Value Of Talent. Delivered

SaM Solutions GmbH & Co. KG
Am Bahnhof 4a
82205, Glinting
Germany

Tel.: +49 81 057 7890
Fax: +49 81 057 78920
info@sam-solutions.com
www.sam-solutions.com

Security Test Plan Template

Contents

Revision History

Revison	Changes	Contributor	Date

1. Document Goals

This test plan is designed to:

- describe the **approach** used by the test group during testing;
- organize and implement the **testing process**;
- define the test **deliverables**;

2. Target Audience

Target audience is the Customer's representatives, SaM's management staff, software engineers and software testing team.

3. Project Description

Place the description of your project here.

4. Glossary

Defect (Bug) – nonconformance of the product to the functional/non-functional requirements of the specification.
Bug Tracking System – a system for storing, processing and tracking defects through their lifecycle.
Test case – a set of inputs, execution conditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
Audit - a review of a system in order to validate it. Generally, this either refers to code auditing or reviewing audit logs.
Authentication – a process of verifying identity, ownership, and/or authorization.
Authorization – guarantees that an authenticated user has the appropriate rights to access a definite entity of the application.
Risk - a possibility of a negative or undesirable occurrence. There are two independent parts of risk: *impact* and *likelihood*.
Impact - describes the negative effect that results from a risk being realized.
Likelihood - describes the chance that a risk will be realized and the negative impact will occur.
Threat – a potential event that will have unwanted consequence if it becomes an attack.
Vulnerability – a flaw or weakness in a system, such as coding bug or design flaw that can compromise the system's security.
Attack – occurs when an attacker has a motive and takes advantage.

5. Testing Objectives

The objective of security testing of the product is to:

- define security goals through understanding security requirements of the applications;
- identify the security threats;
- validate that the security controls operate as expected;
- eliminate the impact of security issues on the safety and integrity of the product;
- guarantee that the product will function correctly under malicious attacks;

6. Roles and responsibilities

The team members will be performing the following roles:

Role	Responsibilities	Contact information
Team Lead	• Test process setting up and adjusting • Security test plan creation • Test strategy authoring • Test activities tracking • Giving conclusion about the quality	mail@mailserver.com
Test Designer	• Security models creation • Test cases and test suites creation and updating	mail@mailserver.com
Test Engineer	• Running test cases • Defects authoring • Test results analysis • Test reports creation	mail@mailserver.com

7. Methodology

All the testing process is built on the basis of the following security attacks classes:

1. Dependencies (dependency testing)

The dependency type of testing supposes that the 3-rd part modules (or libraries, code, etc.) are tested. During this procedure a test engineer verifies whether:

- An application has vulnerabilities of (3-rd part) components it uses;
- Modules that provide security services fail;
- There are security vulnerabilities in the file system;
- There are security vulnerabilities in the registry.

1.2. Client side testing

During this type of testing a test engineer works with the user interface exclusively. He/she tries to enter incorrect input sequences, like:

- Escape characters;
- Long strings;
- Parts of some code in a programming language;
- Incorrect input values;
- Testing for error handling;
- Perform cross site scripting.

1.3. Exposed design vulnerabilities (design testing)

The design vulnerabilities can be caused by an immature design or development process. On this stage the next issues are controlled:

- Open unsecured ports;
- Insecure default values and accounts;
- Debug code intertwined with implementation code;

1.4. Exposed implementation vulnerabilities (implementation testing)

This type of vulnerabilities may occur because of implementation errors:

- Developers who develop only their modules could unintentionally reveal data: for example, incorrect validation;
- Time-of-check-to-time-of-use issues.

8. Scope of Testing

8.1. Features to be tested

Name the features/modules to be tested

2. Features not to be tested

Name the features/modules not to be tested

9. Assumptions for Test Execution

Before starting to test the following conditions should be met:

- The test environment is ready and configured appropriately.
- The version has been launched on the test environment and the version notification has been sent to a QA Team Lead.
- In the built notification all the features planned for the build are claimed to be ready.
- Smoke tests have been passed successfully (all the test cases have the Passed status).

If one of the above mentioned items is not fulfilled the QA Team Lead returns the build to the development team for correction.

10. Test Completion Criteria

In case all the conditions that are indicated below are performed the testing process supposed to be finished:

- All test cases planned for the current build have been run (except blocked ones).
- All the found defects have been posted to the bug tracking system.
- A test result report has been sent to all interested parties.
- A conclusion on the quality of the version has been done.

11. Testing Strategy

The strategy of security testing is built-in in the software development lifecycle (SDLC) of the application and consists of the following phases:

11.1. Requirements and use cases phase

11.1.1. Review policies and standards

On this stage a test engineer makes sure that there are appropriate policies, standards, and documentation in place. The entry point of this process is a list of all needed documents that describe general requirements to the security in the given business area as well as on the project level (process, technical, environment security requirements).

11.1.2. Develop measurement and metrics criteria

On this stage types of measurements are planned and the metrics to be collected are defined. The goal of this process is to focus on the most important aspects of the application security.

11.1.3. Security requirements analysis

Security requirements analysis is a very critical part of the testing process. On this stage a test engineer should understand what exactly security requirements are on the project. Also gaps that exist in the requirements are revealed during the process of analysis.
The security properties which are investigated during this process are the following:

- User management
- Authentication
- Authorization
- Data confidentiality
- Integrity
- Accountability
- Session management
- Transport security
- Tiered System Segregation
- Privacy

11.2. Design phase

11.2.1. Review design and architecture

This type of testing relates to static manual testing and implies that a test engineer reviews the design and architecture to find flaws that can result in insecure behavior of the application.
During the review a test engineer employs the following methodologies:

- The Comprehensive Method for Architecture Evaluation

or

- Independent Software Architecture Review methodology.

List of metrics:

- Number of flaws found

11.2.2. Create and review UML models

UML models allows to look at the subject of testing from the upper level of abstraction and understand the whole picture of the application (module). It allows us to find incompliances on the earlier stages of testing and react immediately. The next models are created on this stage:

- UML Use case diagram/Threat model.
- UML Class diagram

11.2.3. Create and review threat tree

A threat tree is created to break down threats into testable tasks that can be processed easily. The tree depicts the way what basic threats the application may face. It can be transformed into atomic test tasks and integrated in the project plan.

List of metrics:
- Number of threats.

11.3. Development phase

11.3.1. Static analysis

Static analysis is so a called white-box method of testing that allows finding all inconsistencies and errors that cannot be found using black-box methods and often leads to revealing critical vulnerabilities. It includes:

- Source code analysis. This type of testing is going to be done automatically with help of source code analyzer (see the list in the **Appendix A**).

- Code review. The code review is done by the testing team to identify all the flaws that were left after automatic analysis. It also allows to investigate risky code deeper.

List of **metrics** collected on this stage:
- % of code inherited from other projects
- % of code used from 3-rd party suppliers
- Number of vulnerabilities found

11.3.2. Dynamic testing

Dynamic type of testing supposes that the application has been run on the definite environment, in contrast with static types of testing.

11.3.2.1. Penetration testing

Penetration testing is going to be done in two ways: automatically and manually.
Penetration testing is done:
- **manually** using the procedures developed for a particular application and type of threat

or
- **automatically** using:
  - o web application vulnerability scanners,
  - o binary analysis tools,
  - o proxy tools.

The main attacks performed during penetration testing are listed below:

- Cross site scripting;
- SQL injection;
- Server misconfiguration;
- Form manipulation;
- Cookies poisoning;
- Platforms vulnerabilities;
- Weak session management;
- Buffer overflows;
- Command injection.

The list of tools are pointed out in **Appendix A**.

Metrics that are collected on the dynamic testing stage:

- Number of vulnerabilities found (by features, lines of code, etc.)

11.3.2.2. Configuration management testing

Configuration management is tested against errors in configuration of the application, environment, hardware, and database that can lead to vulnerabilities of the application.

List of metrics:

- Number of vulnerabilities found

12. Test results

Test results are sent to all interested parties and can contain:

- List of features which were tested;
- List of vulnerabilities found during testing;
- List of executed test cases with their statuses;
- List of risks which were defined during testing;
- Conclusion about the quality of the product.

13. Toolset

See **Appendix A.**

14. List of Deliverables

Below you can see a list of deliverables:

Name	Responsible person	Date of delivery
Test plan	Name Surname	Date
Security test plan	Name Surname	Date
List of vulnerabilities	Name Surname	Date
Test report	Name Surname	Date

Appendix A. Toolset

Tool	Open source	Commercial
Source-code analyzers	Rough Auditing Tool for Security, Find-Bugs	Klockwork Insight Fortify SCA
Web application scanners	WebScarab, Burp Suite	Acunetix WVS, HP Webinspect software
Database scanners	Scuba, SQLrecon	AppDetective, NeX-pose
Binary analysis tools	Fx-Cop, BugScam	IDA-Pro, Bin-Navi
Runtime analysis tools	CLR Profiler, .NETMon	Rational Purify-Plus
Configuration analysis tools	SSLDigger, Security Configuration Tool set	Desaware CAS/Tester
Proxy servers tools	Paros, Fiddler	
Miscellaneous tools	FireFox toolbar, SiteDigger, JUnit	Visual Studio Team System