Asia Masters Center

Certified Application Security Engineer (CASE) Net

Asia Masters Centre (AMC), Suite 2 B, level 6, Office Block, Grand Millennium Hotel, Bukit Bintang Street, 55100 Kuala Lumpur, Malaysia. | Tel: +60327326992|Mobile: +601 8909 0379 | Fax: +60327326992
Website: http://www.asiamasters.org/ | Email: info@asia-masters.com

Asia Masters Center

Certified Application Security Engineer (CASE) Net

Course Objective

- ➢ To ensure that application security is no longer an afterthought but a foremost one.
- ➢ To lay the foundation required by all application developers and development organizations, to produce secure applications with greater stability and fewer security risks to the consumer, therefore, making security a foremost thought.
- ➢ To ensure that the organizations mitigate the risk of losing millions due to security compromises that may arise with every step of application development process.
- ➢ To help individuals develop the habit of giving importance to security sacrosanct of their job role in the SDLC, therefore opening security as the main domain for testers, developers, network administrator.

Target Audience

- ➢ NET Developers with a minimum of 2 years of experience and individuals who want to become application security engineers/analysts/testers
- ➢ Individuals involved in the role of developing, testing, managing, or protecting wide area of applications

Course Outline

- Module 1: Understanding Application Security, Threats, and Attacks.
- What is a Secure Application
- Need for Application Security
- Most Common Application Level Attacks
  - SQL Injection Attacks
  - Cross-site Scripting (XSS) Attacks
  - Parameter Tampering
  - Directory Traversal
  - Cross-site Request Forgery (CSRF) Attack
  - Denial-of-Service (DoS) Attack
- Denial-of-Service (DoS): Examples
  - Session Attacks
- Cookie Poisoning Attacks
- Session Fixation
- Why Applications become Vulnerable to Attacks
  - Common Reasons for Existence of Application Vulnerabilities
  - Common Flaws Existed due to Insecure Coding Techniques
  - Improper Input Validation
  - Insufficient Transport Layer Protection
  - Improper Error Handling
  - Insecure Cryptographic Storage
  - Broken Authentication and Session Management
  - Unvalidated Redirects and Forwards
  - Insecure Direct Object References
  - Failure to Restrict URL Access
- What Constitutes a Comprehensive Application Security?
  - Application Security Frame

  - 3W's in Application Security
- Insecure Application: A Software Development Problem
  - Solution: Integrating Security in Software Development Life Cycle (SDLC)
  - Functional vs Security Activities in SDLC
  - Advantages of Integrating Security in SDLC
  - Microsoft Security Development Lifecycle (SDL)
- Software Security Standards, Models, and Frameworks
  - The Open Web Application Security Project (OWASP)
  - OWASP TOP 10 Attacks-2017
  - The Web Application Security Consortium (WASC)
  - WASC Threat Classification
  - Software Security Framework
- Software Assurance Maturity Model (SAMM)
- Building Security in Maturity Model (BSIMM)
  - BSIMM vs OpenSAMM

➢ Module 2: Security Requirements Gathering.

- Importance of Gathering Security Requirements
- Security Requirements
- Gathering Security Requirements
- Why We Need Different Approach for Security Requirements Gathering
- Key Benefits of Addressing Security at Requirement Phase
- Stakeholders Involvement in Security Requirements Gathering
- Characteristics of Good Security Requirement: SMART
- Types of Security Requirements
- Functional Security Requirements
- Security Drivers
- Security Requirement Engineering (SRE)
- SRE Phases

- Security Requirement Elicitation
- Security Requirement Analysis
- Security Requirement Specification
- Security Requirement Management
- Common Mistakes Made in Each Phase of SRE
- Different Security Requirement Engineering Approaches/Model
- Abuse Case and Security Use Case Modeling
- Abuse Cases
- Threatens Relationship
- Abuse Case Modeling Steps
- Abuse Cases: Advantages and Disadvantages
- Abuse Case Template
- Security Use Cases
- Security Use Cases are Abuse Case Driven
- Modeling Steps for Security Use Cases
- Mitigates Relationship
- Abuse Case vs Security Use Case
- Security Use Case: Advantages and Disadvantages
- Security Use Case Template
- Security Use Case Guidelines
- Example 1: Use Case for Online Bidding System
- Example 1: Abuse Case for Online Bidding System
- Example 1: Security Use Case for Online Bidding System
- Example 2: Use Case for ATM System
- Example 2: Abuse Case for ATM System
- Example 2: Security Use Case for ATM System
- Example 3: Use Case for E-commerce System
- Example 3: Abuse Case for E-commerce System
- Example 3: Security Use Case for E-commerce System

- Effectiveness of Abuse and Security Case
- Abuser and Security Stories
- Textual Description Template: Abuser Stories and Security Stories
- Examples: Abuser Stories and Security Stories
- Effectiveness of Abuser and Security Stories
- Abuser Stories: Advantages and Disadvantages
- Security Quality Requirements Engineering (SQUARE)
- SQUARE Effectiveness
- SQUARE Process
- SQUARE: Advantages and Disadvantages
- Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE)
- OCTAVE Effectiveness
- OCTAVE Steps
- OCTAVE: Advantages and Disadvantages

➢ Module 3: Secure Application Design and Architecture.

- Relative Cost of Fixing Vulnerabilities at Different Phases of SDLC
- Secure Application Design and Architecture
- Goal of Secure Design Process
- Secure Design Actions
- Security Requirement Specifications
- Secure Design Principles
- Threat Modeling
- Secure Application Architecture
- Secure Design Principles
- Define Secure Design principles
- Secure Design Principles

- Security through obscurity
- Secure the Weakest Link
- Use Least Privilege Principle
- Secure by Default
- Fail Securely
- Apply Defense in Depth
- Do Not Trust User Input
- Reduce Attack Surface
- Enable Auditing and Logging
- Keep Security Simple
- Separation of Duties
- Fix Security Issues Correctly
- Apply Security in Design Phase
- Protect Sensitive Data
- Exception Handling
- Secure Memory Management
- Protect Memory or Storage Secrets
- Fundamentals of Control Granularity
- Fault Tolerance
- Fault Detection
- Fault Removal
- Fault Avoidance
- Loose Coupling
- High Cohesion
- Change Management and Version Control
- Threat Modeling
- Threat Modeling Phases
- Attack Surface Evaluation
- Threat Identification

- Impact Analysis
- Control Recommendations
- Threat Modeling Process
- Identify Security Objective
- Application Overview
- Decompose Application
- Identify Threats
- Identify Vulnerabilities
- Identify Security Objective
- How to Identify Security Objectives
- Create an Application Overview
- Draw the End-to-End Deployment Architecture
- Identify Various User Roles
- Identify Use Cases Scenarios
- Identify Technologies
- Identify Application Security Mechanisms
- Decompose Application
- Prepare and Document Threat Model Information
- Example: Threat Model Information
- Identify the External Dependencies
- External Dependencies: Example
- Identify the Entry Points
- Entry Points: Example
- Identify the Assets
- Assets: Example
- Identify the Trust Levels
- Trust Levels: Example
- Define Trust Levels to Entry points
- Define Trust Levels to Assets

- Perform Application Modelling using Data Flow Diagrams (DFDs)
- Determine the Threats: Identify the Goal of an Attacker and Create Threat Profile
- Example: Attacker’s Goal/Threat Profile and Vulnerabilities Associated
- Determine the Threats: Create a Security Profile
- Identify the Threats
- The STRIDE Model
- Example: Threat Categorized and Identified using STRIDE
- Determine Countermeasures and Mitigation Security Controls
- Document the Threats
- Rating the Threats
- Rating the Threats: DREAD Model
- Secure Application Architecture
- Design Secure Application Architecture

➢ Module 4: Secure Coding Practices for Input Validation.

- Input Validation
- Why Input Validation?
- Input Validation Specification
- Input Validation Approaches
- Client-side Input Validation
- Server-side Input Validation
- Client-Server Input Validation Reliability
- Input Filtering
- Input Filtering Technique o Black Listing o White Listing
- Input Filtering using a Regular Expression
- Secure Coding Practices for Input Validation: Web Forms
- ASP.NET Validation Controls

- Set of ASP.NET Validation Controls
- Required Field Validation Control
- Range Validation Control
- Comparison Validation Control
- Regular Expression Validation Control
- Custom Validation Control
- Validation Summary Control
- SQL Injection Attack Defensive Techniques
- Using Parameterized Queries
- Using Parameterized Stored Procedures
- Using Escape Routines to Handle Special Input Characters
- Using a Least-privileged Database Account
- Constraining Input
- XSS Attack Defensive Techniques
- Output Encoding
- Encoding Unsafe Output using HtmlEncode
- Encoding Unsafe Output using UrlEncode
- Anti-XSS Library
- Encoding Output using Anti-XSS Library
- Directory Traversing Defensive Technique
- Additional Techniques to Prevent Directory Traversal
- Secure Coding Practices for Input Validation: ASP.NET Core
- Input Validation using ModelState Object
- Input Validation using Data Annotation
- Input Validation using Custom Validation Attributes
- Input Validation using Remote Validation
- SQL Injection Attack Defensive Techniques
- Sanitize Inputs using Casting
- Using Parameterized Queries

- Using Stored Procedures
- Using ORM (Object Relation Model)
- XSS Defensive Techniques
- Enable Content Security Policy
- URL Encoding User Input
- Open Redirect Defensive Techniques
- Implement LocalRedirect()
- Disable X-Frame-Options
- Enable Cross Origin Request Sharing
- Enable Cross Origin Request Sharing (CORS) with Middleware
- Guidelines for Secure (CORS) Configuration
- Directory Traversing Defensive Techniques
- Disable Directory Listing
- Disable Non-standard Content Types
- Secure Static Files
- Secure Coding Practices for Input Validation: MVC
- XSS Defensive Techniques
- Enable Content Security Policy
- MVC Output Encoding
- Output Encoding using Anti-XSS Library
- Parameter Tampering Defensive Techniques
- Accept Data from Trusted Sources
- Encrypt and Decrypt Key Values
- Implement LocalRedirect()
- Open Redirect Defensive Techniques

➢ Module 5: Secure Coding Practices for Authentication and Authorization.

- Authentication and Authorization
- Authentication

Asia Masters Center

- Authorization
- Common Threats on User Authentication and Authorization
- Account Hijacking
- Man-in-the-middle
- Phishing
- Unauthorized Access
- Information Leakage
- Privilege Escalation
- Sniffing
- Authentication and Authorization: Web Forms
- .NET Authentication and Authorization
- Different Level of Authentication
- ASP.NET Authentication
- Enterprise Services Authentication
- SQL Server Authentication
- ASP.NET Authentication
- ASP.NET Authentication Modes
- Forms Authentication
- Passport Authentication
- Custom Authentication
- Implementing Custom Authentication Scheme
- Windows Authentication
- Basic Authentication
- Digest Authentication
- Integrated Windows Authentication
- Certificate Authentication
- Anonymous Authentication
- Selecting an Appropriate Authentication Method
- Determining an Authentication Method

- Enterprise Services Authentication
- SQL Server Authentication
- Mixed Mode Authentication
- Windows Authentication
- Different Level of Authorization
- ASP.NET Authorization
- Enterprise Services Authorization
- SQL Server Authorization
- ASP.NET Authorization
- URL Authorization
- File Authorization
- What is Impersonation?
- Impersonation Options
- Impersonation is Disabled
- Impersonation Enabled
- Impersonation Enabled for a specific Identity
- Delegation
- Code-based Authorization
- Explicit Authorization
- Declarative Authorization
- Imperative Authorization
- Authorization using ASP.NET Roles
- Enterprise Services Authorization
- SQL Server Authorization
- User-defined Database Roles
- Application Roles
- Fixed Database Roles
- Authentication and Authorization: ASP.NET Core
- ASP.NET Core Authentication

- AspNetCore.Identity
- ASP.NET Core Authentication
- Implementing Identity on ASP.NET Core (Templates)
- ASP.NET Core External Provider Authentication
- Open Source Authentication Providers
- Enabling ASP.Net Core Identity
- Asp.Net Core Token-based Authentication
- JWT-JSON Web Token
- Configuring JSON Web Token Authentication
- Creating JWT Authentication
- Using Jquery to Access JWT
- IdentityServer4 Authentication
- Implement ASP.NET Identity with IdentityServer
- Configure Windows Authentication
- Windows Authentication
- Impersonation
- ASP.NET Core Authorization
- ASP.NET Core Role-based Authorization
- ASP.NET Core Role Authorization Policy
- Claim-based Authorization
- Custom Policy-based Authorization
- Resource-based Authorization
- View-based Authorization
- Authentication and Authorization: MVC
- Authentication and Authorization
- MVC Authentication Filter
- Implementing Single Sign-On
- Authentication using Third-party Identity Provider
- Implement Page Access Control with Standard Action Filters

- Authentication and Authorization Defensive Techniques: Web Forms
- Securing Forms Authentication Tickets
- Use Strong Hashing Algorithms to Validate Data
- Use Strong Encryption Algorithm to Secure Form Authentication Data
- Secure Form Authentication Cookies using SSL
- Securing Forms Authentication Credentials
- Preventing Session Hijacking using Cookieless Authentication
- Avoiding Forms Authentication Cookies from Persisting using DisplayRememberMe Property
- Avoiding Forms Authentication Cookies from Persisting using RedirectFromLoginPage Method
- Avoiding Forms Authentication Cookies from Persisting using SetAuthCookie Method
- Avoiding Forms Authentication Cookies from Persisting using GetRedirectUrl Method
- Avoiding Forms Authentication Cookies from Persisting using FormsAuthenticationTicket Constructor
- Securing Passwords with minRequiredPasswordLength
- Securing Passwords with minRequiredNonalphanumericCharacters
- Securing Passwords with passwordStrengthRegularExpression
- Restricting Number of Failed Logon Attempts
- Securing Application by using Absolute URLs for Navigation
- Securing Applications from Authorization Bypass Attacks
- Creating Separate Folder for Secure Pages in Application
- Validating Passwords on CreateUserWizard Control using Regular Expressions

- Authentication and Authorization Defensive Techniques: ASP.NET Core
- Configure Identity Services
- Password Policy
- User Lockout
- Sign in
- Configure Identity User Validation Settings
- Configure Application's Cookie Settings
- Configure Identity Services: Cookie Settings
- Enforcing SSL
- HTTP Strict Transport Security (HSTS)
- Authentication and Authorization Defensive Techniques: MVC
- Implement AllowXRequestsEveryXSecondsAttribute to Prevent Brute Force Attack
- MVC Page Access Control: Custom Security Filter
- Page Access Control: Third-party Libraries
- Implementing Control-level Protection
- Implementing Account Lockout
- Forcing HTTPS Protocol using [RequireHttps] • Implement AllowAnonymous Action Filter

➢ Module 6: Secure Coding Practices for Cryptography.

- Cryptographic
- Ciphers
- Block Cipher Modes
- Symmetric Encryption Keys
- Asymmetric Encryption Keys
- Functions of Cryptography
- Use of Cryptography to Mitigate Common Application Security Threats

Asia Masters Center

- Cryptographic Attacks
- Techniques Attackers Use to Steal Cryptographic Keys
- What should you do to Secure .NET Applications from Cryptographic Attacks?
- .NET Cryptography Namespaces
- .NET Cryptographic Class Hierarchy
- Symmetric Encryption
- SymmetricAlgorithm Class
- Members of the SymmetricAlgorithm Class
- Programming Symmetric Data Encryption and Decryption in .NET
- Symmetric Encryption: Defensive Coding Techniques
- Securing Information with Strong Symmetric Encryption Algorithm
- Vulnerability in using ECB Cipher Mode
- Padding
- Padding Modes
- None
- Zero Padding
- PKCS #7 Padding
- ANSIX923 Padding
- ISO10126 Padding
- Problem with Zeros Padding
- Securing Symmetric Encryption Keys from Brute Force Attacks
- Resisting Cryptanalysis Attack using Large Block Size
- Generating Non-Predictable Cryptographic Keys using RNGCryptoServiceProvider
- Storing Secret Keys and Storing Options
- Protecting Secret Keys with Access Control Lists (ACLs)

- Protecting Secret Keys with DPAPI
- Self Protection for Cryptographic Application
- Encrypting Data in the Stream using CryptoStream Class
- Asymmetric Encryption
- AsymmetricAlgorithm Class
- Members of the AsymmetricAlgorithm Class
- Programming Asymmetric Data Encryption and Decryption in .NET
- Asymmetric Encryption: Defensive Coding Techniques
- Securing Asymmetric Encryption using Large Key Size
- Storing Private Keys Securely
- Problem with Exchanging Public Keys
- Exchanging Public Keys Securely
- Asymmetric Data Padding
- Protecting Communications with SSL
- Hashing
- Hashing Algorithms Class Hierarchy in .NET
- Hashing in .Net
- Members of the HashAlgorithm Class
- Programming Hashing for Memory Data
- Programming Hashing for Streamed Data
- Imposing Limits on Message Size for Hash Code Security
- Setting Proper Hash Code Length for Hash Code Security
- Message Sizes and Hash Code Lengths Supported by the .NET Framework Hashing Algorithms
- Securing Hashing using Keyed Hashing Algorithms
- Digital Signatures
- Attacker's Target Area on Digital Signatures
- Security Features of Digital Signatures

- .NET Framework Digital Signature Algorithms
- Digital Certificates
- .NET Support for Digital Certificates
- X509Store
- X509Certificate and X509Certificate2
- X509Certificate2 Collection
- Programming Digital Signatures using Digital Certificates
- XML Signatures
- Need for Securing XML Files
- Securing XML Files using Digital Signatures
- Programming a Digital Signature for a Sample XML File
- ASP.NET Core Specific Secure Cryptography Practices
- ASP.NET Core Data Protection
- Data Protection Machine-wide Policy
- Data Protection Configuration
- Key Persistence
- Key Lifetime
- Application Name
- Automatic Key Generation
- Algorithm
- Generating a Random String
- Hashing String
- Storing App Secrets in Secure Place
- Securing Application settings using Azure Key Vault

➢ Module 7: Secure Coding Practices for Session Management.

- Session Management
- Types of Tokens
- Session Tokens
- Authentication Tokens

- Basic Security Principles for Session Management Tokens
- Common Threats to Session Management
- Session Hijacking Attack
- Account Hopping Attack
- Session Fixation Attack
- Token Prediction Attack
- Token Brute-force Attack
- Cross-site Request Forgery Attack
- Cross-site Scripting Attack
- Session Replay Attack
- Token Manipulation Attack
- Phishing Attack
- ASP.NET Session Management Techniques
- Client-Side State Management
- Client-Side State Management using Cookies
- Client-Side State Management using Hidden Fields
- Client-Side State Management using ViewState
- Client-Side State Management using Control State
- Client-Side State Management using Query Strings
- Server-Side State Management
- Server-Side State Management using Application Object
- Server-Side State Management using Session Object
- In Process Mode
- Out-of-Process Session Mode (State Server Mode)
- SQL-backed Session State o Server-side State Management Using Profile Properties
- Defensive Coding Practices against Broken Session Management
- Session Hijacking
- Securing ASP.NET Application from Session Hijacking

Asia Masters Center

- Implementing SSL to Encrypt Cookies
- Setting a Limited Time Period for Expiration
- Avoid using Cookieless Sessions
- Avoid using UseUri Cookieless Sessions
- Avoid Specifying Cookie Modes to AutoDetect
- Avoid Specifying Cookie Modes to UseDeviceProfile
- Enabling regenerateExpiredSessionID for Cookieless Sessions
- Resetting the Session when User Logs Out
- Token Prediction Attack
- Generating Lengthy Session Keys to Prevent Guessing
- Session Replay Attack
- Defensive Techniques for Session Replay Attack
- Session Fixation
- Session Fixation Attack
- Securing ASP.NET Application from Session Fixation Attack
- Cross-site Script Attack on Sessions
- Preventing Cross-site Scripting Attack using URL Rewriting
- Rewrite the application URL for each session
- Expiring application URLs automatically
- Preventing Session Cookies from Client-side Scripts Attacks
- Cross-site Request Forgery Attack
- Implementing the Session Token to Mitigate CSRF Attacks
- Additional Defensive Techniques to Mitigate CSRF Attack
- Cookie-based Session Management
- Persistent Cookies Information Leakage
- Avoid Setting the Expire Attribute to Ensure Cookie Security
- Ensuring Cookie Security using the Secure Attribute
- Ensuring Cookie Security using the HttpOnly Attribute
- ViewState-based Session Management

- ViewState Data Tampering Attack
- ViewState oneClick Attacks
- Securing ViewState
- Securing ViewState with Hashing
- Securing ViewState with Encryption
- Securing ViewState by Assigning User-specific Key
- ASP.NET CORE: Secure Session Management Practices
- Enabling Session State
- Implementing the CSRF Token to Mitigate CSRF Attacks
- Mitigating CSRF Attacks in JavaScript, AJAX and Single Page Applications
- Angular-Antiforgery Integration -AJAX
- Improve Session Security with Nwebsec Session Security Library •
- Checklist for Secure Session Management

➢ Module 8: Secure Coding Practices for Error Handling.

- What are Exceptions/Runtime Errors?
- Handled Exceptions
- Unhandled Exceptions
- Need of Secure Error/Exception Handling
- Consequences of Detailed Error Message
- Exposing Detailed Error Messages
- Considerations: Designing Secure Error Messages
- Secure Exception Handling
- Handling Exceptions in an Application
- Code-Level Exception Handling
- Page-Level Exception Handling
- Application-Level Exception Handling
- Defensive Coding practices against Information Disclosure

Asia Masters Center

- Avoid Displaying Detailed Error Messages
- Defensive Coding practices against Improper Error Handling
- Avoid Throwing Generic Exceptions
- Avoid Catching Generic Exceptions
- Avoid Swallowing the Exceptions
- Cleanup Code Vulnerability
- Vulnerability in Re-throwing Exception
- Managing Unhandled Errors
- Unobserved Exception Vulnerability
- ASP.NET Core: Secure Error Handling Practices
- ASP.NET Core Error Handling
- Inspect Exception During Development
- Implement Custom Error Handler
- Configure Pages with HTTP Status Codes
- Startup Exception Handling
- Do’s and Don’ts in Exception Handling
- Checklist for Proper Exception Handling
- Secure Auditing and logging
- What is Logging and Auditing?
- Need of Secure Logging and Auditing
- Common Threats to Logging and Auditing
- Denial of Service
- Log Wiping
- Log Bypass
- Log Tampering
- What Should be Logged?
- What Should NOT be Logged?
- Where to Perform Event Logging?
- File-System-based Logging System

- Database-based Logging System
- Performing Log Throttling in ASP.NET Health Monitoring System
- Tracing in .NET
- Writing Trace Output to Windows Event Log using EventLogTraceListener
- Tracing Security Concerns and Recommendations
- Secure Auditing and Logging Best Practices
- Protecting Log Records
- Fixing the Logs
- Auditing and Logging Security Checklists

➢ Module 9: Static and Dynamic Application Security Testing.

- Static Application Security Testing
- Static Application Security Testing (SAST)
- Objectives of SAST
- Why SAST
- Skills required for SAST
- What to look for in SAST
- Common Vulnerabilities Identified Through SAST
- Types of SAST
- Automated Source Code Analysis
- Manual Source Code Review
- Where does Secure Code Review Fit in SDLC?
- SAST Steps
- SAST Activities-flow Chart
- Recommendation for Effective SAST
- SAST Deliverable
- Automated Source Code Analysis
- Static Code Analysis Using Checkmarx Static Code Analysis

- Static Code Analysis Using Visual Code Grepper (VCG)
- Static Code Analysis Using HP Fortify
- Static Code Analysis Using Rational AppScan Source Edition
- Selecting Static Analysis Tool
- Manual Secure Code Review
- Manual Secure Code Review for Most Common Vulnerabilities
- Code Review for PCI DSS Compliance
- Code Review for Blacklisting Validation Approach
- Code Review for Client Side Validation Approach
- Code Review for Non-parametrized SQL Query
- Review Code for Non-parameterized Stored Procedure
- Code Review for XSS Vulnerability
- Review Code for Unvalidated Redirects and Forwards
- Code Review for Weak Password Authentication
- Code Review for Hard-Coded Passwords
- Code Review for Clear-text credentials in for Authentication
- Code Review for Unencrypted Form Authentication Tickets
- Code Review for Clear-text Connection strings
- Code Review for Weak Password Length
- Code Review for Inappropriate Authorization
- Code Review for use of Weak Hashing Algorithm
- Code Review for use of Weak Encryption Algorithm
- Code Review for Use of SSL
- Code Review for use of URL for Storing Session Tokens
- Code Review for Cookies Persistence
- Code Review for Allowing More Number of Failed Login attempts
- Code Review for providing Relative path to Redirect Method
- Code Review for Use of Server.Transfer() Method

- Code Review for Keeping both Public and Restricted pages in Same folder
- Code Review for use of Weak Encryption Algorithm
- Code Review for use of ECB Cipher Mode
- Code Review for use of Zero Padding
- Code Review for use of Small Key Size
- Code Review for use of Small Block Size
- Code Review for Cryptographic Keys Generation Mechanism
- Code Review for Sensitive Information Leakage
- Code Review for Generic Exception Throwing and Catching
- Code Review for use of Unencrypted Cookies
- Code Review for Overly Long Sessions
- Code Review for Cookieless Sessions
- Code Review for regeneration of Expired Sessions
- Code Review for weak Session Key Generation Mechanism
- Code Review for Cookies Vulnerable to Client-side Scripts attacks
- Code Review for Cookies Vulnerable to CSRF Attacks
- Code Review for ViewState Security
- Code Review for allowOverride Attribute
- Code Review for Enabling Trace feature
- Code Review for Enabling Debug feature
- Code Review for Validate Request
- Code Review: Check List Approach
- Sample Checklist
- Imput Validation
- Authentication
- Authorization
- Session Management

- Cryptography o Exception Handling
- Logging
- SAST Finding
- SAST Report
- SAST Reporting
- Dynamic Application Security Testing
- Types of DAST
- Automated Application Vulnerability Scanning
- Manual Application Penetration Testing
- SAST vs DAST
- Automated Application Vulnerability Scanning Tools
- Web Application Security Scanners
- WebInspect
- IBM SecurityAppScan
- Additional Web Application Vulnerability Scanners
- Proxy-based Security Testing Tools •
- Burp Suite
- OWASP Zed Attack Proxy (ZAP)
- Additional Proxy-based Security Testing Tools
- Choosing Between SAST and DAST

➢ Module 10: Secure Deployment and Maintenance.

- Secure Deployment
- Prior Deployment Activity
- Check the Integrity of Application Package Before Deployment
- Review the Deployment Guide Provided by the Software Vendor
- Deployment Activities: Ensuring Security at Various Levels
- Host Level Deployment Security
- IIS level Deployment Security
- SQL Server Level Deployment Security

- Ensuring Security at Host Level
- Check and Configure the Security of Machine Hosting Web Server, Application Server, Database Server and Network Devices
- Physical Security
- Host Level Security
- Ensuring Security at Network Level
- Network level Security
- Router
- Firewall
- Switch
- Ensuring Security at Application Level
- Web Application Firewall (WAF)
- Benefits of WAF
- WAF Limitations
- WAF Vendors
- Ensuing Security at IIS level
- Configure IIS Server Request Filtering Feature
- Editing Request Filtering and Request Limits
- Allowing or Denying a File Name Extension in Request Filtering
- Adding a Hidden Segment in Request Filtering
- Adding Limits for HTTP Headers in Request Filtering
- Denying an HTTP Verbs in Request Filtering
- Setting Request Filtering Attributes using appcmd Sites and Virtual Directories
- Website Location
- Script Mapping
- Anonymous Internet User Account
- Auditing and Logging

- Web Permissions
- IP Address and Domain Name Restrictions
- Authentication
- Parent Path Setting
- Microsoft FrontPage Server Extensions
- ISAPI Filters
- Ensuring Security at .NET Level
- Web.config and Machine.config Deployment Security Settings
- Verify the Configuration Settings
- Verify Lock Per-machine Settings
- Verify trace Element Setting
- Verify CustomError Settings
- Verify maxRequestLength Setting
- Verify debug Settings
- Verify protection Setting •
- Verify timeout Setting
- Verify requireSSL Setting
- Verify passwordFormat Setting
- Verify slideExpiration Setting
- Verify Name and Path Attribute Setting
- Verify Authorization Element Setting
- Verify Identity Element Setting
- Verify roleManager Setting
- Verify cookieProtection Setting
- Verify cookieRequireSSL Setting
- Verify cookieTimeout Setting
- Verify createPersistentCookie Setting
- Verify sessionState Settings
- Verify decryptionKey and validationKey Setting

Asia Masters Center

- Verify decryptionKey and validationKey Setting in Web Farm
- Verify validation Setting
- Verify trust Element Setting
- Verify httphandlers Settings
- Verify processModel Settings
- Verify healthMonitoring Setting
- Ensuring Security at SQL Server Level
- Selecting Authentication Mode in SQL Server
- Secure Mixed Mode Authentication
- Configure Password Enforcement Options for Standard SQL Server Logins
- Delete or Disable Unused Accounts
- Turn Off SQL Server Browser Service
- Disable Unnecessary Features and Services
- Service Account Management and Selection
- Manage Privileged Access
- Hiding SQL Server Instance
- Implement Encryption
- Implement Transparent Data Encryption
- Configure SSL in SQL Server
- Secure the Auditing Process
- Security Maintenance and Monitoring
- Post Deployment Activities: Security Maintenance and Monitoring
- Security Maintenance Activities at OS level
- Security Maintenance Activities at IIS level
- Security Maintenance Activities at Application level

The Feature Of Asia Master Training And Development Center

- we pick up the customer from the airport to the hotel.
- we give the participant training bag includes all the necessary tools for the course.
- Working within groups to achieve the best results.
- All our courses are confirmed and we do not postpone or cancel the courses regardless of the number of participants in the course.
- We can assist you in booking hotels at discounted prices if you wish to book through us.
- We offer the certificate from Asia Masters Center for Training and Administrative Development.

The Cost Of The Training Program Includes The Following:

1) Scientific article on flash memory.
2) Training Room.
3) Training.
4) Coffee break.
5) The training bag includes all the tools for the course.

Asia Masters Center

Price (USD)
Communicate with the training department to know the participation fees ➢ There are offers and discounts for groups
<u>**The details of the bank account**</u>
Bank name: CIMB Bank Berhad Account name: Asia Masters Center SDN. BHD Bank account number: 80-0733590-5 Swift code: CIBBMYKL IBAN: Null