vulnerability—one for which a security update does not exist. Your team must also be prepared to respond to a software security emergency. If you create an emergency response plan before release, you will save time, money and frustration when an emergency response is required for either security or privacy reasons.

Privacy Requirements

For P1 and P2 projects, identify the person who will be responsible for responding to all privacy incidents that may occur. Add this person's e-mail address to the "Incident Response" section of the SDL Privacy Questionnaire. If this person changes positions or leaves the team, identify a new contact and update all SDL Generic Privacy



Questionnaire forms for which that person was listed as the privacy incident response lead.

Identify additional development and quality assurance resources on the project team to work on privacy incident response issues. The privacy incident response lead is responsible for defining these resources in the "Incident Response" section of the SDL Generic Privacy Questionnaire.

After release, if a privacy incident occurs you must be prepared to follow the SDL Privacy Escalation Response Framework, which might include risk assessment, detailed diagnosis, short-term and long-term action planning, and implementation of action plans. Your response might include creating a patch, replying to media inquiries, and reaching out to influential external contacts.

Security Requirements

The project team must provide contact information for people who respond to security incidents. Typically, such responses are handled differently for products and services.

Provide information about which existing sustained engineering (SE) team has agreed to be responsible for security incident response for the project. If the product does not have an identified SE team, they must provide an emergency response plan (ERP) and provide it to the incident response team. This plan must include contact information for 3-5 engineering resources, 3-5 marketing resources, and 1-2 management resources who will be the first points of contact when you need to mobilize your team for a response effort. Someone must be available 24 hours a day, 7 days a week – and contacts must understand their roles and responsibilities and be able to execute on them when necessary.

Identify someone who will be responsible for security servicing. All code developed outside the project team (third-party components) must be listed by filename, version, and source (where it came from).



You must have an effective security response process for servicing code that has been inherited or reused from other teams. If that code has a vulnerability, the releasing team may have to release a security update even though it did not develop the code.

You must also have an effective security response process for servicing code that has been licensed from third parties in either object or source form. For licensed code, you also need to consider contractual requirements regarding which party has rights and obligations to make modifications, associated service level agreements, and redistribution rights for any security modifications.

Create a documented sustaining model that addresses the need to release immediate patches in response to security vulnerabilities and does not depend entirely on infrequent service packs.

Develop a consistent and comprehensible policy for security response for components that are released outside of the regular product release schedule (out of band) but that can be used to update or enhance the software after release.

Final Security Review

As the end of your software development project approaches, you need to be sure that the software is secure enough to ship. The final security review (FSR) will help determine this. The security team assigned to the project should perform the FSR with help from the product team to ensure that the software complies with all SDL requirements and any additional security requirements identified by the security team (such as penetration testing or additional fuzz testing).

A final security review can last anywhere from a few days to six weeks, depending on the number of issues and the team's ability to make necessary changes. It is important to schedule the FSR carefully—that is, you need to allow enough time to address any serious issues that might be found during the review. You also need to allow enough time for a thorough analysis; insufficient time could cause you to make significant changes after the FSR is completed.



Privacy Requirements

Repeat the privacy review for any open issues that were identified in the pre-release privacy review, or for material changes made to the product after the pre-release privacy review. Material changes include modifying the style of consent, substantively revising the language of a notice, collecting different data types, or exhibiting new behavior. If no material changes were made, no additional reviews or approvals are required.

After the privacy review is finished, your privacy advisor will either sign off on the product as-is or will provide a list of required changes.

Security Requirements

The project team must provide all required information before the scheduled FSR start date. Failure to do so will delay completion of the FSR. If the schedule slips significantly before the FSR begins, contact the assigned security advisor to reschedule.

After the FSR is finished, the security advisor will either sign off on the project as-is or provide a list of required changes.

The Final Review Process

Define a due date for all project information that is required to start the FSR. To minimize the likelihood of unexpected delays, plan to conduct an FSR 4-6 weeks before Release to Manufacture (RTM) or Release to Web (RTW). Your team might need to revalidate specific decisions or change code to fix security issues. The team must understand that additional security work will need to be performed during the FSR.

The FSR cannot begin until you have completed the reviews of the security milestones that were required during development. Milestones include in-depth bug reviews, threat model reviews, and running all SDL-mandated tools.

Reconvene the development and security leadership teams to review and respond to the questions posed during the FSR process.



Review threat models. The security advisor should review the threat models to ensure that all known threats and vulnerabilities are identified and mitigated. Have complete and up-to-date threat models at the time of the review.

Review security bugs that were deferred or rejected for the current release. The bug review should ensure that a consistent, minimum security standard was adhered to throughout the development cycle. Teams should already have reviewed all security bugs against the criteria that were established for release. If the release does not have a defined security bug bar, your team can use the standard SDL Security Bug Bar.

Validate results of all security tools. You should have run these tools before the FSR, but a security advisor might recommend that you also run other tools. If tool results are inaccurate or unacceptable, you might need to rerun some tools.

Ensure that you have done all you can to remove bugs that meet your organization's severity criteria, so that there are no known vulnerabilities. Ultimately, the goal of SDL is to remove security vulnerabilities from products and services. No software release can pass an FSR with known vulnerabilities that would be considered as Sev 1, Sev 2, or Sev 3.

Submit exception requests to a security advisor for review. If your team cannot meet a specific SDL requirement, you must request an exception. Typically, such a request is made well in advance of the FSR. A security advisor will review these requests and, if the overall security risk is tolerable, might choose to grant the exception. If the security risk is not acceptable, the security advisor will deny the exception request. It is best to address all exception requests as soon as possible in the development phase of the project.

Possible FSR Outcomes

Possible outcomes of an FSR include:

Passed FSR. If all issues identified during the FSR are corrected before RTM/RTW, the security advisor should certify that the project has successfully met all SDL requirements.



Passed FSR (with exceptions). If all issues identified during the FSR are corrected before RTM/RTW OR the security advisor and team can reach an acceptable compromise about any SDL requirements that the project team was unable to resolve. The security advisor should identify the exceptions and certify that all other aspects of the project have successfully met all SDL requirements.

- All exceptions and security issues not addressed in the current release should be logged, and then addressed and corrected in the next release.

FSR escalation. If a team does not meet all SDL requirements and the security advisor and the product team cannot reach an acceptable compromise, the security advisor cannot approve the project, and the project cannot be released. Teams must either correct whatever SDL requirements that they cannot correct or escalate to higher management for a decision.

Escalations occur when the security advisor determines that a team cannot meet the defined requirements or is in violation of an SDL requirement. Typically, the team has a business justification that prevents them from being compliant with the requirement. In such instances, the security advisor and the team should work together to compose a consolidated escalation report that outlines the issue—including a description of the security or privacy risk and the rationale behind the escalation. This information is typically provided to the business unit executive and the executive with corporate responsibility for security and privacy, to aid decision making.

If a team fails to follow proper FSR procedures - either by an error of omission or by willful neglect - the result will be an immediate FSR failure. Examples include:

- Errors of omission, such as failure to properly document all required information
- Specious claims and willful neglect, including:
 - Claims of “Not subject to SDL” contrary to evidence
 - Claims of “FSR pass” contrary to evidence, and software RTM/RTW without the appropriate signoff Such an incident can result in very serious consequences subsequently, and is always immediately escalated to the project team executive staff and the executive in charge of security and privacy.



Software Release

Software release to manufacturing (RTM) or to the Web (RTW) is conditional to completion of the Security Development Lifecycle process as defined in this document. The security advisor assigned to the release must certify that your team has satisfied security requirements. Similarly, for all products that have at least one component with a privacy impact rating of P1, your privacy advisor must certify that your team has satisfied the privacy requirements before the software can be shipped.

Privacy Requirements

Design and implement a sign-off process to ensure privacy and other policy compliance before you ship. This process should include explicit acknowledgement that the product successfully passed the FSR and was approved for release.

Security Requirements

To facilitate the debugging of security vulnerability reports and to help tools teams research cases in which automated tools failed to identify security vulnerabilities, all product teams must submit symbols for all publicly released products as part of the release process. This requirement is needed only for RTM/RTW binaries and any post-release binaries that are publicly released to customers.

Design and implement a sign-off process to ensure security and other policy compliance before you ship. This process should include explicit acknowledgement that the product successfully passed the FSR and was approved for release.

Post Release

After a software program is released, the product development team must be available to respond to any possible security vulnerabilities or privacy issues that warrant a response. In addition, develop a response plan that includes preparations for potential post-release issues.