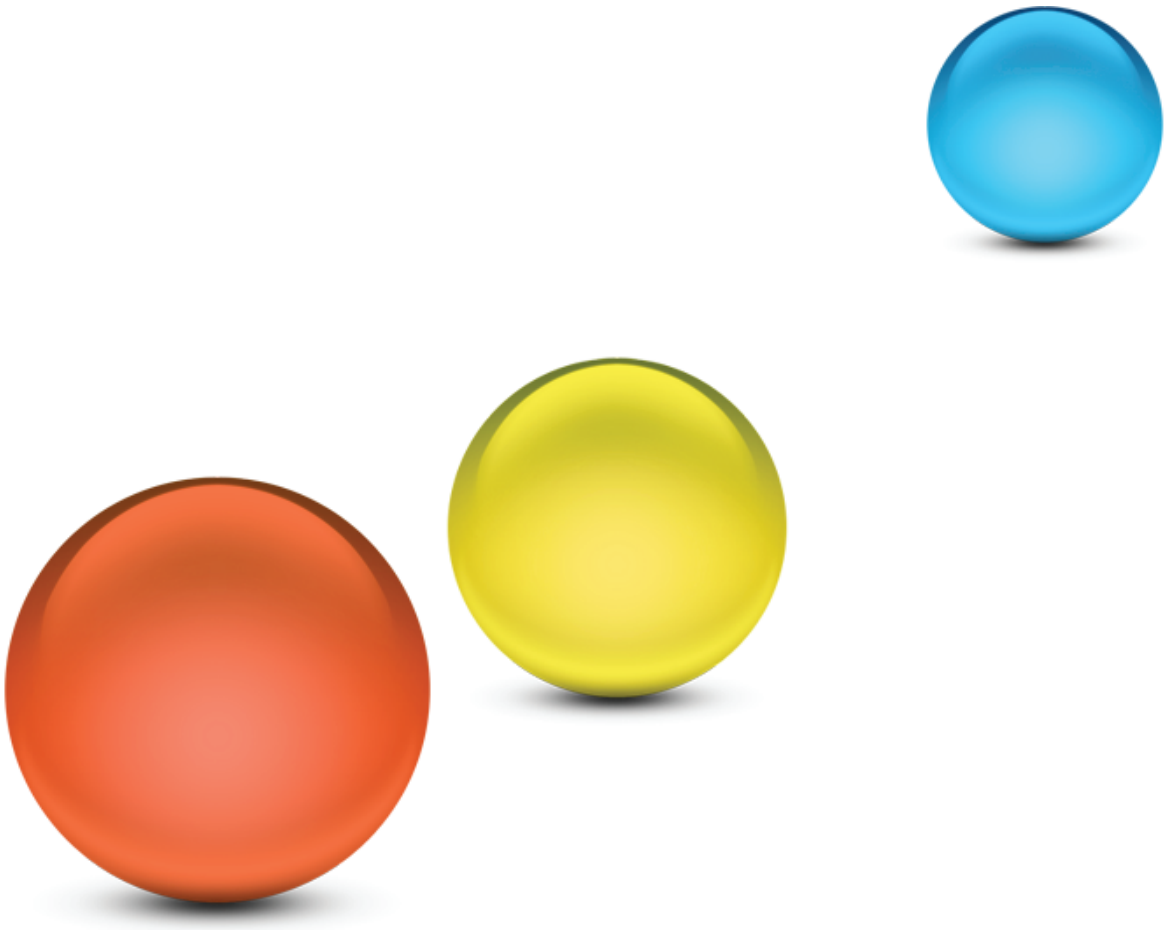
SD ELEMENTS

A product of Security Compass

• Advisory • SD Elements • Assessments • Training



# 5 Steps To Get Started With Software Security Requirements

There are many benefits to 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 5 simple steps to get you started on building a software security requirements program.

## 1. Define your goals for adopting a security requirements program.

Here are some common ones:

- **Lower application security risk:** You view application security as a key area of risk, and see security requirements as a tool to help reduce that risk.
- **Lower cost of application security remediation:** You want to specify security requirements up-front, thereby reducing the need to fix vulnerabilities after they've been identified.
- **Improve compliance:** You wish to use 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 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.

## 2a. Select repository for your re-usable requirements.

*(spreadsheet, Sharepoint/internally developed web app or a commercial Secure Application Lifecycle Management (SALM) system)*

Remember that a static document is not sufficient. Some sort of filtering mechanism, such as a priority, is essential for getting adoption from time-crunched developers.

| Repository Type | Pros | Cons |
|---|---|---|
| Spreadsheet | • Cheap<br>• Easy to get started | • Low fidelity<br>• Not centralized<br>• Hard to maintain<br>• No integration |
| SharePoint / internal app | • More advanced than website<br>• Possibility of integration with custom development | • Hard to maintain<br>• May be expensive to develop |
| Secure Application Lifecycle Management solution | • Extensive features<br>• Integration with development & security tools<br>• Access to continuous and up-to-date requirements from vendor's researchers embedded training | • Requires securing budget & buy-in from other stakeholders |

## 2b. Determine 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:
    i. OWASP Secure Coding Practices – Quick Reference Guide: A PDF/Word doc of concise secure coding practices that can easily be used as requirements.
    ii. 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.
    iii. 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 Secure Application Lifecycle Management (SALM) solution, the vendor is responsible for monitoring external sources of security requirements. A mature SALM solution will allow you to supplement its list of requirements with your own corporate standards and code samples.

## 3. Add requirements to your repository.

*Based on the sources you selected from step 2, build a re-usable repository of software security requirements.*

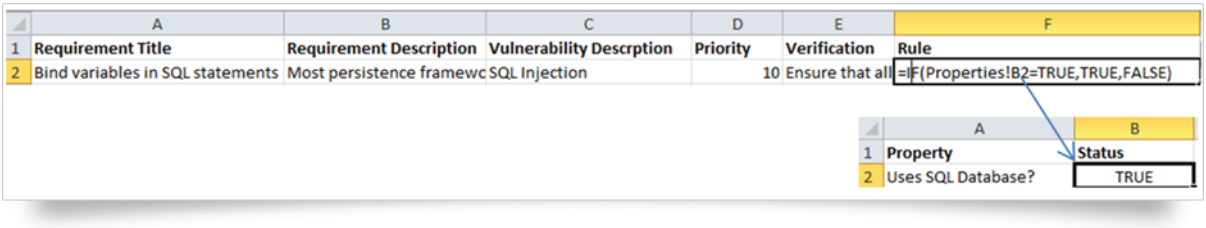
Include the following information for each requirement:

- **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.




- In a custom app or SharePoint site, the rules can be configured using hard-coded logic.
- In a commercial SALM 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.

## 4. Build a list of application properties.

*These properties should be based on those identified in step 3 (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 security requirements for an application.

## 5. Deploy requirements to development teams.

Provide the teams with the requirements tool, from which they should be able to generate a specific set of requirements that applies to them. They can then add these requirements to their standard requirements gathering process. Collect feedback and evaluate against the goals you outlined in step 1.

## About SD Elements

SD Elements is your guide for secure software development. Be more proactive with automated requirements generation that scales quickly. Make security measurable with clear links between requirements & test.  Proactively eliminate up to 97% of application security risks by building more secure software from the start.

Learn more at www.sdelements.com


A product of Security Compass

• Advisory • SD Elements • Assessments • Training