|  |  |  |
| --- | --- | --- |
|  | |  |
|  | | Test Plan |
|  | |  |
|  | Test Plan  **Version: 1.1**  **Last Revised:**  **Author:** Julia Kuchmai | |

Table of Contents

[1 Scope 4](#_Toc279144288)

[1.1 Document Overview 4](#_Toc279144289)

[1.2 System Overview 4](#_Toc279144290)

[1.3 Testing process scope 4](#_Toc279144291)

[1.4 Exemption 5](#_Toc279144292)

[2 Test Environment 5](#_Toc279144293)

[2.1 Software Items 5](#_Toc279144294)

[2.2 Hardware and Firmware Items 5](#_Toc279144295)

[2.3 Equipment 5](#_Toc279144296)

[3 Human Resources/Staffing and Training Needs 5](#_Toc279144297)

[3.1 Staffing and Training needs 6](#_Toc279144298)

[3.2 Responsibilities 6](#_Toc279144299)

[4 Test Strategy 6](#_Toc279144300)

[4.1 Pass/Fail Criteria 6](#_Toc279144301)

[4.2 Defects Tracking Strategy 7](#_Toc279144302)

[4.3 Suspension Criteria 7](#_Toc279144303)

[5 Test Approach 7](#_Toc279144304)

[5.1 Functional Testing 7](#_Toc279144305)

[5.2 Performance and Load Testing 8](#_Toc279144306)

[5.3 Continuous Operation Testing 8](#_Toc279144307)

[5.4 Integration Testing 8](#_Toc279144308)

[5.5 Usability Testing 9](#_Toc279144309)

[5.6 Security Testing 9](#_Toc279144310)

[6 Test items 9](#_Toc279144311)

[6.1 Decoder’s testing. 9](#_Toc279144312)

[6.2 Validator’s testing. 10](#_Toc279144313)

[6.3 XSS decoder’s testing. 10](#_Toc279144314)

[6.4 Security Monitor subsystem 10](#_Toc279144315)

[7 Deliverables 11](#_Toc279144316)

[7.1 Test Logs 11](#_Toc279144317)

[7.2 Defect Reports 11](#_Toc279144318)

[7.3 Traceability Matrix 11](#_Toc279144319)

Document Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Description of Change** |
| **Julia Kuchmai** | 1.0 | November 30, 2010 | Creation |
| **Julia Kuchmai** | 1.1 | December 2, 2010 | Test items section added |
| **Julia Kuchmai** | 1.2 | December 3, 2010 | Minor changes |
|  |  |  |  |

Review

| **Name** | **Title** | **Date Reviewed** | **Date Approved** |
| --- | --- | --- | --- |
| **Kyle Quest** | Architect |  |  |
| **Anatoliy Lokshin** | Development lead |  |  |
|  |  |  |  |
|  |  |  |  |

References

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Author** | **Location** |
| MetraTech Security Framework Specification | Kyle Quest | <http://seceng.metratech.com/gf/download/docmanfileversion/20/78/MtSecurityFrameworkSpec.doc> |
| SecurityFramework Slides with examples | Kyle Quest | <http://seceng.metratech.com/gf/download/docmanfileversion/22/80/SecurityFrameworkSlidesWithExamples.pptx> |
| Validator requirements | Anatoliy Lokshin | <http://seceng.metratech.com/gf/download/docmanfileversion/52/120/Validatorsubsystemrequirements.docx> |
| Security Monitor requirements | Anatoliy Lokshin | <http://seceng.metratech.com/gf/download/docmanfileversion/55/123/SecurityMonitorSubsystemrequirements.docx> |
| Processor requirements | Anatoliy Lokshin | <http://seceng.metratech.com/gf/download/docmanfileversion/37/104/Processorsubsystemrequirements.docx> |
| XSS detector requirements | Maksym Sukhovarov | <http://seceng.metratech.com/gf/download/docmanfileversion/41/108/Cross-sitescriptingdetectorrequirements.docx> |
|  |  |  |

# Scope

## Document Overview

The document is intended for the employees involved in the testing process of the Security Framework and supports the following objectives:

* Identifying project information and the software components that should be tested.
* Describe the testing strategies, levels and types that will be performed during the project testing.
* List the deliverable elements of the test project.
* Defines software quality assurance tasks, responsibilities, tools and practices that are necessary to perform verification on the Security Framework System in accordance with Product Requirements Specification - “MtSecurityFrameworkSpec.doc”.

QA will consult with Design and Requirements subsystems documents in the case of inconsistency detection in the Product Requirements Specification. In this case “<subsystem name> Requirements.docx” and “<subsystem name> Design.docx” will have higher priority than the Product Requirements Specification.

## System Overview

MetraTech product security has become an important business objective once the products gained a significant visibility. A number of efforts have been made to improve the security of the MetraTech products; however, a lot of product and process enhancements need to be made to achieve the necessary level of product security.

Some of the product security problems include the following:

* Limited ability to handle malicious environments.
* Insufficient input validation and output encoding capabilities.
* Inability to handle advanced security threats that cannot be successfully represented using pure regular expression data validation techniques.
* High level of false positives for cookie data validation that results in product deployment issues (e.g., Microsoft).
* Missing advanced security functionality to detect attacks across multiple data inputs, user sessions, and product servers.
* Significant internal application resource exposure to the application users, which allows malicious users to temper with those internal resources to compromise the availability and integrity of the MetraTech applications and data.
* Insufficient access control.
* Hard to change default accounts and passwords.
* Fragmented and partial security functionality utilized in an ad-hoc manner.

## Testing process scope

Unit, System, Regression and Acceptance (smoke) testing stages are planned during Security Framework phase2 testing.

Functional, Behavioral, Continuous Operation, Performance, Usability, Security, Integration with ASP.NET, Automated and Static testing types of testing is planned.

During 2nd phase of Security Framework testing next subsystems and objects will be tested:

* Processor subsystem;
* Configuration Loader;
* Security Monitor subsystem;
* XSS detector;
* Decoder subsystems;
* Validator subsystems;
* Project documentation (design, requirements document and manuals);
* Logger and encoder subsystem testing will be included only in smoke test stage;

Traceability matrix will be created to describe coverage.

## Exemption

Stress, cross-platform and integration with MetraTech product testing are not included in the scope of this testing. QA will perform the testing only for the functionality implemented by Security Framework team.

# Test Environment

## Software Items

Testing will be performed in the following environment:

* Windows 7;
* Microsoft .NET Framework 4
* Microsoft Visual Studio 2010
* IronPython 2.7

## Hardware and Firmware Items

Virtual machine with following characteristic will be used to perform testing:

* CPU: Intel Xeon 2,67 GHz
* RAM: 2GB

## Equipment

No special equipment will be used in the course of testing

# Human Resources/Staffing and Training Needs

## Staffing and Training needs

Two full-time QA employees are required to perform testing of the Security Framework system.

QA employee need to have a basic knowledge of Metratech products, knowledge in Secure Programming Principles and Security testing, medium knowledge of MS Visual Studio and Iron Python. Also QA need to have strong skill in testing principles and practices.

## Responsibilities

A-MTOSS QA are responsible for

* designing test cases, test data, traceability matrix, test report
* reporting defects to the defect-tracking system
* functional testing according to the test cases
* Continuous Operation testing
* Usability testing
* Security testing
* Automated testing
* Integration with ASP.NET applications testing
* regression testing, when defects are fixed
* Acceptance(smoke) testing

A-MTOSS’s developers are responsible for tracking application’s versions and providing a correct version of the application for QA and also for Unit testing.

Both QA and developers are responsible for Performance testing performing.

# Test Strategy

Functional testing will be performed according to the Test Cases. It will cover all functional requirements described in the Product Requirements Specification.  
The set of test data will expand during development and verification progression.

Sanity testing will be performed before functional testing of each web application build to ensure that the major functionality is not broken.

Performance tests allow verifying web application’s characteristics under various regular workload and conditions that are very close to real situation.

Regression testing will be performed after fixing defects by the developers to ensure that the fixes do not result in new defects. Regression testing will be performed, until high and medium priority defects are present. Regression testing will be conducted to ensure that defects have been properly corrected or suggestions have been properly implemented. This type of testing includes selective retesting of related functionality to ensure that new defects have not been introduced or uncovered as a result of the changes made. The QA will be performing regression analysis of the changes and be determining what scope of regression testing is necessary.

## Pass/Fail Criteria

Any discrepancy in the Product Requirements Specification will be reported as a Defect. Functioning of the web application should not result in system crashes, or blocking resources of the computer. Such behavior will be reported as a Defect. Each test must produce the expected results described in Test Cases and Product Requirements Specification in order to pass. Any deviation from the expected behavior for any portion of a test will result in failure of the entire test and will be reported as a discrepancy through the Defect Tracking System.

Improvement of functionality that is not provided in the Product Requirements Specification, improvement of interface and architecture of web application or database will be reported as Change Request. Change Request is not mandatory to be fixed. Project Manager (Architect) will make a decision whether it is necessary to fix a particular Change Request or not.

## Defects Tracking Strategy

Found issues, changes and suggestions will be reported and each registered defect will have one of the following severities/priorities assigned. A defect’s severity indicates the impact of the defect on the system.

* Stopper - errors that prevent system tests (particularly or fully).
* High - system crash, data loss, unavailability of core functionality, such defects are easily reproducible under most configurations or defects with have high fix priority due to the business meaning.
* Medium – Serious or missing data related errors that will not prevent implementation. Those defects that impact not high level functionality and unacceptable user experience. Also includes high-priority issues that happen only under special circumstances (thus affecting only a small number of users).
* Low - Minor errors that do not prevent or hinder functionality. Spelling, broken functionality that has workarounds.

## Suspension Criteria

The testing may be suspended if a discovered defect requires a significant software change for resolution. The QA Lead will determine whether the defect(s) discovered during the testing require the testing to be suspended.

Defect regression testing will be conducted only after a new version of the software containing the fixed defect(s) is obtained from the Software Development team. Suspended testing will be restarted only after the regression testing is conducted for the corrected defect(s). If testing is not suspended, it will continue to its completion.

# Test Approach

## Functional Testing

Functional testing will be performed according to the Test Cases and cover all functional requirements described in the Product Requirements Specification.

### Test Case execution - Manual

Test cases will be developed to verify the product requirements as defined in the Product Requirements Specification. All test cases created will be executed during product testing.

The set of test data will expand during development and/or verification progression.

### Test Case execution - Automated

Automated tests will be included in Functional test cases.

The following main functionalities will be tested during automated testing:

XSS detector’s normalization: Unicode, Html entities, symbols elements;

* XSS detector: VB script, hiding code and obfuscation code detection;
* XSS detector: false positive;
* Decoder’s functionality;
* Validator’s functionality;

### Behavioral testing - Manual

Behavioral testing will be performed on the system in order to detect problematic behavior that may not be covered by the product requirements.

As a minimum, the operational scenarios covering typical and abnormal operation will be defined and executed. Also, some behavioral testing will be done simultaneously with the development to exercise the functionality. The results of behavioral testing will be reported in Defect Tracking System as a Suggestion and need to be discussed with Dev Lead and Project Manager before implementation.

## Performance and Load Testing

Performance testing is intended to verify that the system can normally process (decode, detect, log) huge amount of data.

Performance tests will be considered as passed if Security Framework can normally operates with 1000 (TBD) operations (decode, encode, XSS detect, security action/event log) at the same moment.

Performance testing will be performed according to the Test Cases and cover all performance requirements that will be prescribed (TBD).

## Continuous Operation Testing

Continuous Operation Testing is intended to verify that Security Framework system is stable and works properly 24/7.

Test virtual machine will be set up to simulate Security Framework usage. IronPithon will be used to simulate SF subsystem’s external calls.

State of virtual machine will be monitored on a daily basis to verify that the software works properly. Software should run without any errors for a period of a minimum of seven days as proof of reliability.

This testing will be performed for only release build or release-candidate build.

## Integration Testing

Integration with ASP.NET testing is intended to verify that Security Framework system can be used in other ASP.NET products.

### Test Case execution – Manual

Test cases will be developed to verify the integration with ASP.NET applications. All test cases created will be executed during product testing.

### Test Case execution – Automated

There will be no automated test cases performed during MCP compatibility testing

## Usability Testing

Usability testing will be performed to verify practicalness of using Security Framework system as a component of other applications from developer’s side.

The results of usability testing will be reported in Defect Tracking System as a Suggestion and need to be discussed with Dev Lead and Project Manager (Architect) before implementation.

## Security Testing

Security testing is included in scope of Functional testing.

# Test items

Configuration Loader and Processor subsystem will be tested using phase 1 test cases. Main purpose of testing is verifying that Security Framework system can be initialized and used as required.

## Decoder’s testing.

### Functional automated testing

Escaping/encoding standards have a lot modifications used in different web applications and each standard/modification has a huge amount of escape sequence that we need to test. Automated tests will be developed to ensure the coverage of this area with tests. IronPython will be used to realize automation. YAML format will be used for data sets representation. YAML format:

*# [Comment]*

*id: 1*

*message: [Test description]*

*payload: "[Tested data]"*

*expect: “[Expected result]” \*(optional)*

QA are responsible for developing scripts on IronPython and preparation YAML files with test data. Test data sets will cover all equivalent class (alphanumeric, punctuations, Cyrillic, control characters etc.) for symbols and strings that should be decoded and/or skipped with decoder. Test cases will be designed to ensure coverage and traceability. Positive and negative tests will be designed.

### Functional manual testing

Manual testing will be focused on decoder’s workflow:

* Different types of security event generation (bad input, empty/null, configuration problem etc.);
* Each field of security event/exception (EventType, CategoryName, SessionId etc.);
* Subsystem’s configuration testing;
* Interaction with different subsystems (using Processor rules);
* Decoding of control character sequences (null, escape, delete etc);
* Decoding of some special characters (known bugs of Python).

Development team will support QA team with test application on ASP.NET that allows verifying of each step described above.

## Validator’s testing.

### Functional automated testing

Automated test scripts (on IronPython) will be developed (similar to decoder’s subsystem). Test cases will be designed to ensure coverage and traceability. Positive and negative tests will be designed.

### Functional manual testing

Manual testing will be focused on validator’s workflow:

* Different types of security event generation (bad input, empty/null, configuration problem etc.);
* Each field of security event/exception (EventType, CategoryName, SessionId etc.);
* Subsystem’s configuration testing;
* Interaction with different subsystems (using Processor rules);
* String validation that ccontains control character sequences (null, escape, delete etc);
* String validation that ccontains special characters (known bugs of Python).

Development team will support QA team with test application on ASP.NET that allows verifying of each step described above.

## XSS decoder’s testing.

### Functional automated testing

XSS subsystem test cases (phase 1) will be updated and used for testing.

Set of automated test data will broaden.

### Functional manual testing

Test cases will be updated, Manual testing will be focused on detector’s workflow:

* Different types of security event generation (xss found, bad input, empty/null, configuration problem etc.);
* Each field of security event/exception (EventType, CategoryName, SessionId etc.);
* Subsystem’s configuration testing;
* Interaction with different subsystems (using Processor rules);
* Strings with some special characters verefying (known bugs of Python).

Development team will support QA team with test application on ASP.NET that allows verifying of each step described above.

## Security Monitor subsystem

### Functional manual testing

* XSS subsystem test cases (phase 1) will be updated and used for testing. Main objects of tests for phase 2 are
* Security monitor’s configuration (policies) - verifying creation/modification of Security Monitor rules;
* Logging of Security monitor’s actions;
* Security events generation and logging;

# Deliverables

## Test Logs

Test report will be developed at the end of phase 2 testing.

Testing status can be traced by test cases that will include results of all performed tests.

Test cases that are currently in progress can be found on SF team’s SVN repository:

<svn://qaautoserv/SecurityFramework/branches/SF1.0-Development/Docs/Tesing>

Test cases will be moved to forge portal when testing of subsystem is completed:

<http://seceng.metratech.com/gf/project/secframework/docman/?subdir=10>

## Defect Reports

All Defects will be reported to JIRA portal:

<https://jira.metratech.com/browse/SECFRM>

## Traceability Matrix

Traceability Matrix will be developed during test cases creation and can be found on SF team’s SVN repository:

<svn://qaautoserv/SecurityFramework/branches/SF1.0-Development/Docs/Tesing>

Traceability Matrix will be moved to forge portal when test cases preparation is completed:

<http://seceng.metratech.com/gf/project/secframework/docman/?subdir=10>