EC-Council

JAVA
C|ASE
Certified Application Security Engineer

Certified Application Security Engineer

COURSEWARE

EC-COUNCIL OFFICIAL CURRICULA

This page is intentionally left blank.

Certified Application Security Engineer

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

EC-Council

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

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.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

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

C|SCU
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 Certied 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 rst 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 desgined 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 di-erent 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 thought the Five Phases of Ethical Hacking and thought how you can approach your target and succeed at breaking in every time! The ve phases include Reconnaissance, Gaining Access, Enumeration, Maintaining Access, and covering your tracks.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Penetration Testing: 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.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

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.

Table of Contents

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

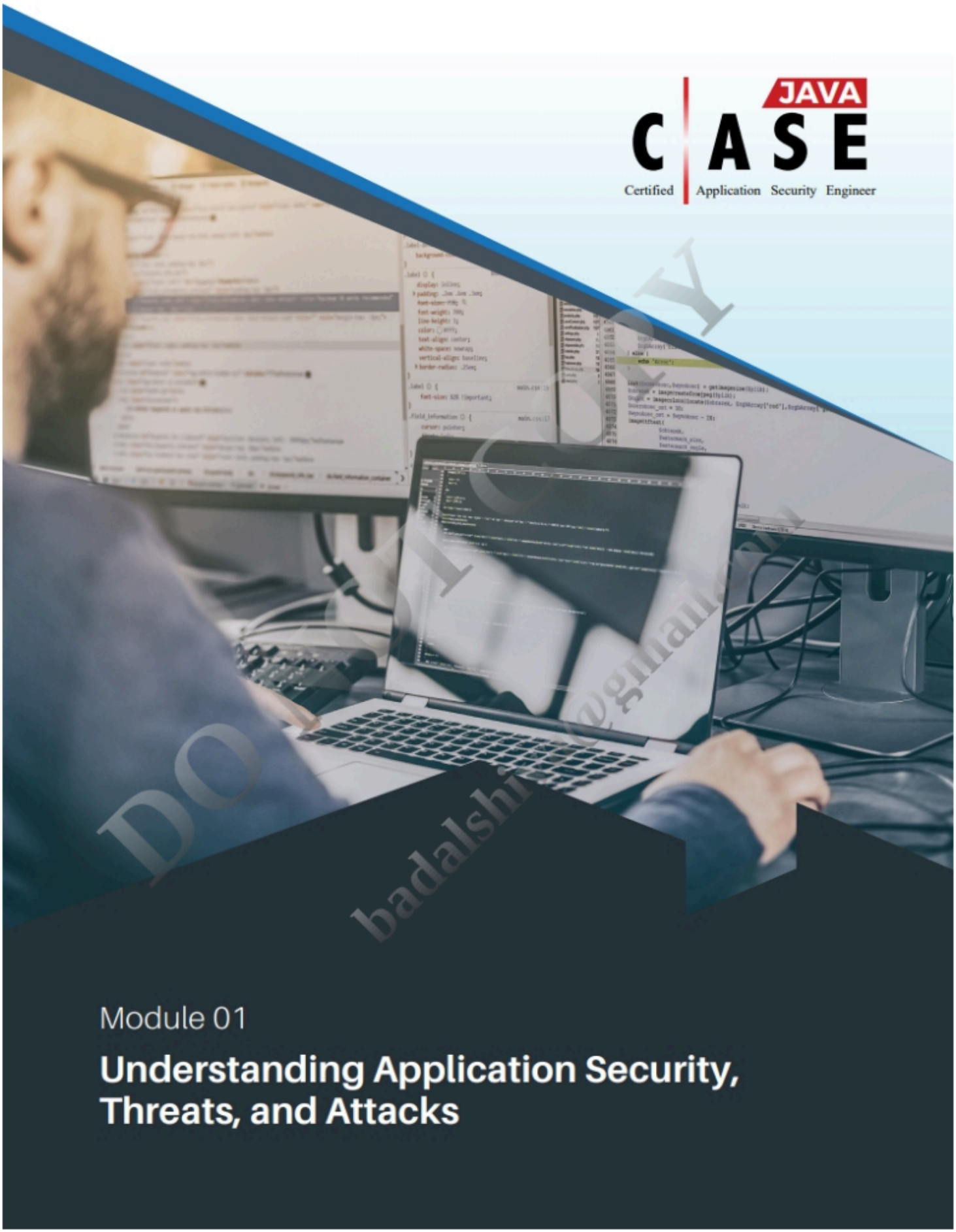
Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.

Module 09: Static and Dynamic Application Security Testing (SAST & DAST) 333

Printed by: badalshiva@gmail.com. Printing is for personal, private use only. No part of this book may be reproduced or transmitted without publisher's prior permission. Violators will be prosecuted.



This page is intentionally left blank.

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

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

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:

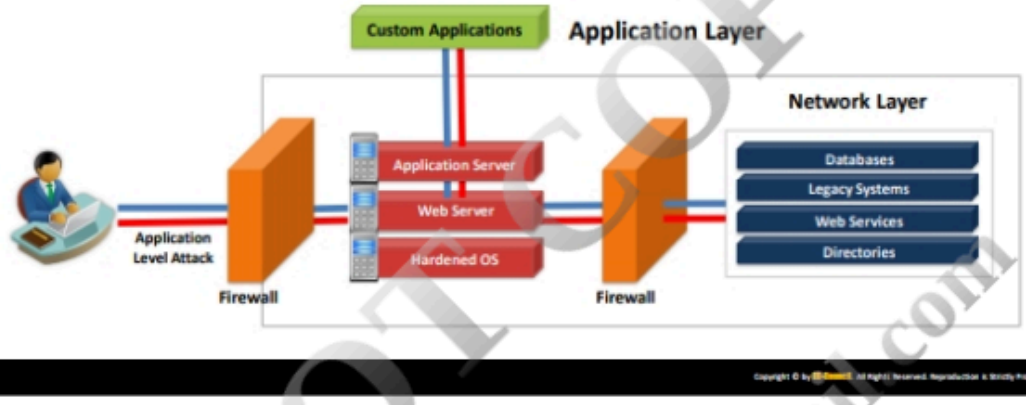
1. Financial Loss
2. Affects Business Continuity
3. Closure of Business
4. Disclosure of Business Information
5. Damages Reputation
6. Fraudulent Transactions

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

--Carlos Lyons, Corporate Security, Microsoft

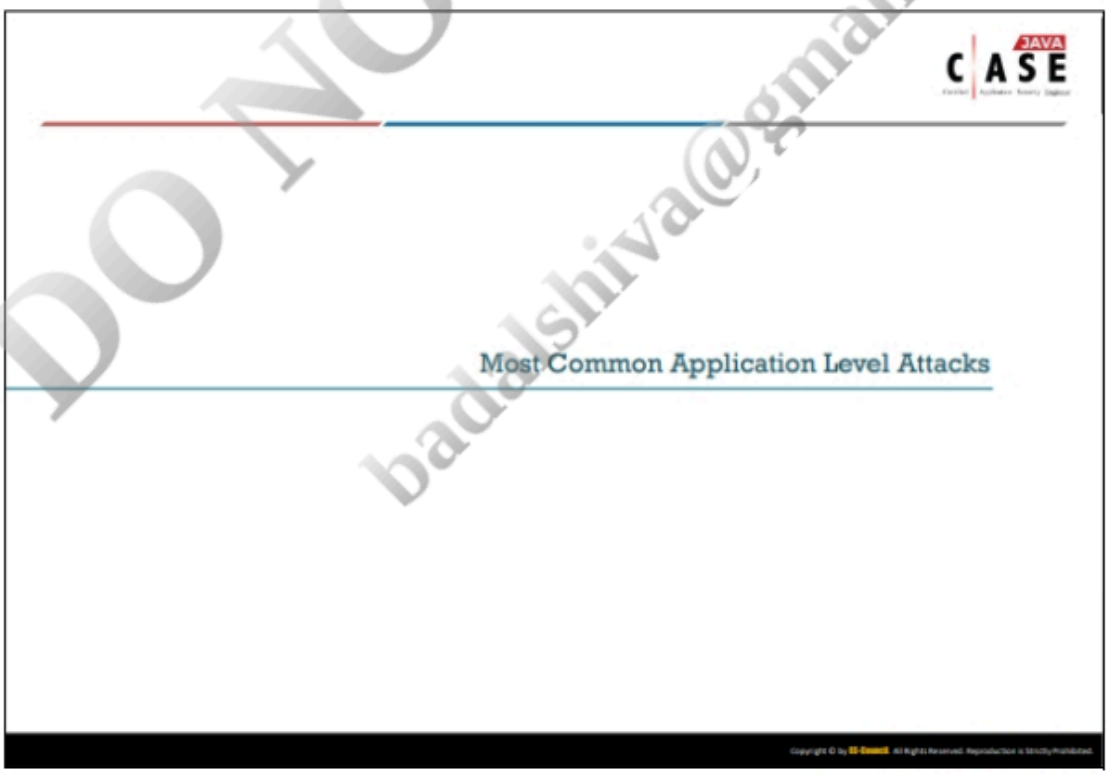
Need for Application Security (Cont'd)

- 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



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

-- Gartner Report



SQL Injection Attacks

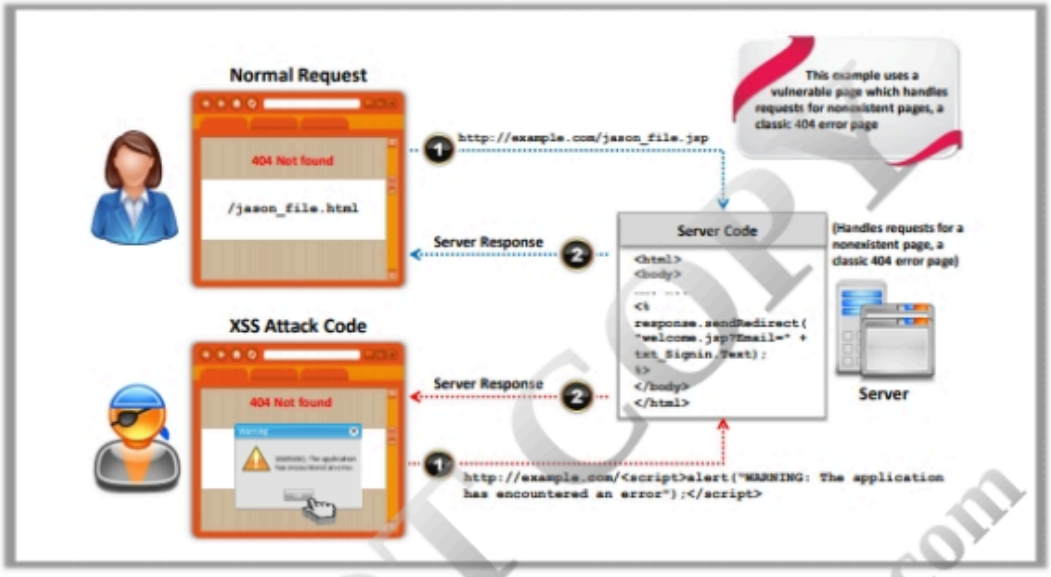
- SQL injection attacks use a **series of malicious SQL queries** to directly manipulate the database
- An attacker can use a vulnerable web application to **bypass normal security measures** and obtain direct access to valuable data
- SQL injection attacks can often be executed from the **address bar**, 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**

Malicious script execution	Session hijacking
Redirecting to a malicious server	Brute force password cracking
Exploiting user privileges	Data theft
Ads in hidden IFRAMES and pop-ups	Intranet probing
Data manipulation	Keylogging and remote monitoring

Cross-site Scripting (XSS) Attacks (Cont'd)

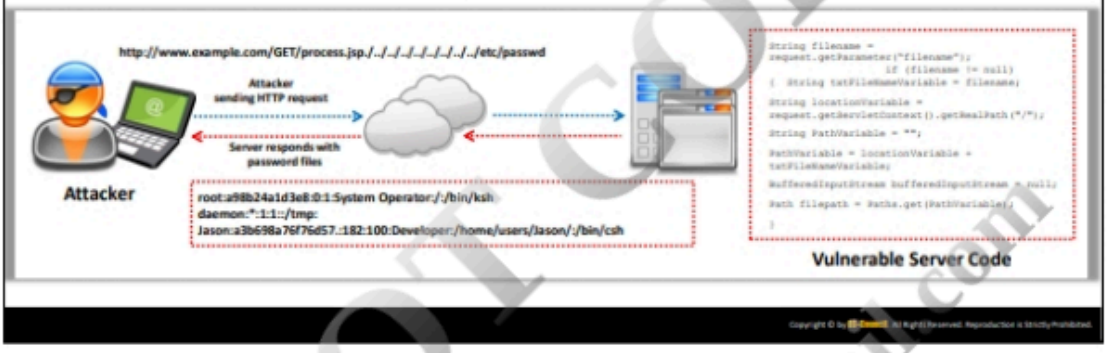


Parameter Tampering

- A web parameter tampering attack involves **manipulation of parameters exchanged** between client and server in order to modify application data such as user credentials and permissions, price, and quantity of products
- A parameter tampering attack **exploits vulnerabilities** in integrity and logic validation mechanisms that may result in XSS, SQL injection, etc.

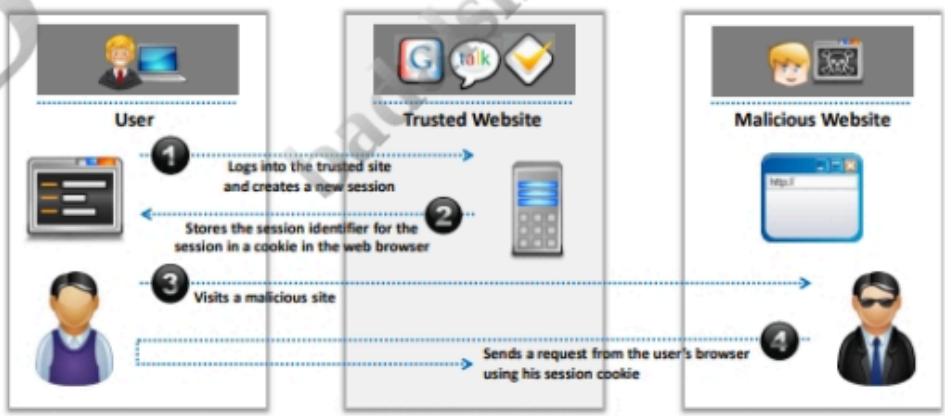
Directory Traversal

- 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=../../../../some dir/some file`
  - `http://www.example.com/../../../../some dir/some file`

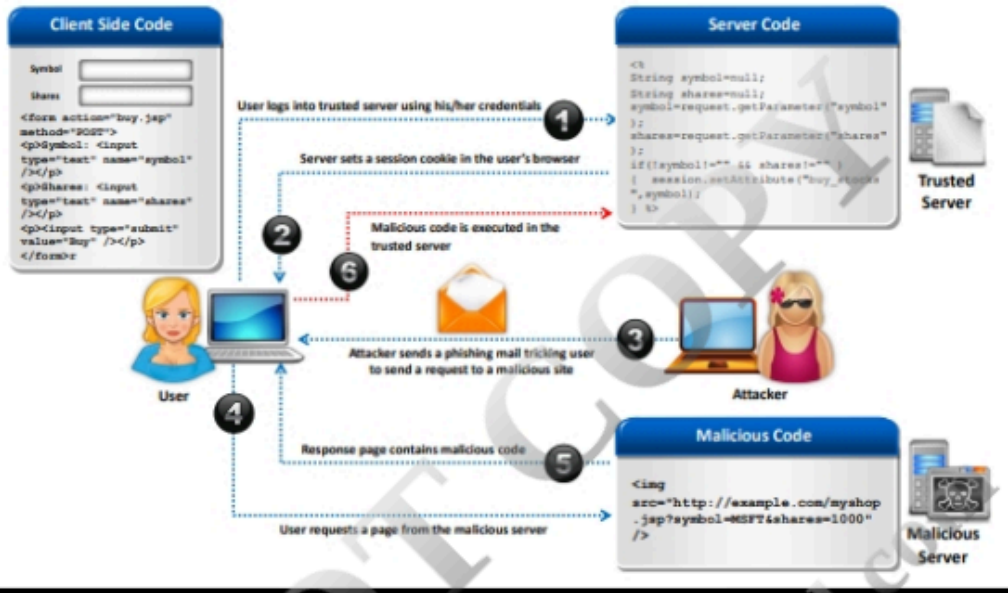


Cross-site Request Forgery (CSRF) Attack

- 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



Cross-site Request Forgery (CSRF) Attack (Cont'd)



Denial-of-Service (DoS) Attack

- Attackers exhaust available server resources by sending hundreds of **resource-intensive requests** such as pulling out large image files or requesting dynamic pages that require expensive search operations on the backend of database servers
- Application-level DoS attacks emulate the same request syntax and network-level traffic characteristics as that of the legitimate clients, which makes it **undetectable by existing DoS protection** measures

Targets

- CPU, Memory, and Sockets
- Disk Bandwidth
- Database Bandwidth
- Worker Processes

Why are Applications Vulnerable to DoS?

- Reasonable Use of Expectations
- Application Environment Bottlenecks
- Implementation Flaws
- Poor Data Validation

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

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

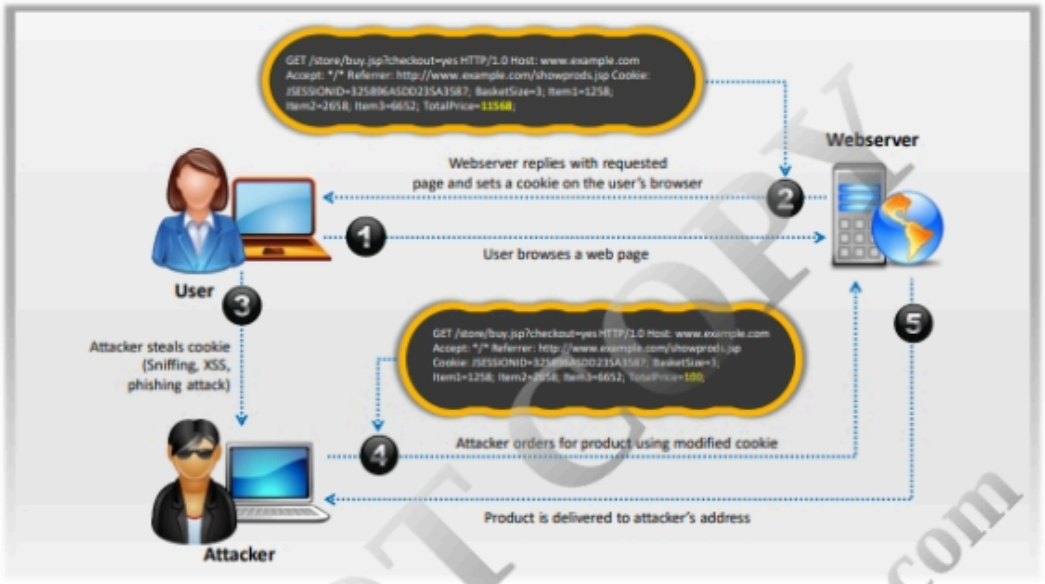
Inject the Malicious Content

- 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

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

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**

Common Flaws due to Insecure Coding Techniques

- Improper Input Validation
- Broken Authentication and Session Management
- Insecure Cryptographic Storage
- Improper Error Handling
- Insufficient Transport Layer Protection
- Unvalidated Redirects and Forwards
- Insecure Direct Object References
- Failure to Restrict URL Access

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

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**

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

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

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=12OMTOIDPXM0OQSABGCKLHCJUN2JV?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

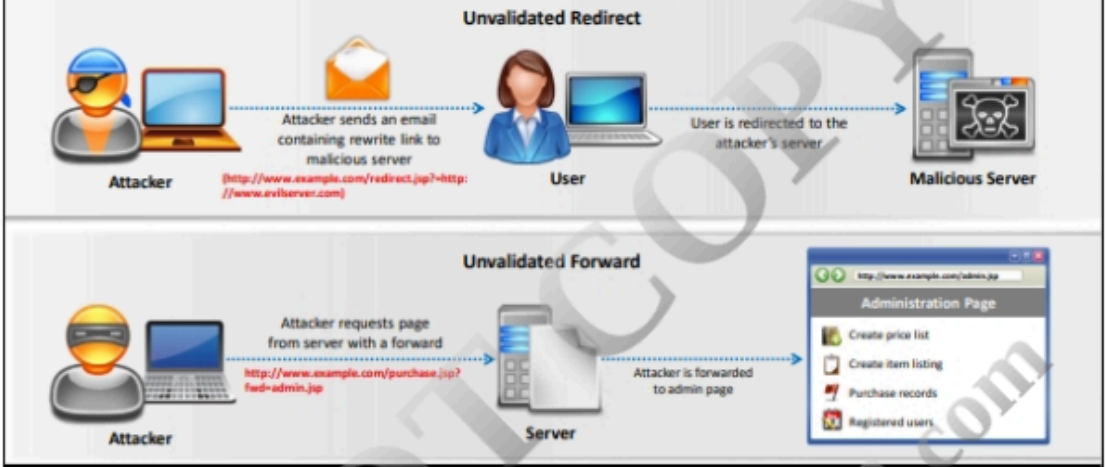
- Attacker gains access to the web application's **password database**. If user passwords are not encrypted, attackers can exploit **passwords of all the** users

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**

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

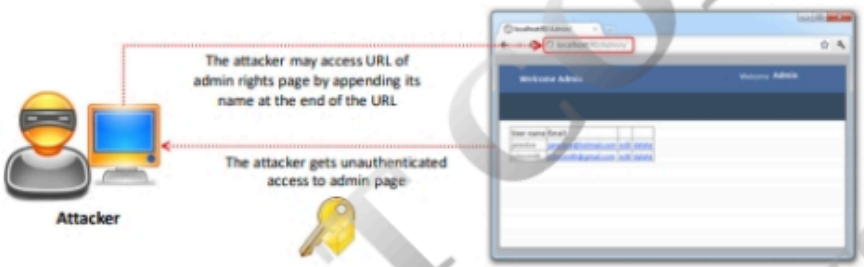


Insecure Direct Object References

- **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**

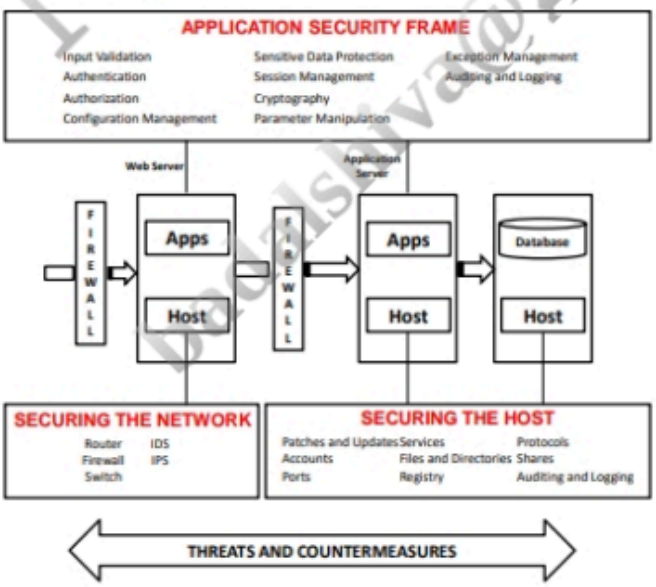


What Constitutes a Comprehensive Application Security?

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

-- OWASP

Application Security Frame



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.

3W's in Application Security

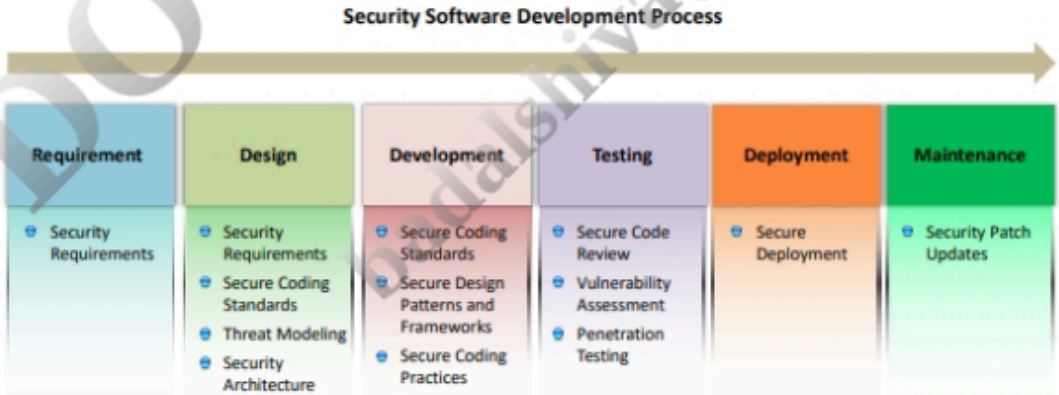
Why	Why should we care about application security?
Due to its globally accessible nature, applications are becoming more popular targets for attackers to compromise an organization's security	

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	

Insecure Application: A Software Development Problem

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



Functional vs Security Activities in SDLC

Software Development Lifecycle	Functional Activities	Security Activities
Requirement	• Functional requirements • Non functional requirements • Technology requirements	Defining security requirements
Design	Design guidelines and architectural design of project	• Secure design • Secure coding standards • Perform threat modeling • Secure architecture
Development	Programming functional logic, unit testing	• 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

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**

Software Security Standards, Models, and Frameworks

The Open Web Application Security Project (OWASP)

- The Open Web Application Security Project (OWASP) is an organization focused on **improving** the security of software
- The mission of this organization is to make **software security visible**, so that individuals and organizations are able to make informed decisions

Source: https://www.owasp.org

OWASP TOP 10 Attacks-2017

- 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
- A10 – Underprotected APIs

The Web Application Security Consortium (WASC)

- The Web Application Security Consortium (WASC) is an international group of **experts**, industry **practitioners**, and organizational **representatives** who produce open source and widely agreed upon best-practice security standards for the World Wide Web

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	Application Misconfiguration
Brute Force	Predictable Resource Location	Directory Indexing
Buffer Overflow	Remote File Inclusion (RFI)	Improper File system Permissions
Content Spoofing	Routing Detour	Improper Input Handling
Credential/Session Prediction	Session Fixation	Improper Output Handling
Cross-Site Scripting	SOAP Array Abuse	Information Leakage
Cross-Site Request Forgery	SSI Injection	Insecure Indexing
Denial of Service	SQL Injection	Insufficient Anti-automation
Fingerprinting	URL Redirector Abuse	Insufficient Authentication
Format String	XPath Injection	Insufficient Authorization
HTTP Response Smuggling	XML Attribute Blowup	Insufficient Password Recovery
HTTP Response Splitting	XML External Entities	Insufficient Process Validation
HTTP Request Smuggling	XML Entity Expansion	Insufficient Session Expiration
Integer Overflows	XML Injection	Insufficient Transport Layer Protection
LDAP Injection	XQuery Injection	Server Misconfiguration
Mail Command Injection		
Null Byte Injection		
OS Commanding		

Source: http://www.webappsec.org

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

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

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

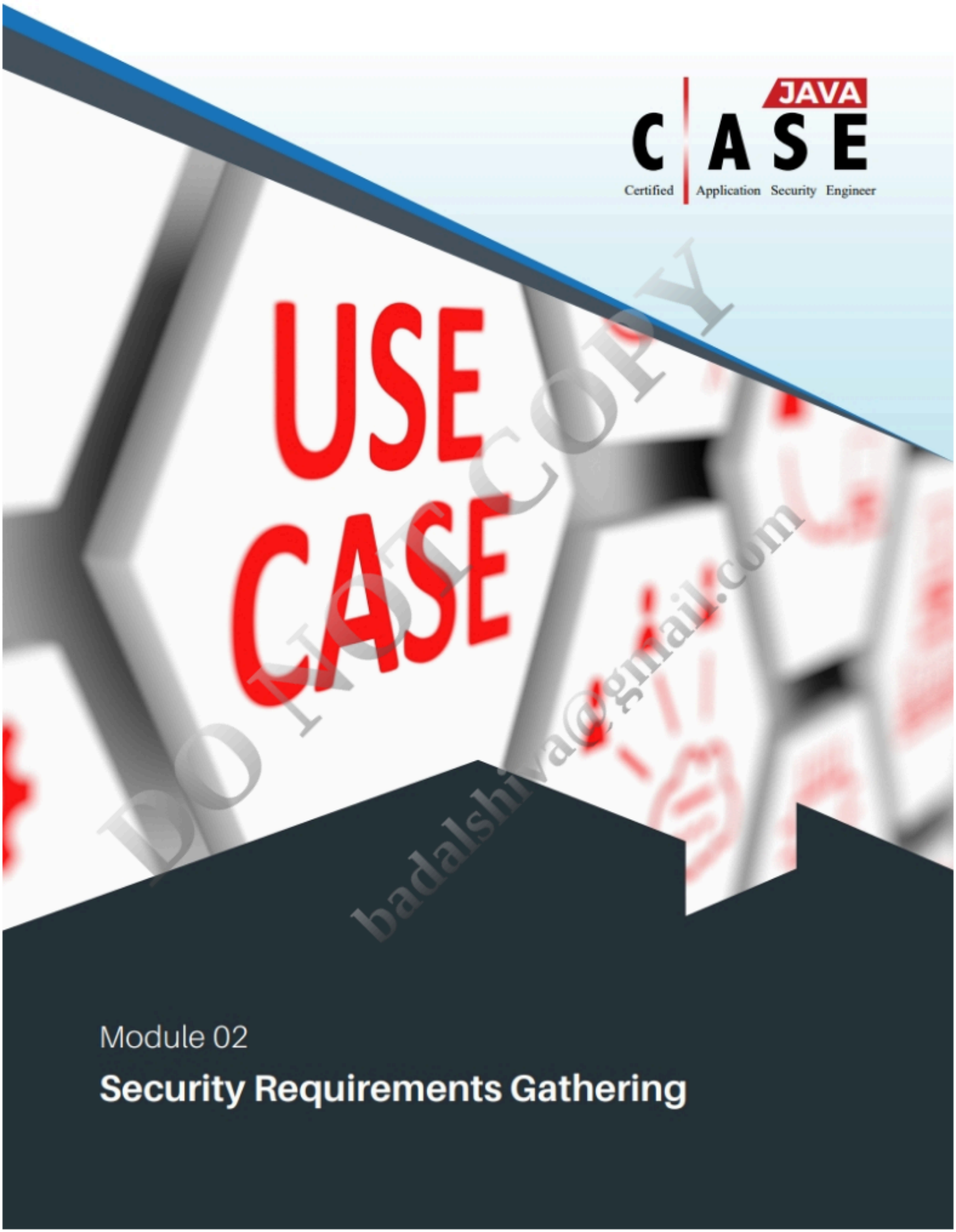
BSIMM vs OpenSAMM

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

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
- Application 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 application 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 application
- Various Software Security Reference Standards, Models, Communities, and Frameworks can help in driving and maintaining application security

This page is intentionally left blank.



This page is intentionally left blank.

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

Importance of Gathering Security Requirements

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**

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**

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

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

Stakeholders Involvement in Security Requirements Gathering

- User, Customer or Business Sponsor
- Developers
- Requirement Engineer
- Security Expert
- Project Manager/Architects

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

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.

Security Requirement Engineering (SRE)

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**

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

SRE Phases: Security Requirement Analysis

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

Typical actions in requirement analysis:

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

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:

1. **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
2. **Document** all the security requirements for future reference

SRE Phases: Security Requirement Management

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

Typical actions in requirement management:

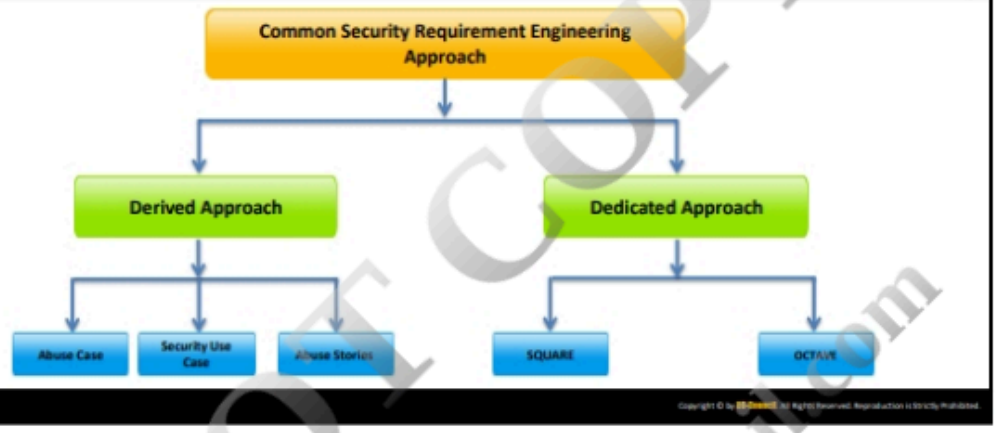
1. **Update** of the security requirements
2. Provide support based on the selected **SRE approach**

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

Different Security Requirement Engineering Approaches/Model

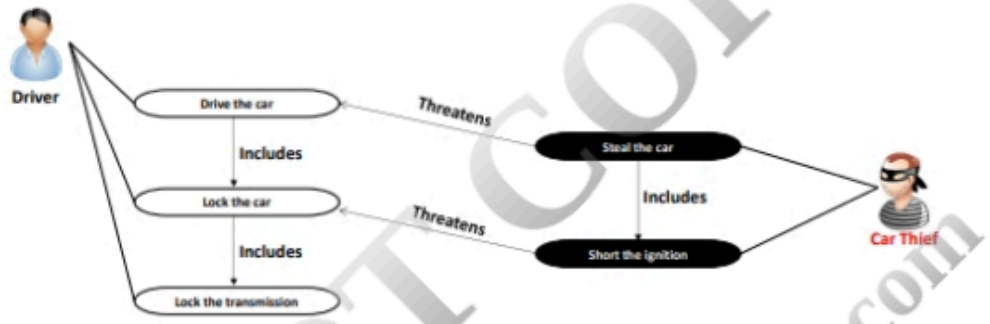
- Typical SRE approaches are broadly classified into:
  - **Derived approach**: It is derived from traditional requirement gathering approach (use case modeling) to address security requirements in software development process
  - **Dedicated approach**: It is developed specifically to address security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



Abuse Case and Security Use Case Modeling

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



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

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**

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

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:	

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