EC-Council Licensed Penetration Tester

Methodology: Source Code Penetration Testing

Penetration Tester:			
Organization:			
Date:		**Location:**	

Test 1: Identify the programming language used

Target Organization	
URL	
Programming Language Identified	
Techniques Used	
Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis:

Test 2: Categorize the application's architecture

Target Organization	
URL	
Application Architecture Determined From	
☐ Design Documents:	
☐ Business Requirement Documents:	
☐ Functional Specification Documents:	
☐ Test Results:	
Application Information Gathered	1. 2. 3. 4. 5.
Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis:

Test 3: Verify input and data validations

Target Organization		
URL		
Data Validation Mechanism is Present	☐ YES	☐ NO
Length Checks Implemented for all Input Fields	☐ YES	☐ NO
Data Validation Strategies		
Tools/Services Used	1. 2. 3. 4. 5.	

Results Analysis:

Test 4: Verify authentication

Target Organization			
URL			
Internal Connection Authentication			
External Connection Authentication			
Method Used when Confidential Information is Passed	☐ POST		
	☐ GET		
Security Issues Outside the Scope of Authentication			
Authentication Credentials Passed In Clear Text Form		☐ YES	☐ NO
Tools/Services Used	1. 2. 3. 4. 5.		

Results Analysis:

Test 5: Check for proper authorization

Target Organization	
URL	
Access Privileges for Users	
Authorization Procedures	
Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis:

Test 6: Identify for proper session management

Target Organization		
URL		
Session ID Discovered		
Next Session ID can be Guessed		
Session Storage Technique	1. 2. 3.	
Session Tracking Method	1. 2. 3.	
Session Validation	☐ True	☐ False
Session Expiry Time		
Tools/Services Used	1. 2. 3. 4. 5.	

Results Analysis:

Test 7: Check for cross site scripting vulnerabilities

Target Organization		
URL		
Input Fields Accept "Html Code" and Execute it	☐ YES	☐ NO
Scripting Languages Accepted by Input Fields		
1.	4.	
2.	5.	
3.	6.	
Information Gathered by Exploiting Cookies	1. 2. 3. 4. 5.	
User Validation Before Performing Security-Sensitive Operations	☐ True	☐ False
Tools/Services Used	1. 2. 3. 4. 5.	

Results Analysis:

Test 8: Check for SQL injection vulnerability

Target Organization		
URL		
Application Input Fields Accept SQL Queries	☐ True	☐ False
Query Construction Method	1. 2. 3.	
Techniques Used to Access Database	1. 2. 3.	
Privilege of the User Account Used to Make Database Connection	1. 2. 3.	
Tools/Services Used	1. 2. 3. 4. 5.	

Results Analysis:

Test 9: Check for proper buffer overflows and overruns

Target Organization	
URL	
Identified Input Areas	
Identify Input Fields that take Strings as Input	1. 2. 3. 4. 5.

Unsafe Functions used by the Code	
1.	4.
2.	5.
3.	6.

Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis:

Test 10: Check for vulnerabilities in error handling mechanisms

Target Organization	
URL	
When an Error Occurs in an Application, Check	
☐ System information is leaked to the user	
☐ Resources are Locked	
☐ Sessions are Terminated	
☐ Calculations and Business Logic are Halted	
Sensitive Information Revealed due to the Error Message	1. 2. 3. 4. 5.
Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis:

Test 11: Check for secured cryptography

Target Organization		
URL		
Cryptographic Methodology Implemented		
Clear Text Used for Storing Confidential Information	☐ True	☐ False
Short Keys Used	☐ Yes	☐ No
Weak Algorithms Used	☐ Yes	☐ No
Keys Storage Location	☐ Secure	☐ Insecure
Algorithms Used	☐ Standard	☐ Non-Standard
Hard Coding of Keys Implemented	☐ Yes	☐ No
Tools/Services Used	1. 2. 3. 4. 5.	

Results Analysis:

Test 12: Check for secured logging

Target Organization	
URL	
Sensitive Data Logged when Error Occurred	1. 2. 3. 4. 5.
Logging Frameworks	1. 2. 3. 4. 5.
Tools/Services Used	1. 2. 3. 4. 5.

Results Analysis: