SECURITY TESTING PROCESS IN SDLC

Table of Contents

1. INTRODUCTION

1.1 Description

This Security testing Strategy document defines the approach to security system. It briefly describes the methods and tools used by Security Engineers to validate the Security of the system.

1.2 Purpose

The purpose of this document is to outline the approach that the Security Engineering team will take to ensure that the Security Acceptance Criteria is met. Specifically, this document details the :->

- Security testing process.
- Security testing life cycle.
- Security strategy
- Security test Environment
- Security Test execution plan
- Acceptance criteria

2). SECURITY TESTING PROCESS

2.1 Security Testing In SDLC

Security testing is very long and time consuming process. This process starts from early phase of the software development life cycle (SDLC). Security testing must necessarily involve two diverse approaches :->

- Testing security mechanisms to ensure that their functionality is properly implemented, and
- Performing risk-based security testing motivated by understanding and simulating the attacker's approach.

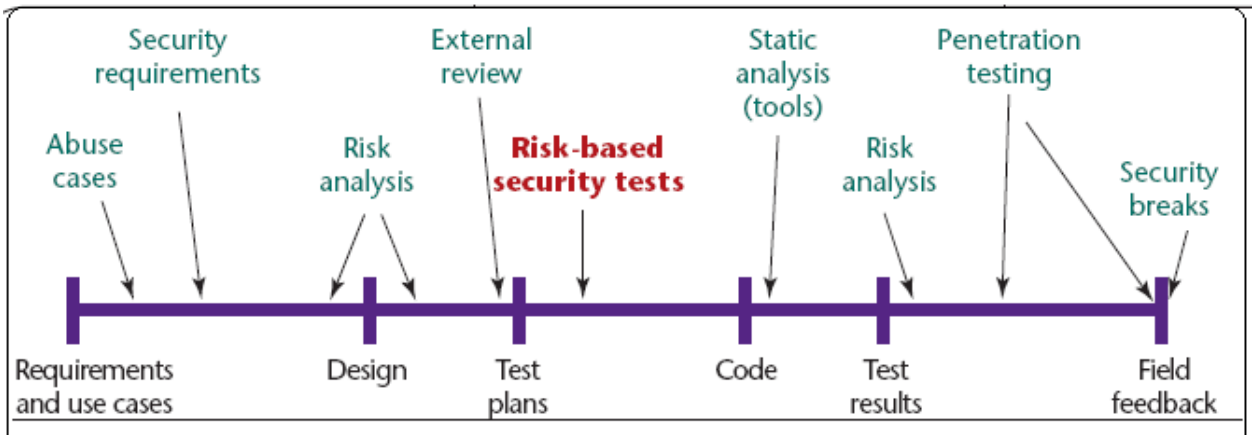

Fig 1. Security Testing Flow through out the software development Life cycle

Test should be designed considering the following points :->

- Probability of occurrence of event
- Risk associated with each occurrence

2.2 Security Testing Vs Other Testing

Security testing is different from functionality testing or any other testing. Its approach is also different. Security testing is defined as a process of identifying the various vulnerabilities in a system which are exposed because of improper design or coding issues. Application level threat cannot be avoided by network firewalls as data comes in HTTP request which these firewalls let pass. So it becomes even more important to handle the security when it applies application level than what happens at Network levels.
The difference between software safety and software security is therefore the presence of an intelligent adversary bent on breaking the system. Security is always relative to the information and services being protected, the skills and resources of adversaries, and the costs of potential assurance remedies; security is an exercise in risk management. Risk analysis, especially at the design level, can help us identify potential security problems and their impact.

2.3 Security testing life cycle

Security testing has four phases in the security life cycle. They all are equally important in the security testing process and have there own significance.

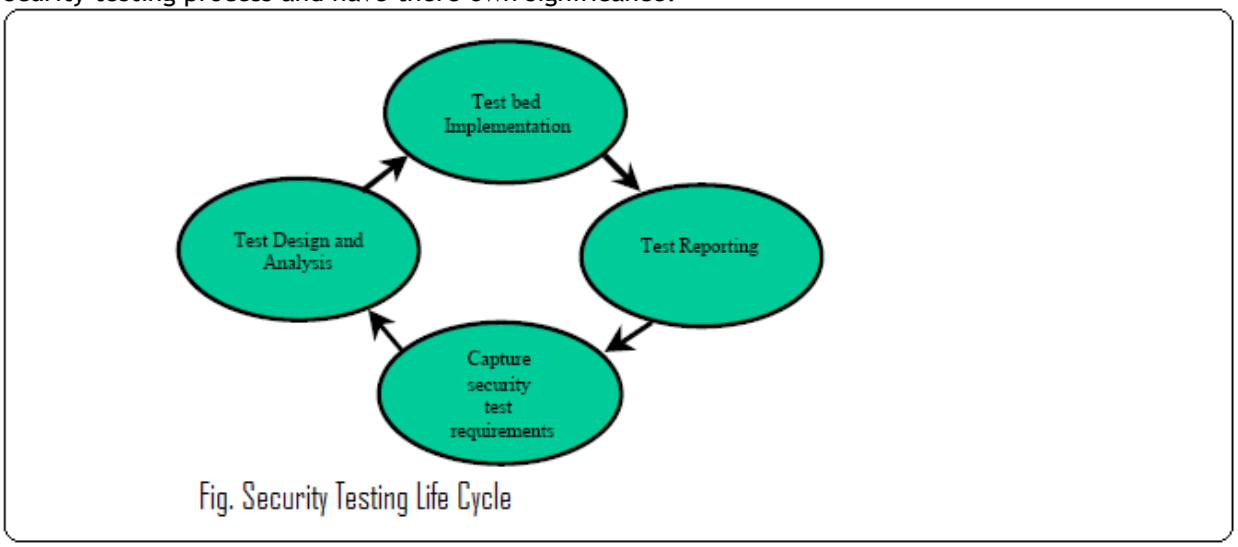

Fig. Security Testing Life Cycle

The lifecycle process of security testing is similar to the software development lifecycle process. In the proposed approach, the security testing lifecycle stages are as follows :->

- **Capture Security Test Requirements**

To define the scope of security testing, check the stated requirement against the parametric template.
This will help in identifying all the missing elements or the gaps in security requirement capture. The
Next step would be to allocate appropriate weightages to the various security sub parameters. these weightages would typically be determined by the nature of business domain that the application caters to and the heuristic data available with the security test experts.

- **Security Test Analysis and Design**

Once the weightages for each security parameter is finalized, the next task is to create test scenarios for testing the security requirements. The weightages assigned to each sub-parameter goes on to suggest the number of scenarios that need to be tested for a particular requirement and the complexity involved.

The parametric approach also helps in identification of the extent of data that would be needed for test execution and also the distribution of data based on various scenarios. Also, this will help in identifying the kind of tools that will be needed and determine the appropriateness of a tool for a given type of test.

- **Security Test bed Implementation**

In case of application level testing the approach of testing the security vulnerabilities should be more logical. Here logical security deals with the use of computer processing and/or communication capabilities to improperly access information. Second, access control can be divided by type of perpetrator, such as employee, consultant, cleaning or service personnel, as well as categories of employees. The type of test to be conducted will vary upon the condition being tested and can include:
Determination that the resources being protected are identified, and access is defined for each resource.

Access can be defined for a program or individual. Evaluation as to whether the designed security procedures have been properly implemented and function in accordance with the specifications.
Unauthorized access can be attempted in on-line systems to ensure that the system can identify and prevent access by unauthorized sources.

Most of the security test scenarios are oriented to test the system in the abnormal conditions. This throws up the challenge of simulating the real life conditions that the test scenarios require. There are various tools that can be used to test various vulnerabilities that the system may have. The choice of such a tool is determined by the extent of vulnerability, the tool is able to test, and Customization of the tool should be possible and testing of the system using both the external as well as the internal views. The local users are the internal view and the external users constitute the external views. There are various programming tools that can be programmed to simulate the various access control breaches that can happen.

- **Interpreting Security Test Reports**

Once tests have been executed and security gaps have been identified, the gaps need to be analyzed in order to suggest improvements. The reports need to be validated as many a times as possible. The Reports may be misleading and the actual loopholes might be elsewhere or might not exist at all. The
Security test reports are like X-ray reports, where a security expert is essential to decipher the vulnerabilities reported and bring out the actual problem report. There are various security reporting tools. While selecting a security-reporting tool, the following parameters have to be taken in to consideration:

1. Extent if vulnerability, the tool is able to report.
2. Customization of the tool should be possible.
3. Testing of the system using both the external as well as the internal views.

Once the analysis of the report is completed a list of vulnerabilities and the set of security features that are working is generated. The list of vulnerabilities is classified. The classification level of vulnerability is determined by the amount risk involved in the security breach and the cost of fixing the defect. Once the classification is done the vulnerability is fixed and retested.

- **Conclusions**

A parametric approach to security testing not only, works very well for security requirement capture,Effort estimate and task planning but also for testing different levels of security requirements by different components within same application. This approach for requirements gathering comes out to be real handy in capturing those security requirements, which cannot be captured by traditional means. This approach enables one to adopt the parametric matrix at each and every stage of security testing, and improve the values in the matrix after each execution of a project. Due to this capability of incorporating experiential data, the parametric approach can provide the most effective mechanism for security testing of Internet applications.

2.4 Security Testing Approach

Security Testing has a defined approach for finding security vulnerabilities in the application.

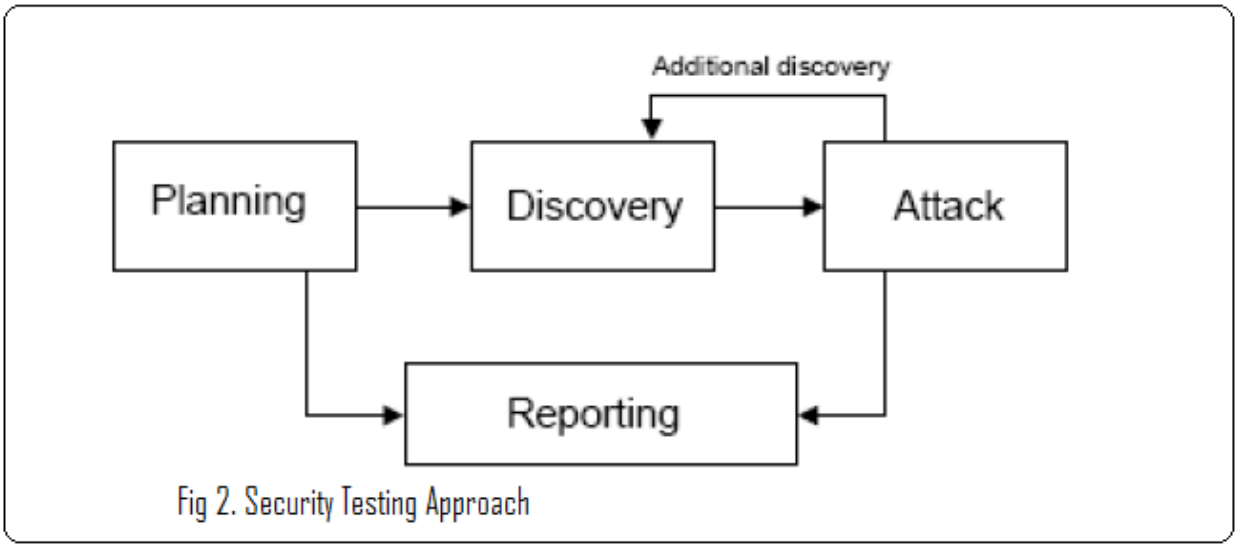

Fig 2. Security Testing Approach

In the Planning Phase, rules are identified, management approval is finalized, and testing goals are set. The Planning Phase sets the groundwork for a successful penetration test. No actual Testing occurs in the planning phase. The Discovery Phase begins the actual testing. The second part of the Discovery Phase is vulnerability analysis. During this phase, services, applications, and operating systems of scanned hosts are compared against vulnerability databases (for vulnerability scanners this process is automatic). Generally, human testers use their own database or public databases to identify vulnerabilities manually. This manual process is superior for identifying new or obscure vulnerabilities, but is much slower than an automated scanner.

Executing an Attack is at the heart of any penetration test. This is where previously identified Potential vulnerabilities are verified by attempting to exploit them. If an Attack is successful, the vulnerability is verified and safeguards are identified to mitigate the associated security exposure. Frequently, exploits that are executed during attack execution do not grant the maximum level of access that can be gained by an attacker. Instead, they may result in the Testing team learning more about the targeted application module and its potential vulnerabilities, or they may induce a change in the state of the security of the targeted

application. In either case,Additional analysis and testing is required to determine the true level of risk for the network.
This is represented in the feedback loop in figure above between the Attack and Discovery Phase of a security test.
The Reporting Phase occurs simultaneously with the other three phases of the penetration test. In the Planning Phase, rules of engagement, test plans and written permission are developed. In The Discovery and Attack Phases, written logs are usually kept and periodic reports are made to System administrators and / or management, as appropriate. At the end of the test an overall testing report is developed to describe the identified vulnerabilities, provide a risk rating, and to give guidance on the mitigation of the discovered weaknesses.

Corrective measures can include closing discovered and exploited vulnerabilities, modifying security policies, creating procedures to improve security practices, and conducting security awareness training for personnel to ensure that they understand the implications of poor application configurations and poor security practices.

So always remember following four points before testing the application for security.

1. Understand deeply what you are testing.
2. Think maliciously about the target.
3. Attack the target by applying your malicious ideas.
4. Stay informed about new attacks that might affect the target you are testing.

3). SECURITY STRATEGY

Threat modeling for security testing

Threat modeling is important part of the security testing. The threat model creation process should include representatives from the design team, the programming team, and the testing team. Each member brings a different point of view and different knowledge about the product. You risk overlooking valuable information about or insights into the product if the threat model creation process doesn't include some one from each of these disciplines. External attackers don't have access to the people who created the product or to the product specifications.

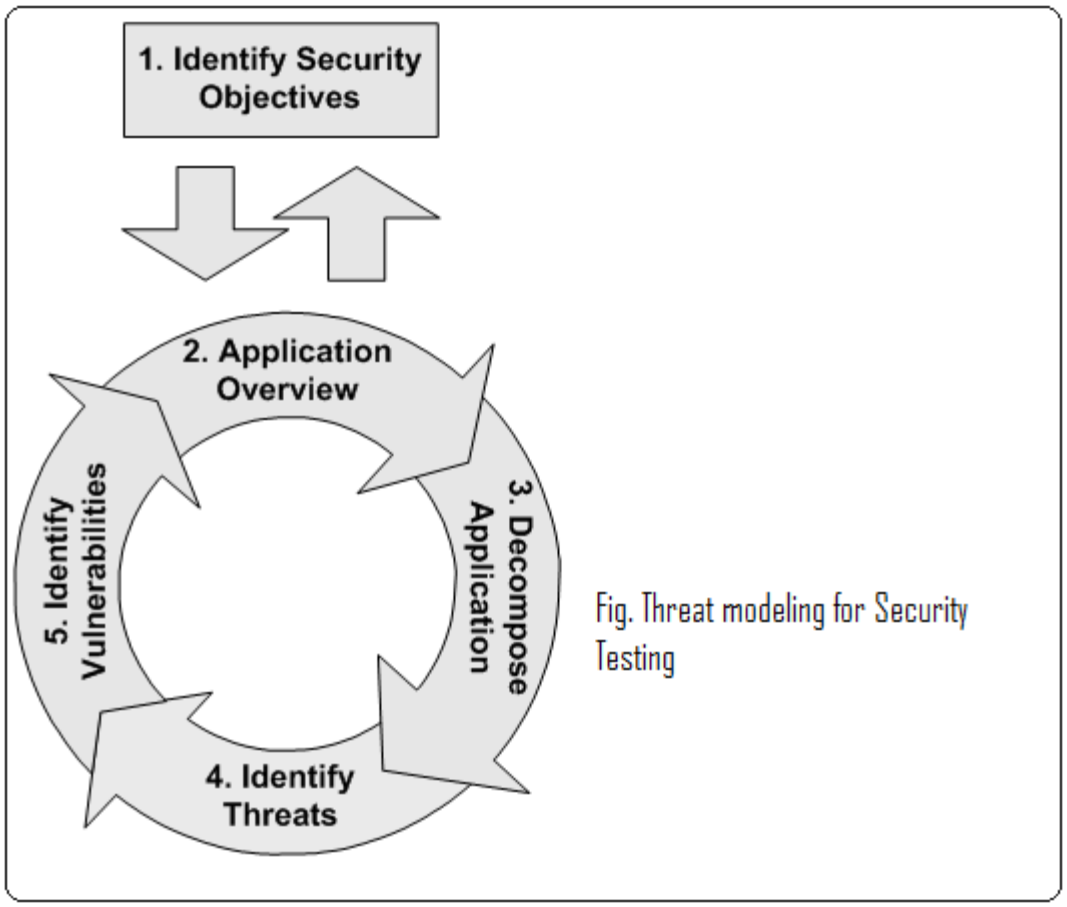

Fig. Threat modeling for Security Testing

Threat model (TMs) typically consist of five key parts :->

- Identify security objectives
- Application overview
- Decompose application
- Identify threats
- Identify vulnerabilities

For shipping more secure software we have to use each part of the threat model to find security problems.

3.1 Finding entry points

An entry point is a place where input can be supplied to the application. For an attacker, an entry point is an optimal place to attempt to break the application. In security testing, it is important that you identify and investigate high-risk entry points as follows:

- **Finding and Ranking Entry Points**

Threat Models for Security Testing and data flow diagrams (DFDs) can be useful to find security threats in an application. This gives us a good understanding of how the software works and how the data flows through it. But what happens when no threat model or DFD is available? Use the hit and trial and other technique for finding the entry points. Even when your application has threat models or DFDs, assume that they might not indicate all of the entry points or that this information is incorrect, out of date, or incomplete.

Generally, attackers won't have the threat model and DFD resources available, so they will use other resources and techniques—some of which help decompose your product and find the entry points. If you rely only on the accuracy of the threat model and DFD, you will definitely miss an entry point that an attacker will find.

Following is a list of some of the most common entry points that attackers probe for security vulnerabilities in applications, but it is by no means all inclusive.

- Files
- Sockets
- Hypertext Transfer Protocol (HTTP) requests
- Named pipes
- Pluggable protocol handlers
- Malicious server responses
- Programmatic interfaces
- Structured query language (SQL; databases in general)
- Registry
- User interfaces
- E-mail
- Command-line arguments
- Environment variables

In security testing, it is crucial for you to identify as many entry points as you can. The more you know about the entry points in your applications, the better you will be able to prioritize and target your security testing efforts. Otherwise, you might miss several vulnerabilities that a malicious attacker will not.

3.3 Different Security Attacks

There many other attacks which also we should consider during security testing :->

1. Session Management

Session management is necessary to maintain the identity of user across multiple requests. Cookies are information which is stored on client machine by web server. They are basically name-value pair which website uses to retrieve data when user visits the site again or across requests. Attackers can tamper this data to acquire information. The various attacks that can happen are:

- **Insufficient Session Expiration**

The application allows the attacker to reuse the old session IDs. All it needs for an attacker is to know the old session id and he can reuse the same.
Scenario: All application pages
Test: Test must be done to make sure, application logs off or session is expired after some time.

- **Session hijacking**

If session ids are predictable, it is possibility that attacker can guess the session id and can use it.
Scenario: Any page after login
Test: Test should be done to check whether session ids are predictable. Test to check multiple session of same user is not allowed. Test to check important data is transferred using HTTPS protocol.

2. Error handling

It is common mistake from developer that errors are not handled properly and lot of information is disclosed and leads to information disclosure attack. The various attacks that can happen are:-

- **General security cases**

For penetrating application some test cases should be designed which contains the following cases which encompasses a large Number of test cases:

- **Buffer Overflow Testing**
- Long strings of a single character
- Lengths of strings with common boundary conditions: 128 bytes, 256 bytes, 1024 bytes, 65535 bytes…
- Varying string patterns
- Random lengths of strings

- **SQL Injection Testing**
- Apostrophe
- Quotation mark
- Comma
- Bracket
- Alternate encodings of the same

- **Cross Site Scripting Testing**
- Less-Than Sign
- Greater-Than Sign
- Quotation Mark
- Apostrophe
- Alternate encodings of the same

- **Format String Testing**
- %s, %x, %n
- Various repetitions of the same

- **Random Data Testing**
- Purely random data included in requests
- Purely random data included as parameters
- Encoded random data included as parameters

- **Random Mutation of Valid Data Testing**
- Bit flipping of known legitimate data
- Byte stream sliding within known legitimate data
- Random Mutation of Valid Data Testing

4). SECURITY TEST ENVIRONMENT

4.1 Security Testing Environment

Testing should be carried out in development and test environment. The penetration testing should be carried out without the intention to exploit the system for personal gains. A large number of freeware tools are available for black box penetration testing. Test should be carried out on manual and automated basis.

4.2 Equipment Environment

For security testing some specific hardware and software is required which can be notified at the time of actual testing.

- **Hardware Requirement**

- **Software Requirement**

4.3 Tools

Security tools are very crucial for the successful completion of the security testing .the various tools used for black box security testing are: Paros, Tamper IE, Web scrap all these tools can be used to tamper the http request and response output and can help in identifying the vulnerabilities in the system.

- **Fiddler**

Fiddler is a HTTP Debugging Proxy which logs all HTTP traffic between our computer and the Internet. Fiddler allows you to inspect all HTTP Traffic, set breakpoints, and "fiddle" with incoming or outgoing data.

4.4 Workload Distribution

The Test Lead and Project Manager will determine when security testing will start and end. The Test lead will also be responsible for coordinating schedules, equipment, & tools for the testers as well as writing/updating the Test Plan, Weekly Test Status reports and Final Test Summary report. The testers will be responsible for writing the test cases and executing the tests.

4.5 Procedures

Testing should be carried out in development and test environment. The penetration testing should be carried out without the intention to exploit the system for personal gains. A large number of freeware tools are available for black box penetration testing. Test should be carried out on manual and automated basis.
There are 5 basic steps for proceeding Security testing:

1. Problem Identification
2. Defect rectification
3. Re-testing
4. Sign-off testing activities
5. Sign-off Testing

It briefly describes the methods and tools used by Security Engineers to validate the Security of the system.

5). SECURITY TEST EXECUTION PLAN

5.1 Introduction

Security Testing occurs in an iterative fashion. During the initial stages of testing, tests and analysis continue until required tuning is identified. Security Testing is the process by which software is tested and tuned with the intent of realizing the required Security level. For achieving required security level all test cases should be run in accordance with the proper process. Let's look more closely at this process. In the simplest terms, this approach can be described as shown in Fig.

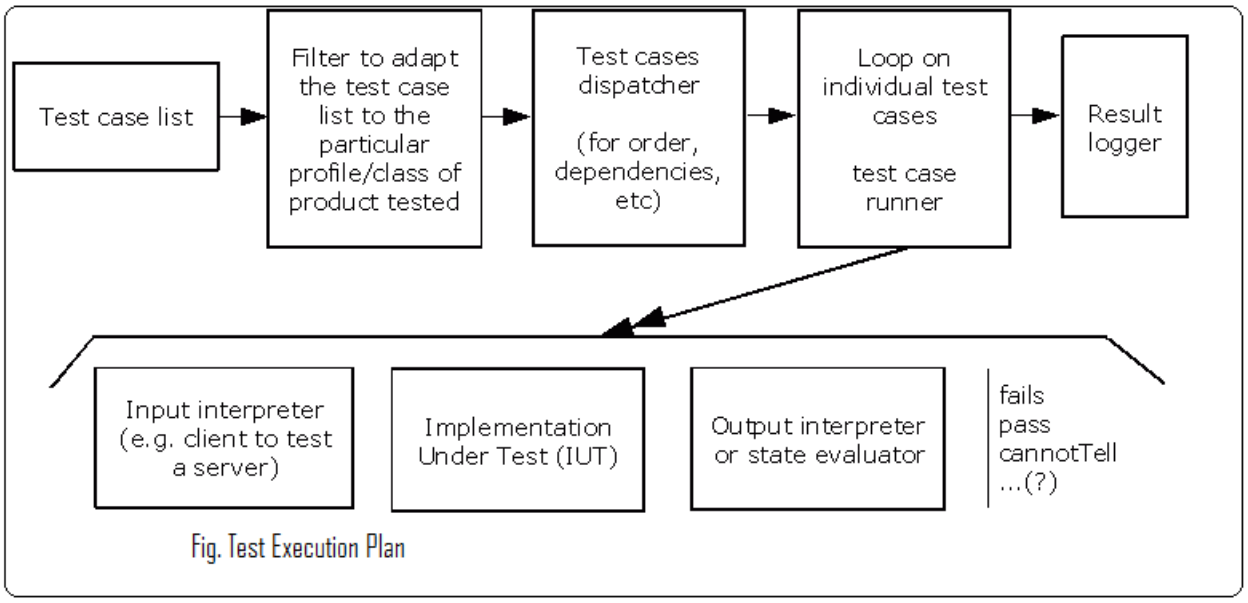

Fig. Test Execution Plan

There are three categories of tests that include all of the specific test types that are normally performed on a system. They are discussed in the three aspects titled :->

- Benchmarks
- Scheduled Tests
- Exploratory Tests.

Various combinations of the tests included in these aspects, coupled with a scientific approach to their execution, can ultimately unveil all detectable Security issues and bottlenecks in the application.

5.2 Evaluate System

The Evaluate System aspect is a detail-oriented phase of a Security testing Process. This aspect includes the following activities :->

- Determine data flow in the application
- Determine entry points in the system
- Determine System Functions
- Determine User Activities
- Determine System Architecture
- Determine Acceptance Criteria
- Validate Acceptance Criteria
- Determine Usage Model
- Validate Usage Model
- Determine Potential Risks
- DB server

5.3 Develop Test Assets

The Develop Test Assets aspect of Security Testing covers three activities :->

- Develop Risk Mitigation Plan
- Develop Test/Engineering Strategy

These two items are living items and will be updated throughout the engineering process. It is important to make these assets as complete and correct as possible early in the project to increase overall project efficiency.

5.4 Execute Benchmark Tests

The Execute Baseline and Benchmark Tests aspect is geared toward executing small security scenarios that are used to validate the correctness of the automated test scripts, identify obvious security issues early in the testing cycle, and provide a basis of comparison for future tests. The activities included in this aspect of Security testing are :->

- Create an Initial Benchmark
- Re-Benchmark after Tuning

5.5 Analyze Results

The Analyze Results aspect is where the decisions are made about how to continue with the engineering effort. All of the remaining aspects are reached as a result of the decisions made in the Analyze results aspect. The activities included in this aspect of Security Testing are:

- Evaluate Results
- Determine if Acceptance Criteria is Met
- Determine of the Test is Conclusive
- Determine if Tuning is Required

Each test execution, regardless of type, must be analyzed to identify obvious or potential Security issues and/or the effects of the tuning effort. This analysis may or may not require formal reporting. At a minimum, notes should be kept and results saved, even for tests that are determined to be invalid. Analysis is often more of an art than a science.
Testers, developers and architects should all be involved in the analysis process, since the results of the analysis drive the tuning effort.

- **Evaluate Results**

Evaluating results is where the art part of Security Testing comes in. There is no single correct way to evaluate all of the data that results from a Security test execution. In general, though, evaluation is comparing to previous tests, finding and analyzing patterns, and applying past experiences.

- **Acceptance Criteria**

Determining if the acceptance criteria have been achieved is often the easiest determination made during a Security Testing engagement. This activity is simply comparing the results from the most recent test, or suite of tests, to the acceptance criteria. If all of the results meet or exceed the criteria, engineering is complete. Move on to the Complete Engagement Aspect. If not, continue evaluating results.

- **Conclusive**

Determining if a particular test or suite of tests is conclusive is often challenging. Most generally, if the results do not meet the acceptance criteria, you can't figure out what is causing the poor Security and the test results are reproducible, the test is probably inconclusive. In the case of an inconclusive test, move on to the Exploratory Test Needed aspect. Otherwise, continue evaluating the results.

- **Tune Now**

If you get this far, there is either a detected Security issue with the system, or more tests need to be conducted to validate compliance with additional acceptance criteria. If all the tests that have been conducted so far have met their associated acceptance criteria, but there are more criteria to be tested, move on to the Scheduled Tests aspect. If this is not the case, there are two options remaining.

1. re-execute the last test to see if the results are reproducible or
2. Move on to the Tune aspect.

5.6 Scheduled Tests

The Execute Scheduled Tests aspect includes those activities that are mandatory to validate the Security of the system. These tests need to be executed even if no Security issues are detected and no tuning is required.

SQL injection

- Finding entry points
- Error handling
- Canonicalization issues
- Finding week permissions
- Anti-Automation
- authentication
- Session Management

6). SECURITY ACCEPTANCE CRITERIA

6.1 Introduction

Security testing efforts always have two sets of criteria associated with them.

- Security criteria
- Engagement completion criteria.

In the sections below, both types of criteria are explained in general and in specific. Detail for security engineering effort. The effort will be deemed complete when either all of the security criteria are met, or any one of the engagement completion criteria is met.

6.2 Security Criteria

Security criteria are the specific target Security requirements and goals of the system under test.

6.3 Engagement Complete Criteria

In cases where Security requirements or goals cannot be achieved due to situations Outside of the control of the Security Engineering Team, the Security effort will be considered complete when any of the following conditions are met.

- All bottlenecks preventing the application from achieving the Security criteria are determined to be outside Security Engineering Team control/contract.
- The pre-determined engagement end date is reached.
- The Security Engineering Team and stakeholders agree that the application performs acceptably, although some Security requirements or goals have not been achieved.