

EC-Council



Certified Application Security Engineer

COURSEWARE

EC-COUNCIL OFFICIAL CURRICULA

DO NOT COPY
This page is intentionally left blank.
badalshiva@gmail.com

Certified Application Security Engineer

EC-Council

Copyright © 2018 by EC-Council. All rights reserved. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication. without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law. For permission requests, write to EC-Council, addressed "Attention: EC-Council," at the address below.

EC-Council New Mexico
101C Sun Ave NE
Albuquerque, NM 87109

Information has been obtained by EC-Council from sources believed to be reliable. EC-Council uses reasonable endeavors to ensure that the content is current and accurate, however, because of the possibility of human or mechanical error we do not guarantee the accuracy, adequacy, or completeness of any information and are not responsible for any errors or omissions or the accuracy of the results obtained from use of such information.

The courseware is a result of extensive research and contributions from subject matter experts from the field from all over the world. Due credits for all such contributions and references are given in the courseware in the research endnotes. We are committed towards protecting intellectual property. If you are a copyright owner (an exclusive licensee or their agent), and if you believe that any part of the courseware constitutes an infringement of copyright, or a breach of an agreed licence or contract, you may notify us at legal@eccouncil.org. In the event of a justified complaint, EC-Council will remove the material in question and make necessary rectifications.

The courseware may contain references to other information resources and security solutions, but such references should not be considered as an endorsement of or recommendation by EC-Council. All product and company names mentioned in this courseware are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

Readers are encouraged to report errors, omissions and inaccuracies to EC-Council at legal@eccouncil.org. If you have any issues, please contact support@eccouncil.org.

NOTICE TO THE READER

EC-Council does not warrant or guarantee any of the products, methodologies, or frameworks described herein or perform any independent analysis in connection with any of the product information contained herein. EC-Council does not assume, and expressly disclaims, any obligation to obtain and include information other than that provided to it by the manufacturer. The reader is expressly warned to consider and adopt all safety precautions that might be indicated by the activities described herein and to avoid all potential hazards. By following the instruction contained herein, the reader willingly assumes all risks in connection with such instructions. EC-Council makes no representations or warranties of any kind, including but not limited to, the warranties of fitness for particular purpose or merchantability, nor are any such representations implied with respect to the material set forth herein, and EC-Council takes no responsibility with respect to such material. EC-Council shall not be liable for any special, consequential, or exemplary damages resulting, in whole or in part, from the reader's use or reliance up this material.

Program Introduction

Today, businesses of all sizes are going online to stay relevant and competitive. Many of these organizations rely solely on their web applications/websites to deliver goods and services. These applications are increasingly serving critical business functions for the customers and performing sophisticated complex business tasks such as real-time sales, transactions, inventory management across multiple vendors including both B2B and B2C e-commerce, workflow and supply chain management, etc. However, these applications also pose huge business risks to the organizations as they hold and process a lot of secret information related to the businesses. The applications have become potent attack vectors for attackers. A successful application-level attack can help attackers in pilfering millions of dollars, affecting business continuity, damaging reputation or even resulting in complete business shutdown.

Organizations' traditional network infrastructure, security processes and platforms such as firewalls, IDS/IPS systems, etc. are ineffective in defending application-level attacks. According to GARTNER report, over 70% of security vulnerabilities exist at the application layer and not at the network or system layer. Implementing traditional security measures can give organizations only a false sense of security at the application level.

According to OWASP, "Software applications are designed and developed with functionality first in mind and security as a distant second or third." The only way to cope with the increasing concerns and business risks associated with insecure applications is to integrate security into the whole software development lifecycle. Coding securely alone does not fully ensure secure software development as a missed requirement or bad web design or insecure architecture can make the application vulnerable to different types of attacks. A comprehensive application security requires security vigilance at requirement gathering, design, implementation and deployment of application.

Certified Application Security Engineer (CASE) is a hands-on, instructor-led, comprehensive application security course, which encompasses security activities involved in all the phases of Software Development Lifecycle (SDLC). The course focuses on secure software development process in the real world. The course will help students gain in-depth knowledge and skills in gathering security requirements, designing secure application architecture, implementing secure coding practices, performing static application security testing (SAST) and dynamic application testing (DAST), and secure deployment.

The CASE program briefs on various application security issues that may arise due to incorrect security strategies incorporated during traditional software development lifecycle and their impact on the businesses. The program also focuses on selecting and implementing the right security strategies, guidelines, and practices that are widely accepted and applicable to most environments used today.

Please visit <https://www.eccouncil.org> for more information.

About EC-Council

The International Council of Electronic Commerce Consultants, better known as EC-Council, was founded in late 2001 to address the need for well-educated and certified information security and e-business practitioners. EC-Council is a global, member-based organization composed of industry and subject matter experts all working together to set the standards and raise the bar in information security certification and education.

EC-Council first developed the *Certified Ethical Hacker* (C|EH) program with the goal of teaching the methodologies, tools, and techniques used by hackers. Leveraging the collective knowledge from hundreds of subject matter experts, the CEH program has rapidly gained popularity around the globe and is now delivered in more than 145 countries by more than 950 authorized training centers. More than 200,000 information security practitioners have been trained.

CEH is the benchmark for many government entities and major corporations around the world. Shortly after CEH was launched, EC-Council developed the *Certified Security Analyst* (ECSA). The goal of the ECSA program is to teach ground-breaking analysis methods that must be applied while conducting advanced penetration testing. The ECSA program leads to the *Licensed Penetration Tester* (LPT). The *Computer Hacking Forensic Investigator* (CHFI) was formed with the same design methodologies and has become a global standard in certification for computer forensics. EC-Council, through its impressive network of professionals and huge industry following, has developed various other programs in information security and e-business. EC-Council certifications are viewed as the essential certifications needed when standard configuration and security policy courses fall short. Providing a true, hands-on, tactical approach to security, individuals armed with the knowledge disseminated by EC-Council programs are security networks around the world and beating hackers at their own game.

Other EC-Council Programs

Security Awareness: Certified Secure Computer User



The purpose of the CSCU training program is to provide students with the necessary knowledge and skills to protect their information assets. This class will immerse students into an interactive environment where they will acquire fundamental understanding of various computer and network security threats such as identity theft, credit card fraud, online banking phishing scams, virus and backdoors, emails hoaxes, sex offenders lurking online, loss of confidential information, hacking attacks and social engineering. More importantly, the skills learnt from the class helps students take the necessary steps to mitigate their security exposure.

Network Defense: Certified Network Defender



Students enrolled in the Certified Network Defender course, will gain a detailed understanding and hands on ability to function in real life situations involving network defense. They will gain the technical depth required to actively design a secure network in your organization. This course gives you the fundamental understanding of the true construct of data transfer, network technologies, software technologies so that you understand how networks operate, understand what software is automating and how to analyze the subject material.

You will learn how to protect, detect and respond to the network attacks. You will learn network defense fundamentals, the application of network security controls, protocols, perimeter appliances, secure IDS, VPN and firewall configuration. You will then learn the intricacies of network traffic signature, analysis and vulnerability scanning which will help you when you design greater network security policies and successful incident response plans. These skills will help you foster resiliency and continuity of operations during attacks.

Ethical Hacking: Certified Ethical Hacker



The Certified Ethical Hacker (CEH) program is the core of the most desired information security training system any information security professional will ever want to be in. The CEH, is the first part of a 3 part EC-Council Information Security Track which helps you master hacking technologies. You will become a hacker, but an ethical one!

As the security mindset in any organization must not be limited to the silos of a certain vendor, technologies or pieces of equipment, This course was designed to provide you with the tools and techniques used by hackers and information security professionals alike to break into an organization. As we put it, "To beat a hacker, you need to think like a hacker".

This program will immerse you into the Hacker Mindset so that you will be able to defend against future attacks. It puts you in the driver's seat of a hands-on environment with a systematic ethical hacking process.

Here, you will be exposed to an entirely different way of achieving optimal information security posture in their organization; by hacking it! You will scan, test, hack and secure your own systems. You will be taught the Five Phases of Ethical Hacking and taught how you can approach your target and succeed at breaking in every time! The five phases include Reconnaissance, Gaining Access, Enumeration, Maintaining Access, and covering your tracks.

Penetration Testing: EC-Council Certified Security Analyst



EC-Council Certified Security Analyst

™ EC Council Certified Security Analyst, ECSA complements the Certified Ethical Hacker, CEH certification by exploring the analytical phase of ethical hacking. While the Certified Ethical Hacker certification exposes the learner to hacking tools and technologies, the Certified Security Analyst course takes it a step further by exploring how to analyze the outcome from these tools and technologies. Through groundbreaking penetration testing methods and techniques, this pen testing security training course helps students perform the intensive assessments required to effectively identify and mitigate risks to the security of the infrastructure.

This makes the Certified Security Analyst “Penetration Training” a relevant milestone toward achieving EC Council’s Licensed penetration Tester, which also ingrains the learner in the business aspect of penetration testing. The Licensed Penetration Tester standardizes the knowledge base for penetration testing professionals by incorporating the best practices followed by experienced experts in the field. The objective of Certified Security Analyst is to add value to experienced security professionals by providing security training that will help them analyze the outcomes of their Vulnerability Assessments. Penetration Testing Training leads the learner into the advanced stages of ethical hacking.

Penetration Testing: Licensed Penetration Tester (Master)



The LPT (Master) is the world's first fully online, remotely proctored LPT (Master) practical exam, which challenges the candidates through a grueling 18 hours of performance based, hands-on exam categorized into three practical exams for six-hour duration each, which will test your perseverance and focus by forcing you to outdo yourself with each new challenge. The exam requires the candidates to demonstrate a methodical approach to test and validate security defenses. The LPT (Master) exam is developed with close collaboration with SMEs and practitioners around the world after a thorough job role, job task, and skills-gap analysis.

Computer Forensics: Computer Hacking Forensic Investigator



™ CHFI is a comprehensive course covering major forensic investigation scenarios that enables students to acquire necessary hands-on experience regarding various forensic investigation techniques. Learn how to utilize standard forensic tools to successfully carryout a computer forensic investigation, which can better prepare you to aid in the prosecution of perpetrators.

EC-Council's CHFI certifies individuals in the specific security discipline of computer forensics from a vendor-neutral perspective. The CHFI certification will fortify the application knowledge of law enforcement personnel, system administrators, security officers, defense and military

personnel, legal professionals, bankers, security professionals, and anyone who is concerned about the integrity of the network infrastructure.

Incident Handling: EC-Council Certified Incident Handler



The EC-Council Certified Incident Handler program is designed to provide the fundamental skills to handle and respond to the computer security incidents in an information system. The course addresses various underlying principles and techniques for detecting and responding to current and emerging computer security threats. Students will learn how to handle various types of incidents, risk assessment methodologies, and various laws and policy related to incident handling. After attending the course, they will be able to create incident handling and response policies and deal with various types of computer security incidents. Network Security Administration: EC-Council Certified Network Administrator.

Management: Certified Chief Information Security Officer



The Certified Chief Information Security Officer (CCISO) program was developed by EC-Council to fill a knowledge gap that existed in the information security industry. Most information security certifications focus on specific tools or practitioner capabilities. At the time that the CCISO program was developed, no certification existed to recognize the knowledge, skills, and aptitudes required for an experienced information security professional to perform the duties of an effective, competent CISO. In fact, at the time this program was developed, many questions existed about what a CISO really is and the value that this role contributes to an organization.

The CCISO Body of Knowledge helps to define the role of the CISO and to answer the questions about the contributions this person makes in an organization. EC-Council enhances this information through training opportunities conducted as instructor-led or self-study modules to ensure candidates have a complete understanding of the role. Then, EC-Council evaluates the knowledge of CCISO candidates with a rigorous exam that tests competence across five domains with which a seasoned security leader should be familiar.

CASE Exam Information

CASE Exam Details	
Exam Title	Certified Application Security Engineer (CASE) - JAVA
Exam Code	312-96
Availability	EC-Council Exam Portal (please visit https://www.eccexam.com)
Duration	2 Hours
Questions	50
Passing Score	Please refer https://cert.eccouncil.org/faq.html

Please visit <https://cert.eccouncil.org> for more information.

DO NOT COPY
badalshiva@gmail.com

Table of Contents

Module 01: Understanding Application Security, Threats, and Attacks.....	01
Module Objectives	3
What is a Secure Application	3
Need for Application Security	4
Most Common Application Level Attacks	6
SQL Injection Attacks.....	7
Cross-site Scripting (XSS) Attacks	7
Parameter Tampering	8
Directory Traversal	9
Cross-site Request Forgery (CSRF) Attack	9
Denial-of-Service (DoS) Attack	10
Session Attacks.....	11
Why Applications become Vulnerable to Attacks.....	13
Common Reasons for Existence of Application Vulnerabilities	13
Common Flaws Existed due to Insecure Coding Techniques	14
Improper Input Validation	14
Insufficient Transport Layer Protection	15
Improper Error Handling	15
Insecure Cryptographic Storage	16
Broken Authentication and Session Management.....	16
Unvalidated Redirects and Forwards.....	17
Insecure Direct Object References	17
Failure to Restrict URL Access	18
What Constitutes a Comprehensive Application Security?	18
Application Security Frame	19
3W's in Application Security.....	20
Insecure Application: A Software Development Problem	21
Solution: Integrating Security in Software Development Life Cycle (SDLC).....	21
Functional vs Security Activities in SDLC.....	22

Advantages of Integrating Security in SDLC	22
Software Security Standards, Models, and Frameworks.....	23
The Open Web Application Security Project (OWASP)	23
OWASP TOP 10 Attacks-2017	24
The Web Application Security Consortium (WASC)	24
WASC Threat Classification	25
Software Security Framework.....	25
BSIMM vs OpenSAMM	26
Module Summary.....	27
Module 02: Security Requirements Gathering	29
Module Objectives	31
Importance of Gathering Security Requirements	31
Security Requirements	32
Gathering Security Requirements	32
Why We Need Different Approach for Security Requirements Gathering	33
Key Benefits of Addressing Security at Requirement Phase	33
Stakeholders Involvement in Security Requirements Gathering	34
Characteristics of Good Security Requirement: SMART.....	34
Types of Security Requirements.....	35
Security Requirement Engineering (SRE).....	36
SRE Phases	36
Common Mistakes Made in Each Phase of SRE	38
Different Security Requirement Engineering Approaches/Model	39
Abuse Case and Security Use Case Modeling	39
Abuse Cases.....	40
Threatens Relationship.....	40
Abuse Case Modeling Steps	41
Abuse Cases: Advantages and Disadvantages.....	41
Abuse Case Template	42
Security Use Cases.....	42
Security Use Cases are Abuse Case Driven.....	43
Modeling Steps for Security Use Cases	43

Mitigates Relationship.....	44
Abuse Case vs Security Use Case	44
Security Use Case: Advantages and Disadvantages	45
Security Use Case Template	45
Security Use Case Guidelines	46
Effectiveness of Abuse and Security Case	51
Abuser and Security Stories	52
Textual Description Template: Abuser Stories and Security Stories	53
Effectiveness of Abuser and Security Stories	54
Abuser Stories: Advantages and Disadvantages	55
Security Quality Requirements Engineering (SQUARE)	55
SQUARE Effectiveness	56
SQUARE Process	57
SQUARE: Advantages and Disadvantages	58
Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)	59
OCTAVE Effectiveness.....	60
OCTAVE Steps	61
OCTAVE: Advantages and Disadvantages.....	61
Module Summary.....	62
 Module 03: Secure Application Design and Architecture.....	63
Module Objectives	65
Relative Cost of Fixing Vulnerabilities at Different Phases of SDLC.....	66
Secure Application Design and Architecture	67
Goal of Secure Design Process.....	67
Secure Design Actions	68
Secure Design Principles	68
Threat Modeling	75
Threat Modeling Phases	77
Threat Modeling Process.....	77
Identify Security Objective	78
Create an Application Overview.....	79

Draw the End-to-End Deployment Architecture	79
Identify Various User Roles	80
Identify Use Cases Scenarios	80
Identify Technologies	81
Identify Application Security Mechanisms	81
Decompose Application	82
Prepare and Document Threat Model Information	82
Identify the External Dependencies	83
Identify the Entry Points.....	84
Identify the Assets.....	85
Identify the Trust Levels	87
Define Trust Levels to Entry points	88
Define Trust Levels to Assets.....	88
Perform Application Modelling using Data Flow Diagrams (DFDs)	89
Determine the Threats: Identify the Goal of an Attacker and Create Threat Profile.....	91
Determine the Threats: Create a Security Profile	92
Identify the Threats.....	94
Determine Countermeasures and Mitigation Security Controls	95
Document the Threats.....	96
Rating the Threats	96
Secure Application Architecture	98
Design Secure Application Architecture	98
Module Summary.....	99
Module 04: Secure Coding Practices for Input Validation.....	101
Module Objectives	103
Input Validation Pattern	103
Validation and Security Issues	104
Impact of Invalid Data Input	104
Data Validation Techniques	105
Input Validation using Frameworks and APIs	106
Open Source Validation Framework for Java.....	107

Servlet Filters	107
Validation Filters for Servlet	108
Data Validation using OWASP ESAPI.....	108
Data Validation: Struts Framework.....	109
Struts Validator.....	109
Struts Validation and Security	110
Data Validation using Struts Validator	110
Avoid Duplication of Validation Forms.....	111
Secure and Insecure Struts Validation Code	111
Struts Validator Class.....	112
Secure and Insecure Code for Struts Validator Class	112
Enable the Struts Validator	113
Secure and Insecure Struts Validator Code	113
Struts2 Framework Validator	114
Struts2 Framework: Built-in Data Validators.....	115
Struts2 Framework Annotation Based Validators.....	116
Struts2 Custom Validation: Workflow Interceptor	116
Struts2 Ajax Validation: jsonValidation Interceptor.....	117
Data Validation: Spring Framework.....	117
Spring Validator	118
Data Validation: Spring MVC Framework	118
Implementing Validator.....	119
JSR 380 Bean Validator API.....	119
Configuring JSR 380	120
Custom Validator Implementation in Spring	120
Spring Validation and Security	121
Input Validation Errors.....	121
Improper Sanitization of Untrusted Data	122
Improper Validation of Strings	123
Improper Logging of User Inputs.....	124
Improper Incorporation of Malicious Inputs into Format Strings	125
Inappropriate Use of Split Characters in Data Structures	125
Improper Validation of Non-Character Code Points	126

Improper Use of String Modification.....	127
Improper Comparison of Locale-dependent Data	128
Common Secure Coding Practices	129
SQL Injection.....	129
Prepared Statement	130
Stored Procedures.....	130
Stored Procedure for Securing Input Validation	131
Cross-site Scripting (XSS)	132
Whitelisting vs Blacklisting	132
Regular Expressions.....	134
Character Encoding	135
Cross-site Scripting (XSS) Countermeasures	136
HTML Encoding.....	137
HTML Encoding using ESAPI Encoder.....	138
Cross-site Request Forgery (CSRF).....	138
Directory Traversal	139
Parameter Manipulation and Countermeasures	141
Protecting Application from Log Injection Attack	142
XML Injection.....	142
Command Injection	143
LDAP Injection	143
XML External Entity Attack.....	144
Unrestricted File Upload Attack	146
Prevent Unrestricted File Upload: Validate File Extension	146
Injection Attacks Countermeasures	147
CAPTCHA.....	147
Best Practices for Input Validation	150
Module Summary.....	151
Module 05: Secure Coding Practices for Authentication and Authorization.....	153
Module Objectives	155
Introduction to Authentication.....	155

Java Container Authentication	156
Authentication Mechanism Implementation	157
Types of Authentication	157
Declarative vs Programmatic Authentication	158
Declarative Security Implementation.....	158
Programmatic Security Implementation	159
Java EE Authentication Implementation Example	160
Basic Authentication.....	160
How to Implement Basic Authentication?	161
Form-based Authentication	161
Form-based Authentication Implementation	162
Implementing Kerberos-Based Authentication.....	162
Secured Kerberos Implementation	163
Client Certificate Authentication.....	164
Certificate Generation with Keytool.....	165
Implementing Encryption and Certificates in Client Application	167
Authentication Weaknesses and Prevention.....	167
Brute Force Attack.....	168
Web-based Enumeration Attack.....	168
Weak Password Attacks	169
Introduction to Authorization.....	169
JEE Based Authorization	171
Access Control Model	171
Discretionary Access Control (DAC)	174
Mandatory Access Control (MAC)	174
Role-based Access Control (RBAC)	175
Servlet Container	175
Authorizing Users by Servlets.....	176
EJB Authorization	177
EJB Authorization Controls.....	177
Declarative Security with EJBs.....	178
Programmatic Security with EJBs	179
Java Authentication and Authorization (JAAS)	180

JAAS Features	181
JAAS Architecture	181
Pluggable Authentication Module (PAM) Framework	182
JAAS Classes.....	182
JAAS Subject and Principal	183
Authentication in JAAS	184
Authorization in JAAS	185
Subject Methods doAs() and doAsPrivileged().....	186
Impersonation in JAAS.....	187
JAAS Permissions	187
LoginContext in JAAS	188
Creating LoginContext	188
LoginContext Instantiation	189
JAAS Configuration	189
Locating JAAS Configuration File	190
JAAS CallbackHandler and Callbacks	190
Login to Standalone Application	191
JAAS Client	192
LoginModule Implementation in JAAS	192
Methods Associated with LoginModule.....	193
LoginModule Example	194
Phases in Login Process	194
Java EE Security	195
Java EE Application Architecture	196
Java EE Servers as Code Hosts	196
Declaring Roles	197
HTTP Authentication Schemes	198
Authorization Common Mistakes and Countermeasures.....	199
Common Mistakes.....	199
Authentication and Authorization in Spring Security Framework	201
Spring Security Framework	201
Spring Security Modules.....	202
Spring Authentication.....	203

Storing Username and Password	203
Securing Authentication Provider	204
Implementing HTTP Basic Authentication.....	205
Form-based Authentication	205
Implementing Digest Authentication.....	206
Security Expressions	207
URL-based Authorization.....	208
JSP Page Content Authorization	208
JSP Page Content Authorization with Domain Object's ACL	209
Method Authorization.....	209
Configuring Anonymous Login	210
Logout Feature Configuration	210
Remember-Me Authentication	211
Integrating Spring Security with JAAS	211
Spring JAAS Implementation	212
Defensive Coding Practices against Broken Authentication and Authorization	213
Do Not Store Password in Java String Object	213
Avoid Cookie based Remember-Me Use Persistent Remember-Me	214
Implement Appropriate Session Timeout	214
Prevent Session Stealing by Securing SessionID Cookie.....	215
Secure Development Checklists: Broken Authentication and Session Management	216
Module Summary.....	216
 Module 06: Secure Coding Practices for Cryptography.....	217
Module Objectives	219
Java Cryptography.....	219
Need for Java Cryptography	220
Java Security with Cryptography	220
Java Cryptography Architecture (JCA)	221
Java Cryptography Extension (JCE).....	222
Encryption and Secret Keys.....	223
Attack Scenario: Inadequate/Weak Encryption	223

Encryption: Symmetric and Asymmetric Key	224
Encryption/Decryption Implementation Methods.....	224
SecretKeys and KeyGenerator.....	225
Implementation Methods of KeyGenerator Class.....	225
Creating SecretKeys with KeyGenerator Class	226
Cipher Class.....	227
The Cipher Class	227
Implementation Methods of Cipher Class.....	228
Insecure Code for Cipher Class using DES Algorithm	228
Secure Code for Cipher Class using AES Algorithm	229
Digital Signatures	229
Attack Scenario: Man-in-the-Middle Attack.....	230
Digital Signatures.....	230
The Signature Class.....	231
Implementation Methods of Signature Class	231
The SignedObjects.....	232
Implementing Methods of SignedObjects.....	232
The SealedObjects.....	233
Implementation Methods of SealedObject.....	233
Insecure and Secure Code for Signed/Sealed Objects	234
Java XML Digital Signature	235
Secure Socket Layer (SSL)	238
Java Secure Socket Extension (JSSE)	239
SSL and Security: Example 1	239
SSL and Security: Example 2	240
JSSE and HTTPS.....	241
Insecure HTTP Server Code	242
Secure HTTP Server Code	243
Key Management.....	243
Attack Scenario: Poor Key Management.....	244
Keys and Certificates	244
Key Management System.....	245
KeyStore	245

Implementation Method of KeyStore Class	246
KeyStore: Persistent Data Stores.....	246
Key Management Tool: KeyTool.....	247
Digital Certificates.....	248
Certification Authorities	249
Signing Jars	250
Signing JAR Tool: Jarsigner	251
Signed Code Sources.....	252
Insecure Code for Signed Code Sources.....	253
Secure Code for Signed Code Sources.....	254
Hashing	254
Hashing Algorithms	255
Securing Hashed Password with Salt.....	256
Implementing Hashing with Salt in Spring Security.....	257
Java Card Cryptography	258
Spring Security: Crypto Module	261
Crypto Module.....	261
Spring Security Crypto Module.....	262
Implementing BCryptPasswordEncoder()	263
Configuring BCryptPasswordEncoder() in Spring Security	263
JavaScript Object Signing and Encryption (JOSE)	264
Attacks against JWT, JWS and JWE	264
Implementing JWS using Jose4J	265
Implementing JWE using Jose4J	265
Implementing JWK using Jose4J	266
Dos and Don'ts in Java Cryptography	266
Dos and Don'ts	267
Best Practices for Java Cryptography.....	269
Module Summary.....	270
Module 07: Secure Coding Practices for Session Management.....	271
Module Objectives	273
Session Management.....	273

Session Tracking	274
Session Tracking Methods.....	274
Session Management in Spring Security.....	278
Spring Session Management	279
Session Management using Spring Security.....	280
Restricting Concurrent Sessions per User using Spring Security.....	280
Controlling Session Timeout.....	281
Prevent using URL Parameters for Session Tracking.....	282
Prevent Session Fixation with Spring Security	282
Use SSL for Secure Connection.....	283
Session Vulnerabilities and their Mitigation Techniques.....	284
Session Vulnerabilities.....	284
Types of Session Hijacking Attacks.....	285
Countermeasures for Session Hijacking	286
Countermeasures for Session ID Protection	287
Best Practices and Guidelines for Secured Sessions Management	287
Best Coding Practices for Session Management	288
Checklist to Secure Credentials and Session IDs.....	291
Guidelines for Secured Session Management	291
Module Summary.....	292
 Module 08: Secure Coding Practices for Error Handling.....	293
Module Objectives.....	295
Introduction to Exceptions	295
Exception and Error Handling.....	296
Example of an Exception	297
Handling Exceptions in Java	298
Exception Classes Hierarchy	299
Exceptions and Threats	300
Erroneous Exceptional Behaviors	300
Suppressing or Ignoring Checked Exceptions.....	301
Disclosing Sensitive Information	302
Logging Sensitive Data.....	303

Restoring Objects to Prior State, if a Method Fails	304
Avoid using Statements that Suppress Exceptions	305
Prevent Access to Untrusted Code that Terminates JVM	306
Never Catch <code>java.lang.NullPointerException</code>	307
Never Allow methods to Throw <code>RuntimeException</code> , <code>Exception</code> , or <code>Throwable</code>	307
Never Throw Undeclared Checked Exceptions	308
Never Let Checked Exceptions Escape from Finally Block.....	309
Dos and Don'ts in Error Handling.....	310
Dos and Don'ts in Exception Handling	310
Avoid using Log Error and Throw exception at Same Time.....	313
Spring MVC Error Handling	313
Handling Controller Exceptions with <code>@ExceptionHandler</code> Annotation	314
Handling Controller Exceptions with <code>HandlerExceptionResolver</code>	315
Spring MVC: Global Exception Handling.....	315
Global Exception Handling: <code>HandlerExceptionResolver</code>	316
Mapping Custom Exceptions to Statuscode with <code>@ResponseStatus</code>	317
Configure Custom Error Page in Spring MVC	317
Exception Handling in Struts 2	318
Best Practices for Error Handling	319
Best Practices for Handling Exceptions in Java.....	320
Introduction to Logging.....	322
Logging in Java.....	323
Example for Logging Exceptions	323
Logging Levels.....	324
Logging using Log4j	324
Log4j and Java Logging API	325
Java Logging using Log4j	326
Secure Coding in Logging	326
Vulnerabilities in Logging	327
Logging: Vulnerable Code and Secure Code	328
Secured Practices in Logging	331
Module Summary.....	332

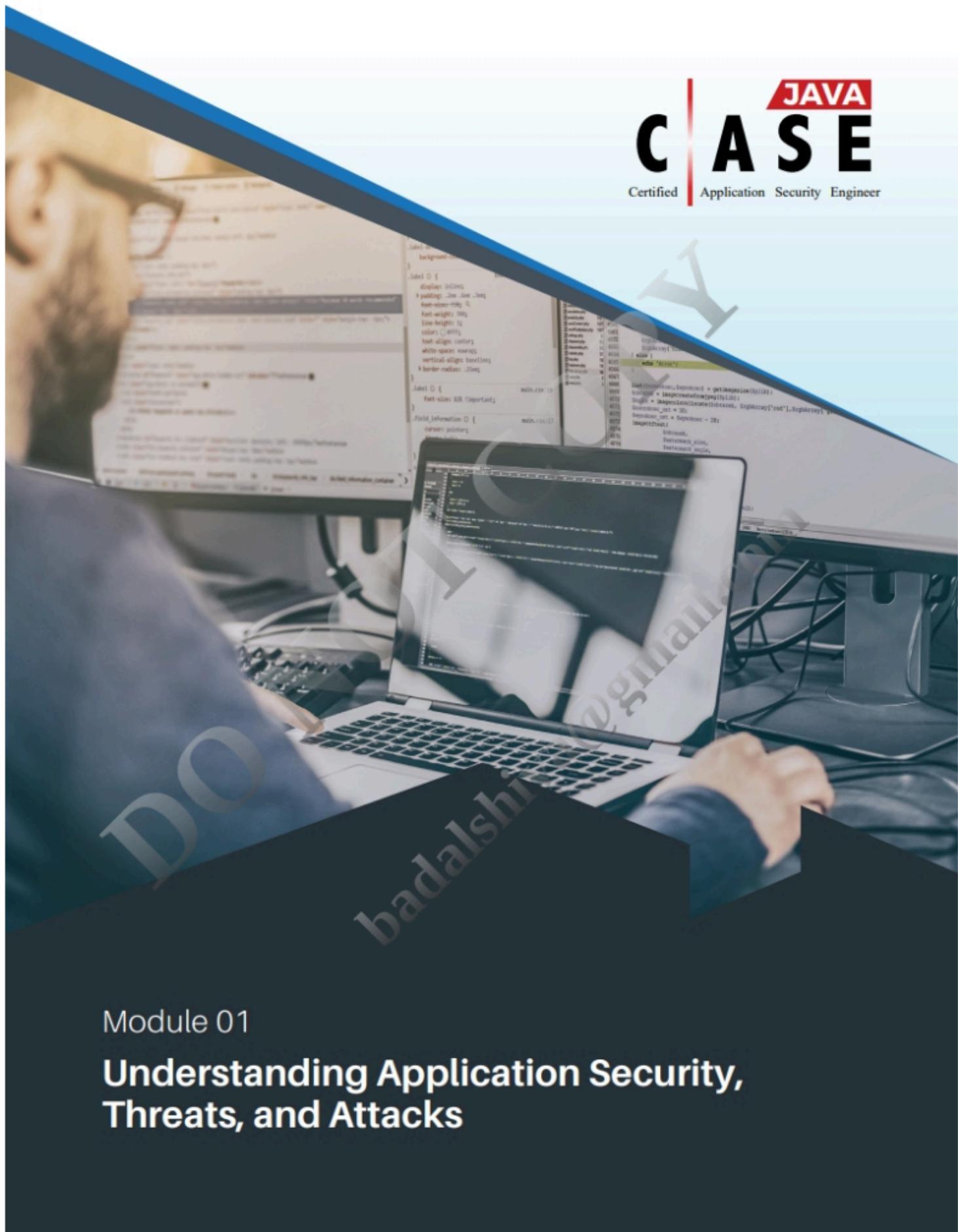
Module 09: Static and Dynamic Application Security Testing (SAST & DAST)	333
Module Objectives	335
Static Application Security Testing.....	335
Static Application Security Testing (SAST)	336
Objectives of SAST.....	336
Why SAST.....	337
Skills required for SAST.....	337
What to look for in SAST.....	338
Common Vulnerabilities Identified Through SAST	338
Types of SAST	339
Where does Secure Code Review Fit in SDLC?	339
SAST Steps	340
SAST Activities- flow Chart	340
Recommendation for Effective SAST.....	341
SAST Deliverable.....	341
Automated Source Code Analysis	342
Selecting Static Analysis Tool	344
Manual Secure Code Review.....	345
Manual Secure Code Review for Most Common Vulnerabilities.....	345
Code Review for PCI DSS Compliance	346
Code Review for Blacklisting Validation Approach	347
Code Review for Client-side Validation Approach	347
Code Review for Non-parametrized SQL Query.....	348
Code Review for XSS Vulnerability	348
Code Review for Weak Password Authentication	349
Code Review for Hard-coded Passwords	349
Code Review for Empty Password in Connection String	350
Code Review for Insecure Basic Authentication	350
Code Review for Open Redirect	351
Code Review for Insecure LDAP Authentication	351
Code Review for Insecure Authorization Mechanism.....	352
Code Review for Weak Password Length.....	352

Code Review for use of Weak Hashing Algorithm.....	353
Code Review for Use of Weak Random Number Generator.....	353
Code Review for Use of Insecure PBE Work Factor.....	354
Code Review for use of Weak Encryption Algorithm	354
Code Review for Use of Insufficient Encryption Key Size.....	355
Code Review for Unsafe Decoding	355
Code Review for Use of SSL.....	356
Code Review for OS Command Injection	356
Code Review for LDAP Injection.....	357
Code Review for XML Injection	357
Code Review for Unsafe use of request parameter to execute SQL Query	358
Code Review for Insecure File Upload.....	358
Code Review for Directory Traversal.....	359
Code Review for Sensitive Information Exposure	359
Code Review for Sensitive Information Leakage.....	360
Code Review for Generic Exception Throwing and Catching.....	360
Code Review for Cookies Vulnerable to Client-side Scripts Attacks	361
Code Review for Cookies Vulnerable to CSRF Attacks	361
Code Review for Enabling Directory Listing	362
Code Review: Check List Approach	362
Sample Checklist.....	363
SAST Finding.....	366
SAST Report.....	367
SAST Reporting	367
Dynamic Application Security Testing (DAST).....	368
Types of DAST	369
SAST Vs DAST.....	370
Automated Application Vulnerability Scanning Tools	370
Web Application Security Scanners.....	371
Additional Web Application Vulnerability Scanners	372
Proxy-based Security Testing Tools	372
Burp Suite	373
OWASP Zed Attack Proxy (ZAP)	373

Additional Proxy-based Security Testing Tools	374
Choosing Between SAST and DAST	374
Module Summary.....	375
Module 10: Secure Deployment and Maintenance.....	377
Module Objectives	379
Secure Deployment.....	379
Prior Deployment Activity.....	380
Check the Integrity of Application Package Before Deployment	380
Review the Deployment Guide Provided by the Software Vendor.....	380
Deployment Activities: Ensuring Security at Various Levels.....	381
Ensuring Security at Host Level.....	381
Check and Configure the Security of Machine Hosting Web Server, Application Server, Database Server and Network Devices	382
Physical Security.....	382
Host Level Security	383
Ensuring Security at Network Level	383
Network Level Security.....	384
Ensuring Security at Application Level	385
Web Application Firewall (WAF)	386
Ensuring Security at Web Container Level (Tomcat)	388
Install and Configure Tomcat Securely.....	388
Remove Server Banner	389
Start Tomcat with Security Manager.....	389
Configure Default Servlet Not to Serve Index Pages	390
Replace Default Error Page.....	390
Replace Default server.xml.....	391
Protect Shutdown Port.....	391
Restrict Access to Tomcat Manager Applications	392
Protecting Resources with Realms	392
Store Passwords as Digest.....	393
Do Not Run Tomcat as Root	393
Configure Restricted Datasets.....	394

Session Handling using App Mode in Tomcat	394
Role Based Security	395
Securing Tomcat at Network level	396
Java Runtime Security Configurations.....	398
Tomcat General Security Setting.....	399
Verify Trace Element Setting in sever.xml.....	401
Verify CustomError Settings in web.xml	401
Verify maxPostSize Setting.....	402
Tomcat Security Checklist	402
Checklist for Security Configuration in server.xml File in Apache Tomcat.....	403
Tomcat High Availability.....	403
Best Practices for Securing Tomcat	405
Ensuring Security in Oracle	406
Oracle Database General Security Overview	406
Methods of Authentication in Oracle	407
Authentication by Oracle Database	407
Oracle Security Features	408
Default Database Installation and Configuration Security	409
Managing User Accounts Securely for the Site	410
Securing User Accounts	410
Password Management.....	412
Lock all Expired Accounts	412
Assign Users to Password Profile	413
Disable Remote Operating System Authentication.....	413
Securing Data.....	414
Restrict Access to Operating System Directories	414
Securing Database Installation and Configuration	415
Securing Network	415
How to Configure Encryption on the Client and the Server	416
Control Access Data.....	416
Virtual Private Database	417
Oracle Label Security	418
Database Vault	418

Audit Vault.....	420
Built-in Audit Tools.....	420
Recommended Audit Settings.....	424
Security Maintenance and Monitoring	424
Post Deployment Activities: Security Maintenance and Monitoring	425
Security Maintenance Activities at OS Level	425
Security Maintenance Activities at Web Container Level	426
Security Maintenance Activities at Application Level	426
Module Summary.....	427
References.....	429



Module 01

Understanding Application Security, Threats, and Attacks

DO NOT COPY
This page is intentionally left blank.
badalshiva@gmail.com

Module Objectives



- 1 Understand the Need and Benefits of Application Security
- 2 Discuss the Most Common Application-level Attacks
- 3 Discuss the Common Reasons Behind the Existence of Application-level Vulnerabilities
- 4 Understand Various Components of Comprehensive Application Security
- 5 Understand the Need and Advantages of Integrating Security in Software Development Life Cycle (SDLC)
- 6 Differentiate Functional vs Security Activities in SDLC
- 8 Discuss Various Software Security Reference Standards, Models, and Frameworks

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

What is a Secure Application



- An application is said to be secure when it ensures **confidentiality**, **integrity** and **availability** of its restricted resources
- Restricted resource is any **object**, **data**, **feature**, or **function** of an application designed to be accessed by only **authorized** users



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Need for Application Security



Organizations are increasingly using **web applications** to provide high value business functions to their customers such as real-time sales, transactions, inventory management across multiple vendors including both B-B and B-C e-commerce, workflow and supply chain management, etc.



Attackers exploit vulnerabilities in the applications to **launch various attacks** and **gain unauthorized access to resources**.

A successful application level attack may result into:

- | | |
|--------------------------------------|---|
| 1 Financial Loss | 4 Disclosure of Business Information |
| 2 Affects Business Continuity | 5 Damages Reputation |
| 3 Closure of Business | 6 Fraudulent Transactions |

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.



"A vulnerability in an application will allow a malicious user to exploit a network or a host."

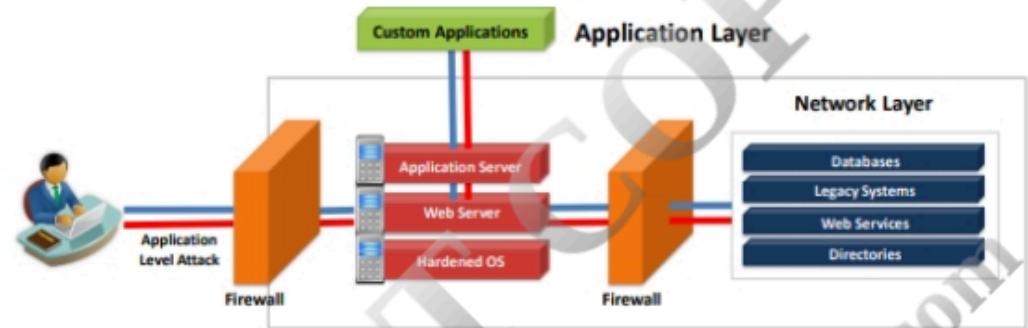
--Carlos Lyons, Corporate Security, Microsoft

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Need for Application Security (Cont'd)



- It is a common myth that perimeter security controls such as firewall and IDS systems can secure your application but it is not true as these controls are **not effective** to defend application layer attacks
- This is because port 80 and 443 are generally open on perimeter devices for legitimate web traffic, which attackers can use to **exploit the application level vulnerabilities** and get into the network



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



"Nearly 75% of all attacks on information security are directed to the web application layer."

-- Gartner Report

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



"2/3 of all web applications are vulnerable."

-- Gartner Report

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.



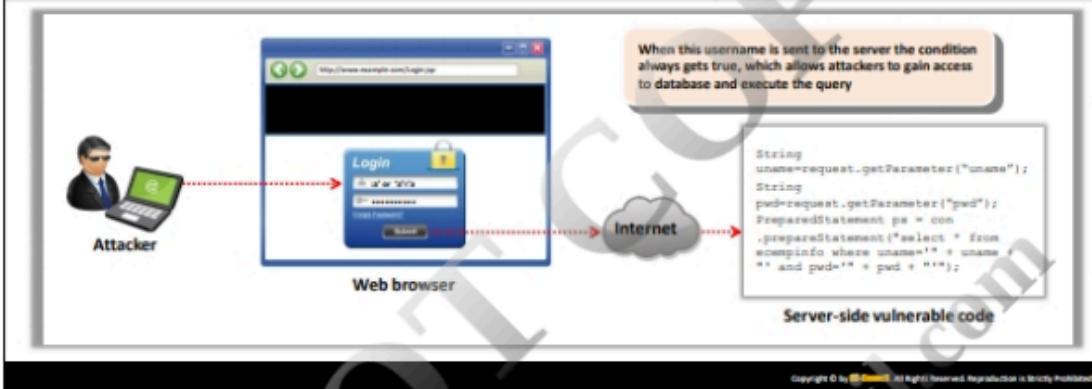
Most Common Application Level Attacks

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

SQL Injection Attacks



- SQL injection attacks use a **series of malicious SQL queries** to directly manipulate the database
- An attacker can use a vulnerable web application to **bypass normal security measures** and obtain direct access to valuable data
- SQL injection attacks can often be executed from the **address bar**, from within application fields, and through queries and searches
- This attack is possible only when the application executes **dynamic SQL statements** and stores procedures with **arguments** based on the user input

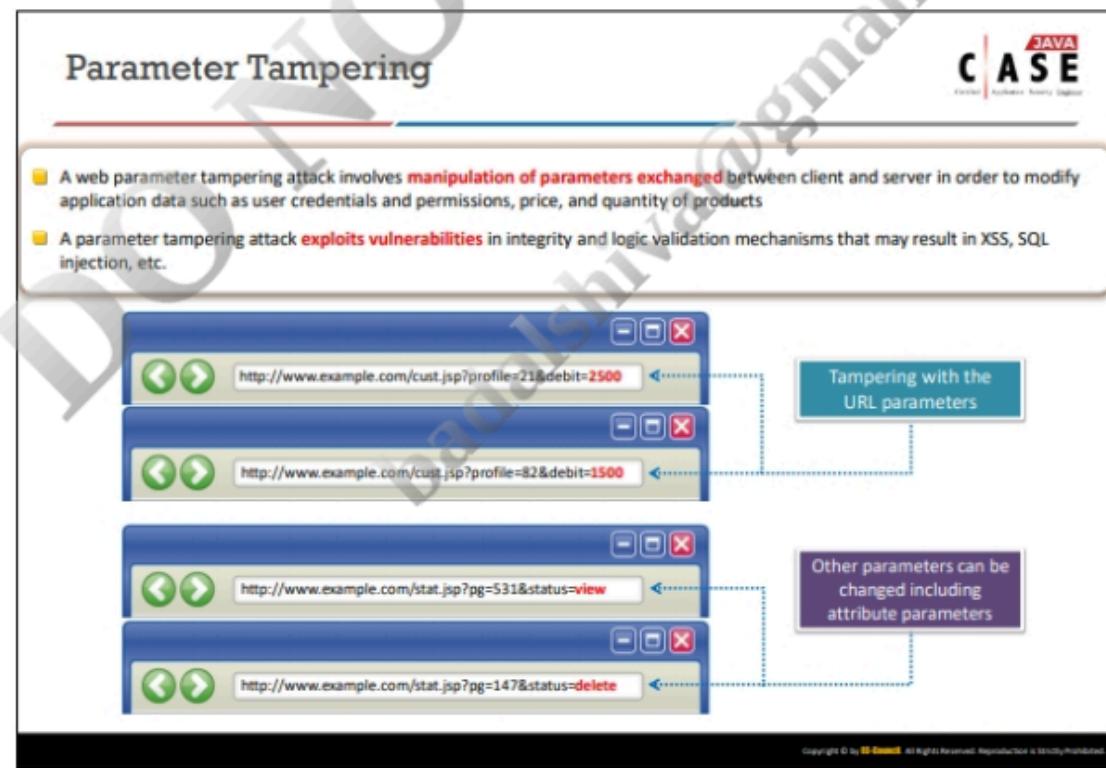
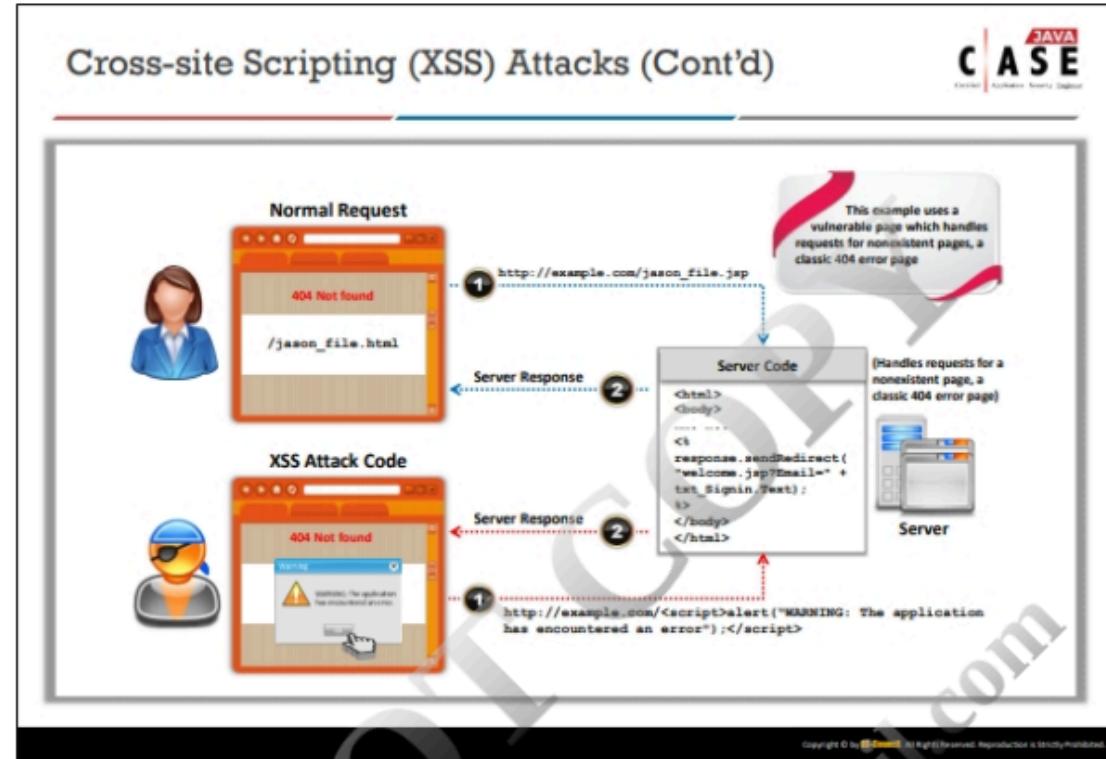


Cross-site Scripting (XSS) Attacks



- Cross-site scripting ('XSS' or 'CSS') attacks **exploit vulnerabilities in dynamically generated web pages**, which enables malicious attackers to inject client-side script into web pages viewed by other users
- It occurs when **invalidated input data** is included in dynamic content that is sent to a user's web browser for rendering
- Attackers inject malicious JavaScript, VBScript, ActiveX, HTML, or Flash for execution on a victim's system by hiding it **within legitimate requests**

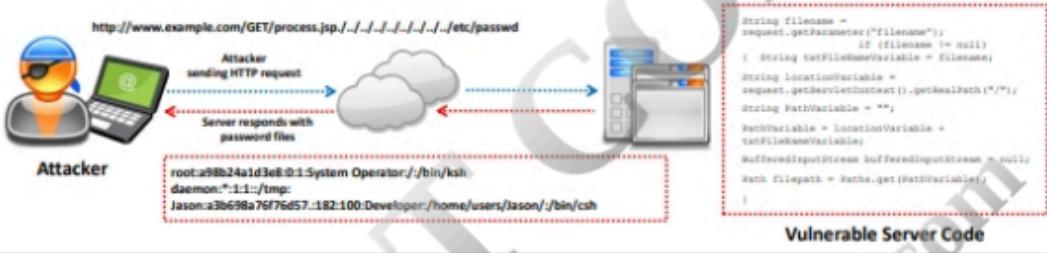




Directory Traversal



- Directory traversal allows attackers to **access restricted directories** including application source code, configuration, and critical system files and execute commands outside the webserver's root directory
- Accessing files located outside the **web publishing directory** using directory traversal
- Attackers can **manipulate variables** that reference files with "dot-dot-slash (..)" sequences and its variations
 - <http://www.example.com/process.jsp=../../../../etc/passwd>
 - <http://www.example.com/../../../../etc/passwd>



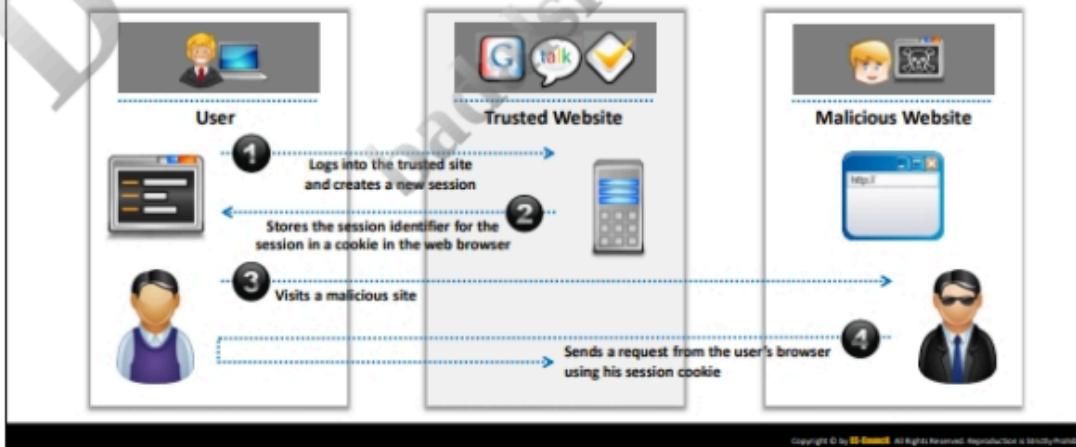
Vulnerable Server Code

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

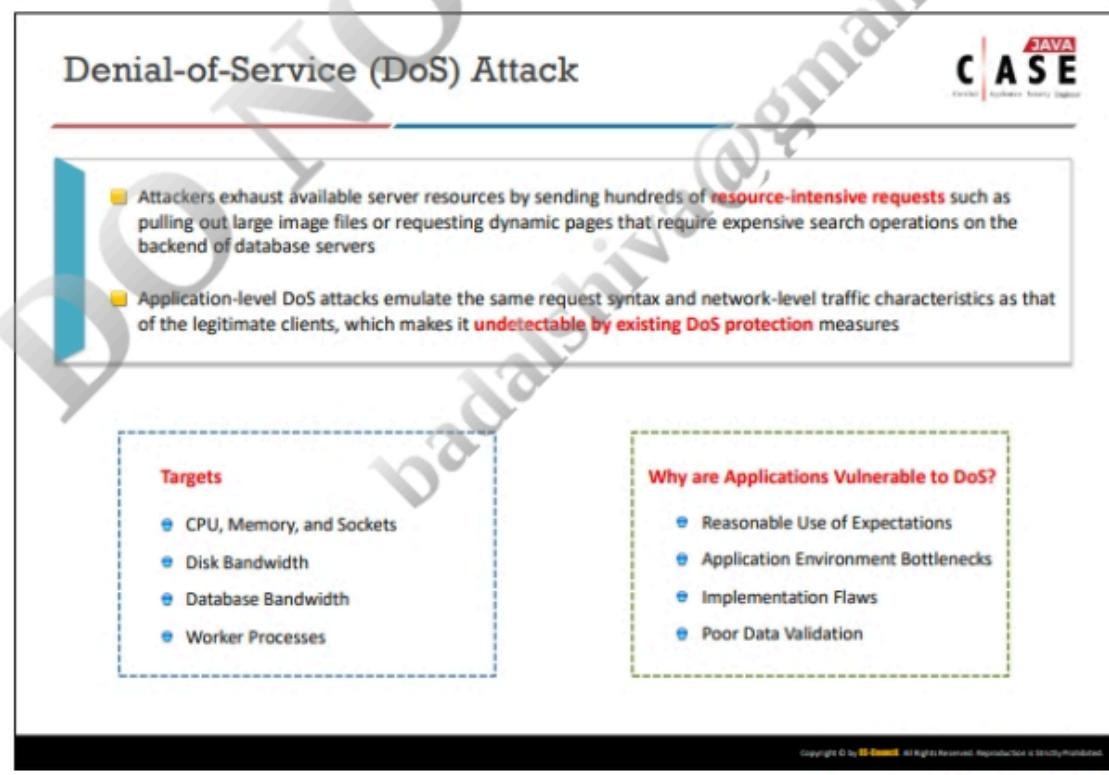
Cross-site Request Forgery (CSRF) Attack



- Cross-site request forgery (CSRF) attacks **exploit web page vulnerabilities** that allow an attacker to force an unsuspecting user's browser to send malicious requests, which they did not intend to send
- The user, who is the victim **holds an active session** with a trusted site and simultaneously visits a malicious site, which **injects an HTTP request** for the trusted site into the victim user's session, compromising its integrity



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Denial-of-Service (DoS) Attack: Examples



User Registration DoS

- The attacker could create a program that submits the registration forms repeatedly; adding a **large number of spurious users** to the application

User Enumeration

- If **application states**, which are a part of the username/password pair is incorrect, an attacker can automate the process of trying **common usernames** from a **dictionary file** to enumerate the users of the application

Login Attacks

- The attacker may overload the login process by continually sending login requests that require the presentation tier to access the authentication mechanism, rendering it **unavailable** or **unreasonably slow** to respond

Account Lock Out Attacks

- The attacker may enumerate usernames through another vulnerability in the application and then attempt to authenticate the site using **valid usernames and incorrect passwords** which will lock out the accounts after a specified number of failed attempts. At this point legitimate users will not be able to use the site

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Session Attacks: Cookie Poisoning Attacks



- Cookies are used to maintain session state in the otherwise stateless HTTP protocol

Modify the Cookie Content

- Cookie poisoning attacks involve modification of the contents of a cookie (personal information stored in a web user's computer) in order to **bypass security mechanisms**

Inject the Malicious Content

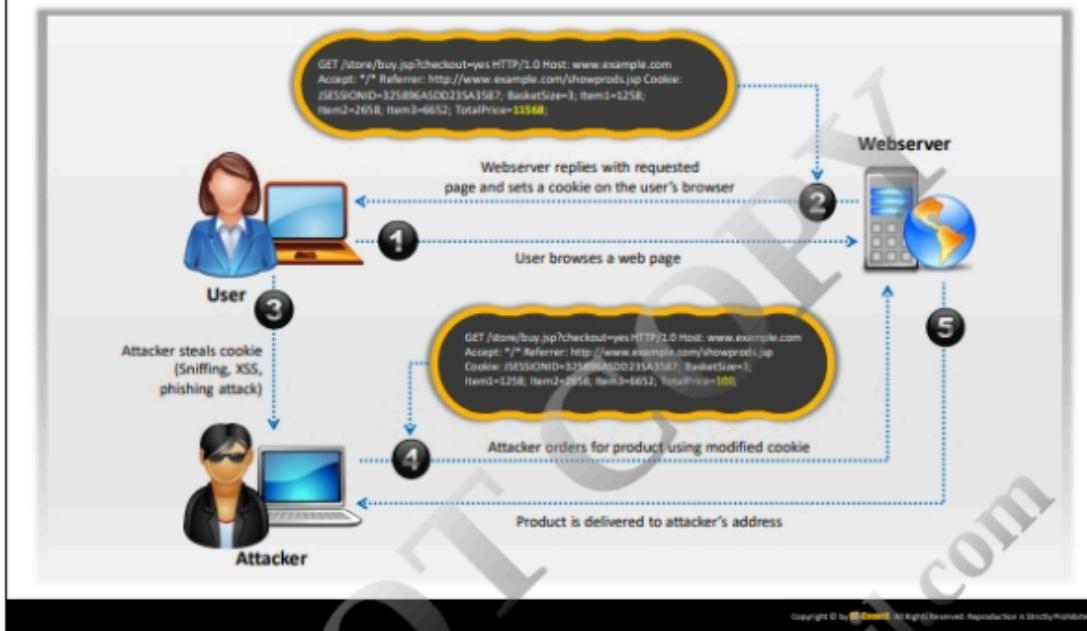
- Poisoning allows an attacker to inject malicious content, modify the user's online experience, and obtain **unauthorized information**

Rewriting the Session Data

- A proxy can be used for rewriting the session data, displaying the cookie data, and/or specifying a new **user ID or other session identifiers** in the cookie

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Session Attacks: Cookie Poisoning Attacks (Cont'd)



Session Attacks: Session Fixation



- In a session fixation attack, the attacker tricks the user to access a genuine webserver using an **explicit session ID** value
- Attacker assumes the identity of the victim and exploits his **credentials** at the server





Why Applications become Vulnerable to Attacks

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Common Reasons for Existence of Application Vulnerabilities



- Most of the software development related curriculum often **do not address security issues**
- **No proper guidance** provided to relevant stakeholders at different phases of the project development
- Failure to **gather** application security requirements in the **inception phase**
- Improper application of **security principles** in the **design phase**
- Insecure **coding techniques** give space to various vulnerabilities
- Lack of security **testing** in the **testing phase**
- Security **negligence** in the **deployment phase**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Common Flaws due to Insecure Coding Techniques



- | | |
|--|---|
| <input type="checkbox"/> Improper Input Validation | <input type="checkbox"/> Broken Authentication and Session Management |
| <input type="checkbox"/> Insecure Cryptographic Storage | <input type="checkbox"/> Improper Error Handling |
| <input type="checkbox"/> Insufficient Transport Layer Protection | <input type="checkbox"/> Unvalidated Redirects and Forwards |
| <input type="checkbox"/> Insecure Direct Object References | <input type="checkbox"/> Failure to Restrict URL Access |

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Improper Input Validation



Invalidated Input Data

- The application receives inputs from various sources such as human users, software agents (browsers), and network/peripheral devices that can be **suspicious or untrusted**.
- Processing inputs without proper validation can expose application to numerous threats

Malicious Script Execution

- Improper validation of input may provide path to the attackers to perform **injection attacks** such as cross site scripting attacks, SQL injection attacks, etc.
- The attacker can use various tricks and techniques to exploit the weaknesses in input validation mechanism of an application. They may **submit bogus data** to crash the system, maliciously manipulate the database, and corrupt the database of the application

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Insufficient Transport Layer Protection



■ Insufficient Transport Layer Protection

Exposes Data

This vulnerability exposes user's data to **untrusted third parties** and can lead to account theft

Launch Attacks

Underprivileged SSL setup can also help the attacker to launch phishing and **MITM attacks**

Supports Weak Algorithm

Insufficient transport layer protection supports weak algorithms, and uses **expired** or **invalid certificates**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

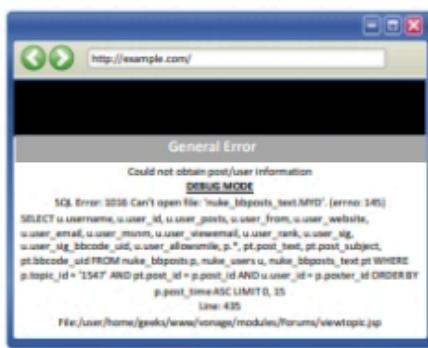
Improper Error Handling



- Improper error handling **gives insight into source code** such as logic flaws, default accounts, etc.
- Using the information received from an error message, an attacker **identifies vulnerabilities** for launching various web application attacks

Information Gathered

- Out of memory
- Null pointer exceptions
- System call failure
- Database unavailable
- Network timeout
- Database information
- Web application logical flow
- Application environment



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Insecure Cryptographic Storage



- Insecure cryptographic storage refers to an **application** when it **uses poorly written encryption code** to encrypt and store sensitive data in the database

- This flaw allows an attacker to **steal or modify weakly protected data** such as credit card numbers, SSNs, and other authentication credentials

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Broken Authentication and Session Management



- An attacker uses vulnerabilities in the **authentication** or **session management functions** such as exposed accounts, session IDs, logout, password management, timeouts, remember me, secret question, account update, and others to impersonate users

Session ID in URLs

`http://example.com/sale/saleitems=304;jsessionid=120MTOIDPXM00QSABGCKLHCJUN2JV?dest>NewMexico`

- Attacker sniffs the network traffic or tricks the user to get the session IDs, and reuses the session IDs for malicious purposes

Password Exploitation

- Attacker gains access to the web application's **password database**. If user passwords are not encrypted, attackers can exploit **passwords of all the users**

Timeout Exploitation

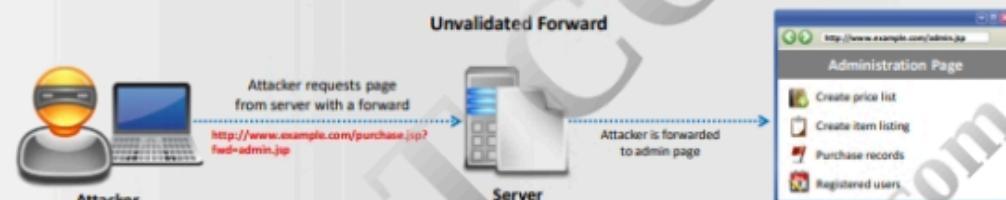
- If an application's timeouts are not set properly and a user simply closes the browser without logging out from sites accessed through a public computer then the attacker can use the same browser later and exploit the user's privileges

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Unvalidated Redirects and Forwards



- Unvalidated redirects enable attackers to **install malware or trick victims** into disclosing passwords or other sensitive information, whereas unsafe forwards may allow access control bypass



Insecure Direct Object References



Direct Object References

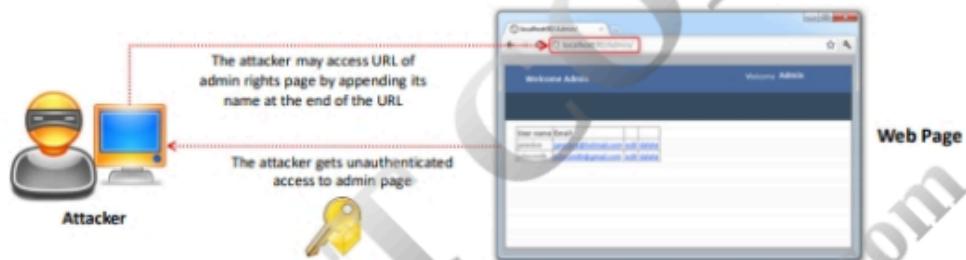
- Direct object reference vulnerability exists when a developer uses **internal object implementation** directly on the user interface
- These objects may include **file, directory, database record, or key**
- Attacker gets the idea about internal implementation and may use these objects as part of **URL** to access and modify sensitive information without any authorization
- In bank application, attacker may use **account number** directly as an input field in the user interface, as he/she knows that account number is generally the primary key of any bank application
- This may give path to the attacker to commit attacks on the **bank database**



Failure to Restrict URL Access



- There are certain pages in the application which handles sensitive operations and data. Access to these pages are restricted, given based on authorization. For example, certain functionality of the application needs only **administrator rights**
- During development, the **developer** tries to hide these pages from others by not providing direct links to them
- The attacker uses **forceful browsing techniques** to find the URL of such sensitive pages to gain **unauthorized access**



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



What Constitutes a Comprehensive Application Security?

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

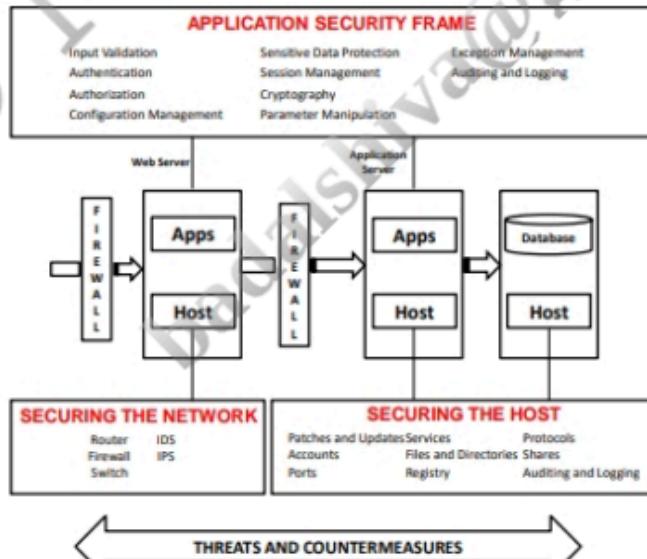


"Software application are designed and developed with functionality first in mind and security as a distant second or third."

-- OWASP

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Application Security Frame



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Coding securely alone does not ensure secure software development, a **missed requirement** or **bad web design** or **insecure architecture** can make application vulnerable to different types of attacks.

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

3W's in Application Security



Why

Why should we care about application security?

Due to its globally accessible nature, applications are becoming more popular targets for attackers to compromise an organization's security

What

What do we need for application security ?

A constant security vigilance at various phase of the application development lifecycle

Who

Who is responsible for application security?

Managers, Architects, Developers, Testers, and Administrators

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.



Insecure Application: A Software Development Problem

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Solution: Integrating Security in Software Development Life Cycle (SDLC)



Security Software Development Process

Requirement	Design	Development	Testing	Deployment	Maintenance
<ul style="list-style-type: none">• Security Requirements	<ul style="list-style-type: none">• Security Requirements• Secure Coding Standards• Threat Modeling• Security Architecture	<ul style="list-style-type: none">• Secure Coding Standards• Secure Design Patterns and Frameworks• Secure Coding Practices	<ul style="list-style-type: none">• Secure Code Review• Vulnerability Assessment• Penetration Testing	<ul style="list-style-type: none">• Secure Deployment	<ul style="list-style-type: none">• Security Patch Updates

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Functional vs Security Activities in SDLC

Software Development Lifecycle	Functional Activities	Security Activities
Requirement	<ul style="list-style-type: none">⊕ Functional requirements⊕ Non functional requirements⊕ Technology requirements	Defining security requirements
Design	Design guidelines and architectural design of project	<ul style="list-style-type: none">⊕ Secure design⊕ Secure coding standards⊕ Perform threat modeling⊕ Secure architecture
Development	Programming functional logic, unit testing	<ul style="list-style-type: none">⊕ Implementing security requirements⊕ Implementing secure coding standards⊕ Adopting secure coding practices
Testing	Functional testing such as black box testing, white box testing etc.	Security testing
Deployment	Deployment	Secure deployment
Maintenance	Update functionality	Updating with security patches

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Advantages of Integrating Security in SDLC



- Reduce the presence of **software vulnerabilities** to great extent
- Ability to **comply with regulations**, standards or requirements for secure software development
- Reduce **costly rework** by detecting and eliminating flaws at the earliest phase
- Improves developer **job satisfaction**
- Improves **customer satisfaction**
- Embeds **security culture** to improve quality and reliability
- Reuse of **trusted software** in future development
- Reduce the **maintenance cost**

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

The slide features a large watermark 'NOT COPY' diagonally across the center. At the top right is the CASE logo with 'JAVA' above it. The title 'Software Security Standards, Models, and Frameworks' is centered below the logo. A horizontal red bar is at the top and a blue bar is near the bottom. The footer contains a copyright notice: 'Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.'

The screenshot shows the OWASP homepage. On the left, there is a sidebar with various links like Home, About OWASP, Administrations, Advertising, and Events. The main content area has a large blue circular logo and the text 'Welcome to OWASP - the free and open software security community'. It also lists 'Dependency Check', 'ZAP', 'ASVS', 'SAMM', 'Top 10', 'OWASP', 'Development Guide', 'Test Gui', and 'Mac Rule'. Below this is a paragraph about the mission of OWASP. The footer includes a source link 'Source: https://www.owasp.org' and a copyright notice: 'Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.'

OWASP TOP 10 Attacks-2017

The diagram displays the OWASP Top 10 Attacks for 2017 in a 5x2 grid. The attacks are: A1 – Injection, A2 – Broken Authentication and Session Management, A3 – Cross-site Scripting (XSS), A4 – Broken Access Control, A5 – Security Misconfiguration, A6 – Sensitive Data Exposure, A7 – Insufficient Attack Protection, A8 – Cross-site Request Forgery (CSRF), A9 – Using Components with Known Vulnerabilities, and A10 – Underprotected APIs.

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

The Web Application Security Consortium (WASC)

The screenshot shows the homepage of the Web Application Security Consortium (WASC). The page features a navigation bar with links to Home, About Us, Projects, Library, News, Links, and Contact Us. Below the navigation is a main content area with sections for About Us, Charter Members, Mailing Lists, and Library. On the left, there's a sidebar with news items like "Hackers Put Social Networks Such as Twitter on Crosshairs" and "Real-World Software Security InformationWeek". The right sidebar contains a "Web Security List" with links to various security articles and notices. The footer includes a copyright notice: "Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited."

Source: <http://www.webappsec.org>

WASC Threat Classification



- Threat Classification is an effort to classify the **weaknesses**, and **attacks** that can lead to the compromise of a website, its data, or its users

Attacks	Weaknesses
Abuse of Functionality	Path Traversal
Brute Force	Predictable Resource Location
Buffer Overflow	Remote File Inclusion (RFI)
Content Spoofing	Routing Detour
Credential/Session Prediction	Session Fixation
Cross-Site Scripting	SOAP Array Abuse
Cross-Site Request Forgery	SSI Injection
Denial of Service	SQL Injection
Fingerprinting	URL Redirector Abuse
Format String	XPath Injection
HTTP Response Smuggling	XML Attribute Blowup
HTTP Response Splitting	XML External Entities
HTTP Request Smuggling	XML Entity Expansion
Integer Overflows	XML Injection
LDAP Injection	XQuery Injection
Mail Command Injection	
Null Byte Injection	
OS Commanding	

Source: <http://www.webappsec.org>

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Software Security Framework: Software Assurance Maturity Model (SAMM)



- The Software Assurance Maturity Model (SAMM) is an open framework to help organizations formulate and implement a strategy for **software security** that is tailored for the specific risks faced by the organization

- SAMM helps you:

- 1 Evaluate an organization's existing software security practices
- 2 Build a balanced software security assurance program in well-defined iterations
- 3 Demonstrate concrete improvements in the security assurance program
- 4 Define and measure security-related activities throughout an organization

Source: <http://www.opensamm.org>

Copyright © by EC-Council® All Rights Reserved. Reproduction is Strictly Prohibited.

Software Security Framework: Software Assurance Maturity Model (SAMM) (Cont'd)



- The **maturity model** consists of four business functions:

- Governance:** Assess the management of application security in an organization
- Construction:** Assess the software creation process in an organization
- Verification:** Assess the software testing of the application
- Deployment:** Assess the deployment (Software release management) and production of the application

- Each function posses three security practices



Source: <http://www.opensamm.org>

Copyright © by EC-Council®. All Rights Reserved. Reproduction is Strictly Prohibited.

Software Security Framework: Building Security In Maturity Model (BSIMM)



- The main objective of BSIMM is to enable the organization to **analyze** and **implement security features** required for the organization by evaluating most frequently implemented security features in other companies
- BSIMM is made up of a **software security framework** used to organize the **113 activities** used to assess initiatives
- The framework consists of **12 practices** organized into four domains

The Software Security Framework (SSF)

Governance	Intelligence	SSDL Touchpoints	Deployment
Strategy and Metrics	Attack Models	Architecture Analysis	Penetration Testing
Compliance and Policy	Security Features and Design	Code Review	Software Environment
Training	Standards and Requirements	Security Testing	Configuration Management and Vulnerability Management

Source: <https://www.bsimm.com>

Copyright © by EC-Council®. All Rights Reserved. Reproduction is Strictly Prohibited.

BSIMM vs OpenSAMM



BSIMM	OpenSAMM
Descriptive model	Prescriptive model
Has 12 security practices consisting of 112 activities	Has 12 security practices consisting of 72 activities
Based on things actually followed in an organization	Based on certain list of activities developed for software security
Has an active community that enables the organizations to understand the security features followed in other organizations	Do not have any active community activities

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

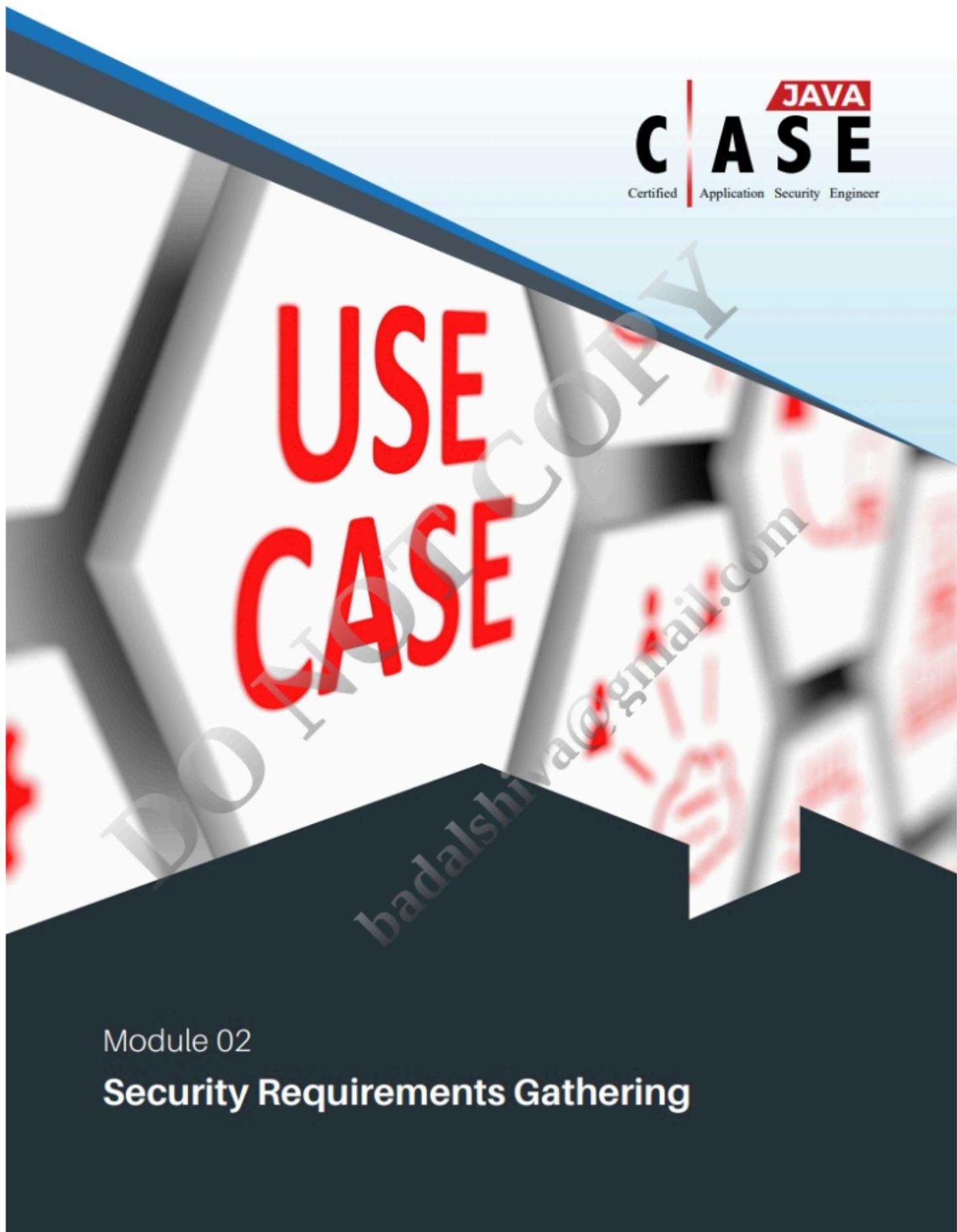
Module Summary



- An application is said to be secured when its restricted resources and secrets are protected from malicious use
- Attackers are exploiting various application vulnerabilities to steal secrets or gain unauthorized access to their protected resources
- The network security perimeter has huge holes at the application layer
- Applications are vulnerable to different types of attacks carried out at application level
- There are various reasons behind the existence of vulnerabilities in the applications
- Software applications are designed and developed with functionality first in mind and security as a distant second or third
- A constant security vigilance is required at various phases of the development lifecycle to develop secure applications
- Various Software Security Reference Standards, Models, Communities, and Frameworks can help in driving and maintaining application security

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

DO NOT COPY
This page is intentionally left blank.
badalshiva@gmail.com



Module 02

Security Requirements Gathering

DO NOT COPY
This page is intentionally left blank.
badalshiva@gmail.com

Module Objectives



- 1** Understand the Importance of Gathering Security Requirements
- 2** Understand Security Requirement Engineering (SRE) and its Phases
- 3** Describe Abuse Cases and Abuse Case Modeling
- 4** Describe Security Use Cases and Security Use Case Modeling
- 5** Describe Abuser and Security Stories
- 6** Explain Security Quality Requirements Engineering (SQUARE) Model
- 7** Explain Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) Model

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Importance of Gathering Security Requirements

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Security Requirements



- Software security requirements are **non functional** requirements, which need to be addressed to maintain the **confidentiality, integrity, and availability** of the application
- Stakeholders often overlook security requirement during the inception phase of software development
- This **negligence** may result in the application being vulnerable to different types of attacks or getting abused
- Gathering security requirements should be part of the **strategic application development process**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Gathering Security Requirements



- 1 Eliciting software security requirements takes different approach
- 2 It should be **enumerated** separate from the functional requirement so that they can be **reviewed** and **tested** separately
- 3 Mixing **security requirement** with **functional requirement** can make security requirement gathering process more **complicated** and **inaccurate**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Why We Need Different Approach for Security Requirements Gathering



- 1 Functional requirements are **positive requirements** specifying what the software should do
- 2 Security requirements are **negative requirements** specifying what the software should not do
- 3 It is **against** natural tendency of people that they are clear about what they want but quite find it difficult to understand what they don't want
- 4 A software needs to be viewed in a more **negative, critical, and destructive** way to reveal its non-intended use and its associated security requirements

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Key Benefits of Addressing Security at Requirement Phase



- Addressing security at requirement phase can save the economy **billions of dollars** as compared to addressing security at later phase of software development
- Security requirements give the developer an overview about **key security controls** required to build secure application
- It also specifies the **security mechanisms** that need to be implemented in order to comply with regulations, standards or requirements for the secure application development and attack protection
- Correctly understood security requirements can help in implementing security in **design, development, and testing** stages

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Stakeholders Involvement in Security Requirements Gathering



User, Customer or Business Sponsor



Developers



Requirement Engineer



Security Expert



Project Manager/Architects

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Characteristics of Good Security Requirement: SMART



Specific

Wording of the software requirement should be **clear** and **precise**. It should not be vague and all-encompassing

Measurable/Testable

There should be a **clear way** to test whether the specific requirement was met or not

Actionable

Developer should get clear understanding of what they exactly need to do to satisfy the requirements of the client

Realistic

It should be **implementable** in **real time** considering all the constraints

Timely

It should be of **high priority** when it is decided to implement

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Types of Security Requirements



- Security requirements are broadly classified into:

Functional Security Requirements

- It defines requirements specific to **business security**
- Example: Input validation, authentication, strong password, etc.

Security Drivers

- It defines requirements specific to **compliance with security policies, standards, etc.**
- Example: HIPPA, SOX, etc.

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Security Requirement Engineering (SRE)

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Security Requirement Engineering (SRE)



- Security Requirement Engineering (SRE) empowers **security requirement gathering** in the application development process
- SRE is the **systematic approach** to identify, analyze, elicit, and specify security requirements for software system
- SRE plays a major role in integrated security **software development lifecycle process**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

SRE Phases: Security Requirement Elicitation



- Requirement elicitation is the **initial activity** of any SRE approach

Typical actions in requirement elicitation:

- 1 **Analyze** the functional requirement of software system
- 2 Identify the relevant **stakeholders**
- 3 Gather security expectation required for various **functioning** of the system
- 4 Use different elicitation techniques such as brainstorming sessions, storyboarding, white boarding sessions, interviews, workshops, questionnaires, etc. to elicit security requirements

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

SRE Phases: Security Requirement Analysis



- Requirement analysis provides valid **blueprint** of security requirements elicited in earlier phase

Typical actions in requirement analysis:

- Consider various aspects of the security requirements gathered in the **requirements elicitation phase**
- Checks for any **conflicts** and **ambiguity** in the requirements
- Check for its **completeness** and **clarity**
- Check whether the requirement meets **stakeholder's expectation**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

SRE Phases: Security Requirement Specification



- The requirement specification phase plays an **integral** part in the requirement engineering process as the later stages in the development of the application completely depends on the specifications provided in the document

Typical actions in requirement specification:

- Record and organize the security requirements analyzed in requirement analysis phase in such a way that they can be clear, consistent, and traceable at any time

- Document all the security requirements for future reference

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



SRE Phases: Security Requirement Management

- The security requirement management phase deals with maintaining the security requirements **gathered** and **documented**

Typical **actions** in requirement management:



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Common Mistakes Made in Each Phase of SRE



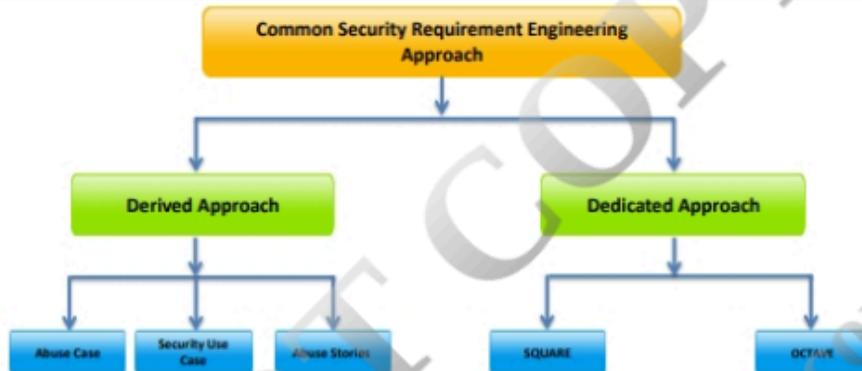
Requirements Elicitation	Requirement engineer will not be able to identified all security requirements due to lack of knowledge of modern elicitation techniques or does not include all the relevant stakeholder during requirement phase
Requirements Analysis	The identified requirements are directly specified without any analysis or modeling
Requirements Specification	The identified requirements is not SMART
Requirements Management	The identified requirements is not prioritized or scheduled properly

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Different Security Requirement Engineering Approaches/Model



- Typical SRE approaches are broadly classified into:
 - **Derived approach:** It is derived from traditional requirement gathering approach (use case modeling) to address security requirements in **software development process**
 - **Dedicated approach:** It is developed specifically to **address** security requirements



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



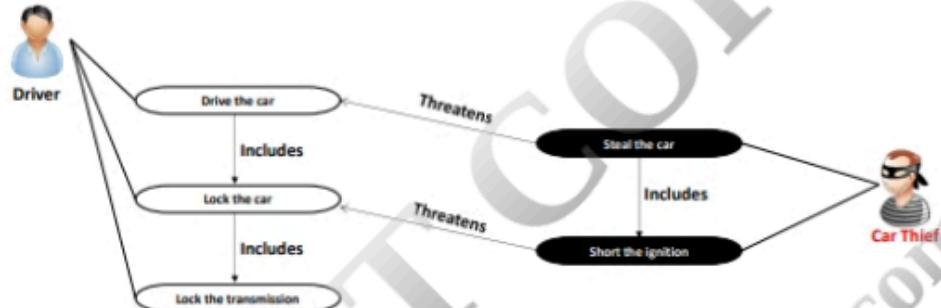
Abuse Case and Security Use Case Modeling

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Abuse Cases



- Abuse cases (or misuse case) derived from **use cases** of software system
- Abuse cases capture **abnormal behavior** of the system
- It depicts the actions taken by malicious users with the intent to **intrude** or **hack** or **compromise** the software functioning
- It helps **stakeholders** to differentiate between the appropriate and inappropriate use of software system
- Abuse cases lay the foundation for **threat modeling** in design phase



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Threatens Relationship



- The “threaten” relationship is used to describe **abuse case** scenario
- It describes the way the **attacker** can abuse the system
- It helps to **portray** the activity where an abuse case can threaten a use case with failure



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Abuse Case Modeling Steps



Step 1: Conduct **brainstorming** sessions or whiteboard sessions with relevant stakeholders and security experts

- Abuse case should be considered while developing use cases for functional requirements

Step 2: Analyze the use cases generated to capture **functional requirements**

- Start with High level use cases
- Continue to analyze use cases at granular level
- Understand the complete picture about what system does

Step 3: Think from **adversary's** perspective

- How system can be misused?
- How system can be disrupted?

Step 4: Identify the **malicious** actor

Step 5: Identify, specify, and define the **abuse** cases

Step 6: Build abuse case for every **use case**

Step 7: Check for **granularity** of detail

Step 8: Check **completeness** and **minimality**

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Abuse Cases: Advantages and Disadvantages



Advantages

- It is easy for the managers to **learn** and **use** abuse cases
- Abuse cases contribute a lot during **requirement gathering phase**

Disadvantages

- Abuse cases can only be used with **object oriented software systems**
- Abuse case models do not specify any detail regarding the relation between use case and abuse case

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Abuse Case Template



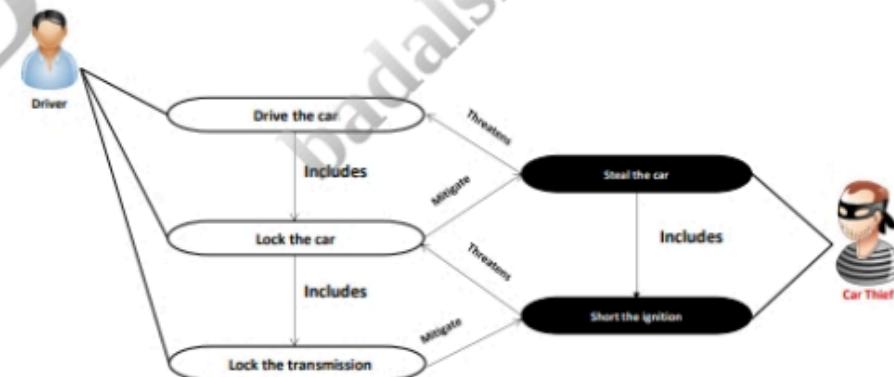
Abuse Case ID:	
Name:	
Priority:	Low or Medium or High
Scope:	
Development Environment:	
Mis-actors:	
Access Right Levels:	Low Level System Users or/and Medium Level System Users or/and High Level System Users or/and System Admin or/and Other Network User
Point of Entry:	
Security Attributes Affected:	Confidentiality or/and Integrity or/and Availability
Description:	
Sophistication:	Low or Medium or High
Pre-conditions:	
Assumptions:	
Post-conditions:	
Potential Mis-actor Profiles:	
Stakeholders and Threats:	
Related Use Cases:	
Related Threats:	
Architectural Recommendation:	

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Security Use Cases



- Security use cases are derived from **abuse cases**
- Security use cases **capture** the security requirements needed to defeat each abuse case
- It **uncovers** software abuse cases and its response



Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.