

SECURITY COMPASS WHITEPAPER 2017

6 Steps to Getting Started with Software Security Requirements

INTRODUCTION

There are many benefits to implementing a Software Security Requirements program, including:

- Lowering costs to build secure software
- Making security measurable
- Turning unplanned work into planned work
- Freeing up time away from remediation, and into feature development
- Having a single process that works with in-house, outsourced, and commercial software
- Providing confidence that software is secure, when requirements are linked to verification

This guide provides 6 simple steps to get you started on building a Software Security Requirements program.

The steps are as follows:

1. Define your goals for adopting such a program.
2. Select a repository for your re-usable requirements.
3. Determine the sources for your requirements.
4. Add requirements to your repository.
5. Build a list of application properties.
6. Deploy requirements to development teams.

1. DEFINE YOUR GOALS FOR ADOPTING A SOFTWARE SECURITY REQUIREMENTS PROGRAM

Here are some common ones:

- **Lower application security risk:** You view application security as a key area of risk, and see software security requirements as a tool to help reduce that risk.
- **Lower cost of application security remediation:** You want to specify security requirements upfront, thereby reducing the need to fix vulnerabilities after they've been identified.
- **Improve compliance:** You wish to use software security requirements to improve compliance to and auditability of a particular regulation.
- **Understand application risks:** You wish to understand the types of risks affecting applications, including risks that existing assessment techniques may be unable to uncover.
- **Disseminate guidance across the organization:** You wish to centrally manage software security and other non-functional requirements and automatically push them to other teams or into other organizational processes.
- **Increase security of 3rd party developed software:** You wish to generate detailed technical security requirements for 3rd party software suppliers.

2. SELECT A REPOSITORY FOR YOUR RE-USABLE REQUIREMENTS

A [static document](#) is not sufficient for managing software security requirements. Some sort of filtering mechanism, such as a priority, is essential for getting adoption from time-crunched developers.

Here are a few common options:

Repository Type	Pros	Cons
Spreadsheet	<ul style="list-style-type: none">• Cheap• Easy to get started	<ul style="list-style-type: none">• Low fidelity• Not centralized• Hard to maintain• No integration
Sharepoint or Internally Developed Application	<ul style="list-style-type: none">• More advanced than spreadsheet• Possibility of integration with custom development	<ul style="list-style-type: none">• Hard to maintain• May be expensive to develop
Commercial Software Security Requirements solution	<ul style="list-style-type: none">• Extensive features• Integration with development & security tools• Access to continuous and up-to-date requirements from vendor's researchers• Integrated training	<ul style="list-style-type: none">• Requires securing budget & buy-in from other stakeholders

3. DETERMINE THE SOURCES FOR YOUR REQUIREMENTS

Here are a few examples:

- **Compliance regulations:** If compliance to standards such as PCI DSS is driving your software security requirements gathering program, then you should reference those compliance initiatives and include any code-level issues in your list of requirements.
- **Internal corporate standards:** You may have some existing corporate standards for areas like password management and encryption, so you'll want to at least reference these in your requirements library.
- **Secure development practices:** Your internal development teams may have a document with code samples to address well-known weaknesses. You may want to augment these samples and best practices with the information available online:
 - [OWASP Secure Coding Practices – Quick Reference Guide](#): A PDF/Word doc of concise secure coding practices that can easily be used as requirements.
 - [OWASP Application Security Verification Standard \(ASVS\)](#): A large set of verification requirements for web applications. While ASVS standards are not requirements, you can generally reverse engineer the corresponding requirement from each verification standard.
 - [Common Weakness Enumeration \(CWE\)](#): The most comprehensive set of software security weaknesses available. You will need to invest a significant amount of time if you wish to cover the breadth of the CWE. Also, CWE weaknesses do not necessarily cover countermeasures, so you will need to determine the countermeasure for each weakness yourself.

If you elect to use a commercial Software Security Requirements solution, the vendor is responsible for monitoring external sources of security requirements. A mature Software Security Requirements solution will allow you to supplement its list of requirements with your own corporate standards and code samples.

4. ADD REQUIREMENTS TO YOUR REPOSITORY

Based on the sources you selected from step 3, build a re-usable repository of software security requirements.

Include the following information for each requirement:

- **Requirement title:** Title to describe the requirement.
- **Requirement description:** A short abstract of what the requirement is. Remember developers are often time-crunched, so keep the description short and link to more detailed information if you need to.
- **Vulnerability description:** A brief description of what vulnerability the requirement is mitigating, so that developers know why they need to perform this action.
- **Base risk or priority:** A number to help teams understand how urgent the requirement generally is.
- **Verification:** how can somebody verify that this requirement has been implemented?
- **Inclusion Rule:** When is this requirement relevant to an application? What application properties need to be true? These rules are best implemented using boolean logic.

For example, a requirement which protects against certain database attacks “Bind variables in SQL statements” might apply to all applications where the application property “Uses SQL database == TRUE”. Keep track of all of the properties you reference in these rules.

- In Excel, the rules can be made up of simple formulas that reference the sheet containing properties.

The screenshot shows two Excel spreadsheets. The top spreadsheet has columns A-F and rows 1-2. Row 1 headers are: Requirement Title, Requirement Description, Vulnerability Description, Priority, Verification, Rule. Row 2 data is: Bind variables in SQL statements, Most persistence framework SQL Injection, 10, Ensure that all, and a formula =IF(Properties!B2=TRUE,TRUE,FALSE). The bottom spreadsheet has columns A-B and rows 1-2. Row 1 headers are: Property, Status. Row 2 data is: Uses SQL Database?, TRUE. A blue arrow points from the formula cell in the top spreadsheet to the TRUE cell in the bottom spreadsheet.

	A	B	C	D	E	F
1	Requirement Title	Requirement Description	Vulnerability Description	Priority	Verification	Rule
2	Bind variables in SQL statements	Most persistence framework SQL Injection		10	Ensure that all	=IF(Properties!B2=TRUE,TRUE,FALSE)

	A	B
1	Property	Status
2	Uses SQL Database?	TRUE

- In a custom app or SharePoint site, the rules can be configured using hard-coded logic.
- In a commercial Security Requirements Management solution, requirements should already be populated and updated by the vendors. You will have the ability to overwrite rules if you wish to customize them.

5. BUILD A LIST OF APPLICATION PROPERTIES

These properties should be based on those identified in step 4 (i.e. “Uses SQL database” in the example above).

Some common properties are:

- Application type (e.g. web application, mobile application, etc.)
- Uses SQL database
- Exposes RESTful web services
- Exposes SOAP web services
- Uses passwords for authentication
- Stores / transmits / processes credit card data

With a set of application properties and requirements tied to these properties you can generate a tailored list of software security requirements for an application.

6. DEPLOY REQUIREMENTS TO DEVELOPMENT TEAMS

Provide the development teams with the requirements tool, from which they should be able to generate a specific set of software security requirements that applies to their applications. They can then add these requirements to their standard requirements gathering process. A recommended final step is to collect feedback and evaluate against the goals you outlined in step 1.

ABOUT SD ELEMENTS

Our award-winning policy-to-execution platform, SD Elements, addresses applications' security and compliance requirements during the early stages of development. Powered by Ph.D. researchers, this automation platform translates policies into actionable tasks, which can then be used by enterprise IT and Engineering teams to meet security and compliance objectives. SD Elements allows your organization to efficiently deliver technology that's secure by design.