Cyreath.co.uk
Empirical Pragmatic Testing

Testing Software Security.

A discussion outlining a model for testing software security

Discussion Document
By Mark Crowther, Empirical Pragmatic Tester

1.0 INTRODUCTION

Software Security elements are present at all levels of the product. Testing for vulnerabilities in both the software and its related infrastructure requires a different approach to that which would usually be applied for more traditional testing. If a business hasn't acquired a reasonable level of mastery over other types of testing such as Functional, System or Acceptance testing then it's even more difficult to execute valid security testing that will traverse these weakly tested areas.

It's important to be clear on the fact that security testing includes testing of expected security functionality such as access control, authentication and cryptography but also includes risk based testing specific to the application. Examples of these are buffer overruns, race conditions and code injections.

Software security is an emergent quality of the software and it starts at the design phase with security specific requirements being captured for the product or system. It's these the test engineer will use to select the application specific test techniques. Ideally these are captured alongside the system requirements and not as an isolated or generic set being cut-and-copied from the previous requirements catalogue.

Yet, how often do we see good security requirements elicitation and analysis processes that provide categorised and proritised requirements? Rarely, if ever on a typical software development project. As with a basic mastery of traditional testing leading onto more complete security testing it follows that good system requirements capturing will lead onto capturing more complete security requirements.

1.1 Where to begin? Elicit, Capture, Analyse.

Performing good security requirements capture starts early in the Software Development Life Cycle (SDLC) and my preferred approach is to apply the SQUARE methodology, as defined by CERT, using appropriate CASE tools in support of this where they are available.

Methods to elicit application specific security requirements are similar to those used for the system requirements and doing so at the same point in the SDLC will bring the same benefits. However, when defining security requirements the requirements engineer will benefit from the input of artefacts such as Scenarios, Misuse Cases, defined Security Goals as well as reference to definitions of the agreed level of security needed.

In addition the engineer will need to understand any legislative, legal or contractual requirements that must be achieved. Ensuring expert groups such as the legal and sales teams as well as the SOX advisor are involved will reduce the risk of missing an essential requirement.

With Security Requirements covered, what's next?

Now we are in a position to look at what specific security testing will be needed. Starting with test planning and performing test analysis to elicit the specific test requirements.

2.0 TEST PLANNING AND ANALYSIS

Now that we have a clear idea of the security requirements, in addition to the system requirements we expected, it's possible for the test team to add time to the test project schedule for planning the test activities related to security. We didn't forget that would be needed did we? If there are a significant number of security requirements the team may choose to produce a separate Security Test Plan. This is often useful as the test plan will include elements such as Features to be tested, Approach and Approvals. Breaking the security testing out from system testing will allow these to be managed with more visibility, delivered more effectively and progress tracked as a distinct work stream by the project manager.

2.1 Test Design, Test Case & Test Procedure Specifications

In the Test Plan template there is a section for test requirements that isn't part of the IEEE 829 model. For security testing it's recommended this is removed and a set of test design specifications are created. These will describe the features and combinations of features that will be tested for the in scope items listed in the test plan, in addition to the specific test techniques to be used.

Remember, this is features such as access control, authentication and cryptography but also now the application specific security test requirements. Each test design specification should reference at least the security requirement it applies to and as with the test plan need approval from the project authorities.

Test case specifications detail the individual tests that will be executed and detail specific inputs, outputs and features expected. They should include exact values, tolerances and volumes for data or boundary values for example. Test procedure specifications are critical to understanding what collection of test cases are planned for execution against each feature, in what order and to record what the outcome of the test set was. As always get the test procedure specifications reviewed and approved by the project authorities.

Hardware and software should be drawn from the approved test configuration items and environments matching the known configuration assemblies. This will ensure the aforementioned documents have accurate and known environmental data and reduce uncertainty over environmental impact on test results.

2.3 Regression testing, canned tests and coverage analysis.

If the business already has a set of security tests for earlier versions of the software these should be assessed for use as a regression suite. In addition, suites of tests that come with security testing tools should be assessed for use. However, 'canned tests' must not be relied upon to provide sufficient testing. As for system functionality consider using a code coverage tool to assess test coverage and look for fault sensitive lines of code that are not tested or tested incompletely.

At this stage we've got a good set of security specific requirements, planned the overall approach and performed analysis to identify specific test cases needed for the new development. Next we'll go about delivering the actual testing.

3.0 DELIVERING THE TESTING

2.1 Level 1 - Testing Security features

At this point we'll have two sets of tests, one that tests our standard security functionality and one that tests the application specific security features. For standard security features such as access control and user authentication we'll use the in house design specifications to identify the tests needed.

These may require testing of user authentication against the database, enablement/disablement of browser functionality, submission of syntactically correct but invalid data, infrastructure failures such as the loss of intermediate servers or weaknesses due to 3rd party applications and other 'expected' scenarios already anticipated and defensively coded for in the application.

2.2 Level 2 - Testing the software's security.

In selecting the risk based tests which will address security testing outside of expected features we can follow the model provided by Fortify, namely the Seven Kingdoms: Input validation and representation, API abuse, Security features, Time and state, Errors, Code quality, Encapsulation and Environment.

Against these we select the appropriate Phyla dependent on the software system, development technology and its target environment. Typical phyla include Command Injection, Log forging, Cross Site Scripting and SQL Injection. Our security testing approach introduces Defect Driven Exploratory Testing at this point.

2.3 Level 3 - Penetration testing on deployment

This may be referred to as vulnerability testing or risk assessment, either way the testing is conducted once the product is deployed on the target environment. It is possible to conduct penetration testing at the Beta phase of the SDLC but a limited subset of the agreed tests should be conducted on the live environment. Any security issues that are found will be reported to the customer together with an assessment of their impact and a proposal for mitigation or a technical solution.