



EC-COUNCIL CERTIFIED SECURE PROGRAMMER



Course Description

EC-Council Certified Secure Programmer lays the basic foundation required by all application developers and development organizations to produce applications with greater stability and posing lesser security risks to the consumer. The Certified Secure Application Developer standardizes the knowledge base for application development by incorporating the best practices followed by experienced experts in the various domains.

The distinguishing aspect of ECSP is that unlike vendor or domain specific certifications, it exposes the aspirant to various programming languages from a security perspective. This drives greater appreciation for the platform / architecture / language one specializes on as well as an overview on related ones.

Who Should Attend

The ECSP certification is intended for programmers who are responsible for designing and building secure Windows/Web based applications with .NET/Java Framework. It is designed for developers who have C#, C++, Java, PHP, ASP, .NET and SQL development skills.

Prerequisites

You must have programming fundamental knowledge.

Duration

5 days (9:00 – 5:00)

Certification

The ECSP 312-92 exam will be conducted on the last day of training. Students need to pass the online Prometric exam to receive the ECSP certification.

Course Outline v2

Module I: Introduction to Secure Coding

- Software Security Scenario
- Secure Coding
- Common Security Mistakes
- Why Security Mistakes Are Made
- Need for Secure Programming
- Building Blocks of Software Security
- Types of Security Vulnerabilities
- Vulnerability Cycle
- Types of Attacks
- Hackers and Crackers or Attackers
- Risk Assessment and Threat Modeling
- STRIDE Threat Model
- Common Criteria
- Security Architecture
- Security Principles
- Secure Development Checklists
- Use of Privilege
 - Data, Configuration, and Temporary Files
 - Network Port Use
 - Audit Logs

- User-Server Authentication

Module II: Designing Secure Architecture

- Introduction
- Secure Architecture
- Application Security
- Factors Affecting Application Security
- Software Engineering and System Development Life Cycle (SDLC)
- Different Phases of Software Development Life Cycle
 - System Requirements
 - Specifications
 - Design
 - Coding
 - Testing
 - Integration Testing
 - Maintenance
- Software Methodology Models
 - Waterfall Model
 - RAD (Rapid Application Development)
 - JAD (Joint Application Development)
 - Fountain Model
 - Spiral Model
 - Build and Fix
 - Synchronize-and-Stabilize
- Agile Methodologies
- Extreme Programming (XP)

- XP Practices
- The Rules and Practices of Extreme Programming
- Unified Modeling Language (UML)
 - Primary Goals
 - Diagram
 - UML Tool
 - Rational Rose
- Vulnerabilities and Other Security Issues in a Software Application
 - Security Through Obscurity
 - Buffer Overflows
 - Format String Vulnerabilities/ Race Conditions
 - Locking Problems
 - Exception Handling
 - Fundamentals of Control Granularity
 - Concepts Of Fail Safe Design Strategies
 - Fail Safe Design Strategies
 - Fault Tolerance and Detection
 - Fault Removal and Avoidance
 - Input and Parameter Validation
 - Encrypting Secrets in Memory and Storage
 - Scrubbing Information
 - Privilege Levels for Information Access
 - Loose Coupling
 - High Cohesion
 - Change Management and Version Control
- Best Practices for Software Development Projects

Module III: Cryptography

- Introduction to Cryptography
 - Encryption
 - Decryption
- Use of Cryptography
- Classical Cryptographic Techniques
- Modern Cryptographic Techniques
- Cipher
- RSA (Rivest Shamir Adleman)
 - Example of RSA Algorithm
 - RSA Attacks
 - RSA Challenge
 - Implementation of RSA in C++
- Data Encryption Standard (DES)
 - DES Overview
 - Implementation of DES in Java
- RC4, RC5, RC6, Blowfish
 - RC5
- Blowfish Algorithm in C
- Message Digest Functions
 - One-way Hash Functions
 - MD5
 - Implementation of MD5 in Java
- SHA (Secure Hash Algorithm)
 - SHA Implementation in Java

- SSL (Secure Sockets Layer)
- What is SSH?
 - SSH (Secure Shell)
- Algorithms and Security
- Disk Encryption
- Government Access to Keys (GAK)
- Digital Signature
 - Components of a Digital Signature
 - Method of Digital Signature Technology
 - Use of Digital Signature
 - Digital Signature Standard
 - Digital Signature Algorithm: Signature Generation/Verification
 - Digital Signature Algorithms: ECDSA, ElGamal Signature Scheme
 - Challenges and Opportunities
- Digital Certificates
 - Creating and Verifying a Simple XML Digital Signature in C#
 - Cleversafe Grid Builder <http://www.cleversafe.com/>
- PGP (Pretty Good Privacy)
- CypherCalc
- Command Line Scriptor
- CryptoHeaven
- Cryptanalysis
- Cryptography Attacks
- Brute-Force Attack
- Use Of Cryptography

Module IV: Buffer Overflows

- Buffer Overflows
- Reasons for Buffer Overflow Attacks
- Why are Programs/Applications Vulnerable?
- Understanding Stacks
- Understanding Heaps
- Types of Buffer Overflows: Stack-based Buffer Overflow
 - A Simple Uncontrolled Overflow of the Stack
 - Stack Based Buffer Overflows
- Types of Buffer Overflows: Heap-based Buffer Overflow
 - Heap Memory Buffer Overflow Bug
 - Heap-based Buffer Overflow
- How to Detect Buffer Overflows in a Program
 - Attacking a Real Program
- Defense Against Buffer Overflows
 - Tool to Defend Buffer Overflow: Return Address Defender (RAD)
 - Tool to Defend Buffer Overflow: StackGuard
 - Tool to Defend Buffer Overflow: Immunix System
 - Vulnerability Search – ICAT
 - Valgrind
 - Insure++
- Buffer Overflow Protection Solution: Libsafe
 - Comparing Functions of libc and Libsafe
- Simple Buffer Overflow in C
 - Code Analysis

Module V: Secure C and C++ Programming

- Introduction of C/C++
- Vulnerable C/C++ Functions
 - Strcpy()
 - Strncat()
 - Strncpy()
 - Sprintf()
 - Gets()
- C/C++ Vulnerabilities
 - Buffer Overflow
 - Strings
 - Countermeasures
 - Integer Vulnerabilities
 - Truncation
 - Sign Error
 - Countermeasures
 - Pointer Subterfuge
 - Dynamic Memory Management
 - Stack Smashing
 - GCC Extension to Protect Stack-Smashing Attacks
 - Heap-Based Buffer Overflow
 - Off By One/Five Errors
 - Double Free Vulnerability
- Secure Memory Allocation Tips
- Symmetric Encryption

- Symmetric Encryption in C++
- Blowfish Algorithm in C
- Public Key Cryptography
 - Public Key Cryptography in C++
- Networking
 - Creating an SSL Client in C++
 - Creating an SSL Server
- Random Number Generation Problem
- Anti-Tampering
 - Anti-Tampering Techniques
- Erasing Data from Memory Securely using C/C++
- Preventing Memory From Being Paged to Disk
- Using Variable Arguments Properly
- Signal Handling
- Encapsulation in C++
- Best Practices for Input Validation
- Code Profiling And Memory Debugging Tool: Val grind

Module VI: Secure Java and JSP Programming

- Introduction to Java
- JVM
- Java Security
- Sandbox Model
- Security Issues with Java
 - SQL Injection Attack
 - SQL Injection using UNION

- Preventive Measures for SQL Injection
 - URL Tampering
 - Denial-of-Service (DoS) Attack on Applet
 - Sample Code for DoS Attack
 - DoS by Opening Untrusted Windows
 - Preventing DOS Attacks
 - .Class File Format
 - Byte Code Attack
 - Reverse Engineering/ Decompilation by Mocha
 - Obfuscation Tools: Jmangle
 - Cinnabar Canner
- Byte Code Verifier
- Class Loader
 - Building a SimpleClassLoader
- Security Manager
- jarsigner - JAR Signing and Verification Tool
- Signing an Applet Using RSA-Signed Certificates
 - Signing Tools
 - Getting RSA Certificates
 - Bundling Java Applets as JAR Files
 - Signing Java Applets Using Jarsigner
 - Signing Java Applets Using Netscape Signing Tool
- Security Extensions
 - Java Authentication and Authorization Service (JAAS)
 - Java Cryptographic Extension (JCE)
 - Java Cryptography Architecture

- JCE: Pseudo Code for Encryption
- JCE: Pseudo Code for Decryption
- Sample Code for Encryption and Decryption
- Java(TM) Secure Socket Extension (JSSE)
- Creating Secure Client Sockets
- Creating Secure Server Sockets
- Choosing the Cipher Suites
- Java GSS Security
 - Code for GSS Server
 - Code for GSS Client
 - Problem of Untrusted User Input
- Security From Untrusted User Input
- Cross Site Scripting
 - Overcoming Cross Site Scripting Problem
- Permissions in Java
 - How to create new types of permissions?
- Security Policy
 - Specifying an additional Policy File at runtime
 - Policy Tool
 - Policy Tool: Creating a new Policy File
- Best practices for developing secure Java Code

Module VII: Secure Java Script and VB Script Programming

- Script: Introduction
- JavaScript Vulnerability
 - Cross-Site Scripting (XSS)

- How to Avoid XSS?
- JavaScript Hijacking
 - Defending Against JavaScript Hijacking
 - Decline Malicious Requests
 - Prevent Direct Execution of the JavaScript Response
 - Malicious Script Embedded in Client Web Requests
 - Tool: Thicket Obfuscator for JavaScript
- JavaScript Security in Mozilla
 - JavaScript Security in Mozilla: Same Origin Policy
 - Same Origin Check
 - JavaScript Security in Mozilla: Signed Script Policy
- Netscape's SignTool
 - Netscape's SignTool: Signing a File
- Privileges
- Tool for Encryption: TagsLock Pro
- JavaScript Shell (Jash): Javascript Command-Line Debugging Tool
- Tool: Script Encoder
- Tool: Scrambler
- VBScript: CryptoAPI Tools
- Signing A Script (Windows Script Host)
- Verifying a Script
- Signature Verification Policy
- Software Restriction Policies for Windows XP
- Step-by-Step Guide for Designing a Software Restriction Policy
- Step-by-Step Guide for Creating Additional Rules
- Rule for Blocking Malicious Scripts

Module VIII: Secure ASP Programming

- ASP- Introduction
- ASP Design Problems
- Improving ASP Design
 - Using Server-Side Includes
 - Using Server-Side Includes: Example
 - Using Server-Side Includes: Protecting the Contents of Include Files
 - Taking Advantage of VBScript Classes
 - Using Server.Execute
 - Using Server.Transfer
- #include Directive
- .BAK Files on the Server
- Programming Errors
 - Detecting Exceptions with Scripting Language Error-Handling Mechanisms
 - Using VBScript to Detect an Error
 - Using Jscript to Detect an Error
- Notifying the Support Team When an Error Occurs Using CheckForError
- Attacks on ASP
- ASP DypsAntiSpam: A CAPTCHA for ASP
 - How To Prevent Automatic Submission With DypsAntiSpam
 - CAPTCHA: Examples
- How to Use Database and ASP Sessions to Implement ASP Security
 - Step 1: Create A User Database Table
 - Step 2: Create And Configure The Virtual Directory
 - Step 3: Create The Sample Pages

- Step 4: Add Validation Code To Pages
- Protecting Your ASP Pages
 - Encoding ASP Code: Script Encoder
 - Protecting Passwords of ASP Pages with a One-way Hash Function
- ASP Best Practices
 - ASP Best Practices: Error Handling

Module IX: Secure Microsoft.NET Programming

- Common Terminology
- Microsoft .NET: Introduction
- .NET Framework
 - .NET Framework Security Policy Model
- Security Policy Levels
- Security Features in .NET
- Key Concepts in .NET Security
- Code Access Security (CAS)
- Evidence-Based Security
- Role-Based Security
 - Role-Based Security: Windows Principal
 - Role-Based Security: Generic principal
- Declarative and Imperative Security
- Cryptography
- Generate Key for Encryption and Decryption
 - Symmetric Encryption in .Net
 - Asymmetric Encryption in .Net
 - Symmetric Decryption in .Net

- Asymmetric Decryption in .Net
- Protecting Client and Server Data Using Encryption
- Cryptographic Signatures
 - Write a Signature in .Net
 - Verify a Signature in .Net
- Ensuring Data Integrity with Hash Codes
 - Hash Code Generation
 - Verification of Hash Code
- Permissions
 - Code Access Permissions
 - Identity Permissions
 - Role-Based Security Permissions
- SkipVerification
- Stack Walk
- Writing Secure Class Libraries
- Runtime Security Policy
- Step-By-Step Configuration of Runtime Security Policies
- Creating a Security Policy Deployment Package
- Type Safety
- Canonicalization
- Access Control List Editor
- Securing User Credentials and Logon Information
- Obfuscation
- Dotfuscator: .NET Obfuscator Tool
- Administration Tool: Authorization Manager (AzMan) with ASP.Net
- ASP.NET Security Architecture

- Authentication and Authorization Strategies
 - URL Authorization
 - File Authorization
 - Windows Authentication
 - Forms Authentication
 - Passport Authentication
 - Custom Authentication
 - Implementing Custom Authentication Scheme
- Configuring Security with Mscorcfg.msc
- Process Identity for ASP.NET
- Impersonation
 - Impersonation Sample Code
- Secure Communication
- Storing Secrets
 - Options for Storing Secrets in ASP.NET
- Securing Session and View State
- Web Form Considerations
- Securing Web Services
- Secure Remoting
 - Create a Remotable Object
- Secure Data Access
- .NET Security Tools
- Code Access Security Policy Tool
 - Caspol.exe
 - Caspol.exe Parameters
- Certificate Creation Tool: Makecert.exe

- Options in Makecert.exe
- Certificate Manager Tool: Certmgr.exe
- Certificate Verification Tool: Chktrust.exe
- Permissions View Tool: Permview.exe
- PEVerify Tool: Peverify.exe
- Best Practices for .NET Security

Module X: Secure PHP Programming

- Introduction to PHP (Hypertext Preprocessor)
 - PHP Security Blunders
 - Unvalidated Input Errors
 - Solution for Access Control Flaws
 - Solution for Session ID Protection
 - Error Reporting
 - Data Handling Errors
 - Security Sensitive PHP Functions: File Functions
 - Security Sensitive PHP Functions: ezmlm_hash
- PHP Vulnerabilities
 - Informational Vulnerabilities
 - Common File Name Vulnerability
 - Revealed Source Code Vulnerability
 - Revealing Error Message Vulnerability
 - Sensitive Data in Web Root Vulnerability
 - Session File in Shared Server Vulnerability
 - Sensitive Data in Globally Readable File Vulnerability
 - Revealing HTML Comment Vulnerability

- Web Application Fingerprint Vulnerability
- Packet Sniffing Vulnerability
- Attack Vulnerabilities
- Global Variable Vulnerability
- Default Password Vulnerability
- Online Backup Vulnerability
- Common PHP Attacks
 - Remote Code Execution
 - Cross-Site Scripting Attack (CSS)
 - Cross Site Scripting Attack: Example
 - Cross-Site Request Forgeries (CSRF, Sea-Surf or XSRF)
 - Workaround for Cross-Site Request Forgeries
 - SQL Injection
 - Defending SQL Injection Attacks
 - PHP Configuration Attacks
 - Preventing PHP Configuration Attacks
 - File System Attacks
 - Defending File System Attacks
 - Information Gathering Attacks
 - PHP Injection Attacks
- Secure PHP Practices
 - Safe Mode
 - Disable Register Globals
 - Validating Input
 - PHP Input Filter Class
- Best Practices for PHP Security

- PHP Tools
 - Acunetix Web Vulnerability Scanner
 - Encryption Software: PHP Code Lock
 - Zend Guard
 - POBS stands for PHP Obfuscator/Obscurer

Module XI: Secure PERL Programming

- Common Terminology
- Introduction: Practical Extraction and Report Language (PERL)
- Security Issues in Perl Scripts
- Basic User Input Vulnerabilities
- Overcoming Basic User Input Vulnerabilities
- Insecure Environmental Variables
- Algorithmic Complexity Attacks
- Perl: Taint, Strict, and Warnings
 - Taint Mode
 - How Does Taint Mode Work?
 - Taint Checking
 - Using Tainted Data
 - Securing the Program Using Taint
 - Strict Pragma
- Setuid
 - Setuid Sample Code
 - Setuid: Authenticating the user
 - Security bug with Setuid
- The Perl crypt() Function

- Logging Into a Secure Web Site with Perl Script
- Secure Log-in Checklist
- Program for Secure Log-in
- Securing open() Function
- Unicodes
- Displaying Unicode As Text

Module XII: Secure XML, Web Services and AJAX Programming

- Web Application and Web Services
- Web Application Vulnerabilities
 - Coding Errors
 - Design Flaws
- XML- Introduction
- XSLT and XPath
- XML Signature
 - Applying XML Signatures to Security
- An Enveloped, Enveloping and Detached XML Signature Simultaneously
- XML Encryption
 - The abstract <Encrypted-Type> Element
- Security Considerations for the XML Encryption Syntax
- Canonicalization
- Validation Process in XML
- XML Web Services Security
 - XML-aware Network Devices Expand Network Layer Security
- Security of URI in XML
- Security of Opaque Data in XML

- Growth of XML as Percentage of Network Traffic
- XML Web Services Security Best Practices
- XML Security Tools
 - V-Sentry
 - Vordel SOAPbox
- AJAX- Introduction
- Anatomy of an AJAX Interaction (Input Validation Example)
- AJAX: Security Issues
- How to Prevent AJAX Exploits
- Tool: HTML Guardian [™]
- Tool: Sprajax- AJAX Security Scanner
- Tool: DevInspect

Module XIII: Secure RPC, ActiveX and DCOM Programming

- RPC Introduction
 - RPC Authentication
 - RPC Authentication Protocol
 - NULL Authentication
 - UNIX Authentication
 - Data Encryption Standard (DES) Authentication
 - Data Encryption Standard (DES) Authentication on Server Side
 - Diffie-Hellman Encryption
 - Security Methods
 - Security Support Provider Interface (SSPI)
 - Security Support Providers (SSPs)
 - Writing an Authenticated SSPI Client

- Writing an Authenticated SSPI Server
- Secure RPC Protocol
- RpcServerRegisterAuthInfo Prevents Unauthorized Users from Calling your Server
- RPC Programming Best Practices
- Make RPC Function Calls
 - Making RPC Function Calls: Using Binding Handles
 - Making RPC Function Calls: Choose the Type of Binding Handles and Choose a Protocol Sequence
 - Use Context Handles
- Deal of RPC With Network
- Write a Secure RPC Client or Server
- ActiveX Programming: Introduction
 - Preventing Repurposing
 - SiteLock Template
 - IObjectSafety Interface
 - Code Signing
 - How to Create Your Own Code Signing Certificate and Sign an ActiveX Component in Windows
 - Protecting ActiveX Controls
- DCOM: Introduction
 - Security in DCOM
 - Application-Level Security
 - Security by Configuration
 - Programmatic Security
 - Run As a Launching user
 - Run As a Interactive User
 - Run As a Specific User

- Security Problem on the Internet
- Security on the Internet
- Heap Overflow Vulnerability
- Workarounds for Heap Overflow Vulnerability
- Tool: DCOMbobulator
- DCOM Security Best Practices

Module XIV Secure Linux Programming

- Introduction
- Is Open Source Good for Security?
- Linux – Basics
- Linux File Structure
- Basic Linux Commands
- Linux Networking Commands
- Linux Processes
- POSIX Capabilities
 - UTF-8 Security Issues
 - UTF-8 Legal Values
- Advantages of Security Functionality
 - Security Audit
 - Communication
 - Encryption
 - Identification and Authentication
 - Security Management
- Requirements for Security Measure Assurance
 - Enabling Source Address Verification

- iptables and ipchains
- Code to save the iptables state
- Controlling Access by MAC Address
- Permitting SSH Access Only
- Network Access Control
 - Layers of Security for Incoming Network Connections
 - Prohibiting Root Logins on Terminal Devices
 - Authentication Techniques
 - Authorization Controls
 - Running a Root Login Shell
 - Protecting Outgoing Network Connections
 - Logging in to a Remote Host
 - Invoking Remote Programs
 - Copying Remote Files
- Public-key Authentication between OpenSSH Client and Server
 - Authenticating in Cron Jobs
 - Protecting Files
 - File Permissions
 - Shared Directory
 - Encrypting Files
 - Listing Keyring
 - Signing Files
 - Encrypting Directories
- POP/IMAP Mail Server
- Testing an SSL Mail Connection
- Securing POP/IMAP with SSL and Pine

- SMTP Server
- Testing and Monitoring
 - Testing Login Passwords (John the Ripper)
 - Testing Login Passwords (CrackLib)
 - Testing Search Path
 - Searching Filesystems Effectively
 - Finding Setuid (or Setgid) Programs
 - Securing Device Special Files
 - Looking for Rootkits
 - Tracing Processes
 - Observing Network Traffic
 - Detecting Insecure Network Protocols
 - Detecting Intrusions with Snort
 - Log Files (syslog)
 - Testing a Syslog Configuration
 - Logwatch Filter
- Linux Security Best Practices
- Structure Program Internals and Approach
- Minimize Privileges Sample Code
- Filter Cross-Site Malicious Content on Input
- Filter HTML/URIs that may be Re-Presented
- Avoid Buffer Overflow
- Language-Specific Issues
 - C/C++
 - C/C++ (cont'd)
 - Dangers in C/C++

- Sample Codes
- Perl
- Perl (cont'd)
- Ada
- Java
- Java (cont'd)
- Tcl
- Tcl Sample Code
- PHP
- PHP (cont'd)
- Linux Security Tools
 - Linux Application Auditing Tool: grsecurity
 - grsecurity Configuration

Module XV: Secure Linux Kernel Programming

- Introduction
- What to do after Building Kernel?
- Linux Kernel Configuration Menu
- Steps to compile a Linux Kernel
 - Compiling the Kernel

Module XVI: Secure Xcode Programming

- Introduction to Xcode
- Mac OS X applications
 - Cocoa
 - Carbon

- AppleScript
- Script Editor
- Script Window
- CDSA
- Secure Transport API Set and Cryptographic Service Provider (CSP)
- Creating SSL Certificate on Mac OS X Server
 - Using SSL with the Web Server
 - Setting up SSL for LDAP
- Protecting Security Information
- Security in Mac OS X
- Security Management Using System Preferences
- Authentication Methods
- Encrypted disk images
- Networking Security Standards
- Personal firewall
- Checklist of recommended steps required to secure Mac OS X

Module XVII: Secure Oracle PL/SQL Programming

- Introduction: PL/SQL
- PL/SQL in Oracle Server
- Security Issues in Oracle
 - SQL Injection
 - Defending SQL Injection Attacks
 - SQL Manipulation
 - Code Injection Attack
 - Function Call Injection Attack

- Buffer Overflow and Other Vulnerabilities
- DBMS_SQL in PL/SQL
- Prevent DBMS_SQL in PL/SQL
- Types of Database Attacks
- Establishing Security Policies
- Password Management Policy
 - Password Management policy: Password History
- Auditing Policy
- Oracle Policy Manager
- Oracle Label Security (OLS)
- Create an Oracle Label Security Policy
 - Step 1: Define the Policy
 - Step 2: Define the Components of the Labels
 - Step 3: Identify the Set of Valid Data Labels
 - Step 4: Apply Policy to Tables and Schemas
 - Step 5: Authorize Users
 - Step 6: Create and Authorize Trusted Program Units (Optional)
 - Step 7: Configure Auditing (Optional)
- Using Oracle Label Security with a Distributed Database
- Oracle Identity Management
- Security Tools
- Secure Backups: Tool
- Encryption and Its Types: Obfuscation
- Obfuscation Sample Code
- Encryption Using DBMS_CRYPTO
- Advanced Security Option

- Row Level Security
- Oracle Database Vaults: Tool
- Auditing
 - Auditing Methods
 - Audit Options
 - View Audit Trail
 - Oracle Auditing Tools
 - Fine-Grained Auditing (FGA)
- Testing PL/SQL Programs
- SQL Unit Testing Tools: SPUnit
- SQL Unit Testing Tools: TSQLUnit
- SQL Unit Testing Tools: utPLSQL
- Steps to Use utPLSQL

Module XVIII: Secure SQL Server Programming

- Introduction
- SQL Server Security Model
 - SQL Server Security Model: Login
- Steps to Create a SQL Server Login
- Database User
- Guest User
- Permissions
- Database Engine Permissions Hierarchy
- Roles
 - Public Role
 - Predefined Roles

- Fixed Server Roles
- Fixed Database Roles
- User-Defined Roles
- Application roles
- Security Features of MS-SQL Server 2005
- SQL Server Security Vulnerabilities:
 - Buffer Overflow in pwdencrypt()
 - Extended Stored Procedures Contain Buffer Overflows
- SQL Injection
- Prevent SQL Injection
- Sqlninja:
 - SQL Server Injection & Takeover Tool
 - Finding Target
- Data Encryption
- Built-in Encryption Capabilities
- Encryption Keys
- Encryption Hierarchy
- Transact-SQL
- Create Symmetric Key in T-SQL
- Create Asymmetric Key in T-SQL
- Certificates
- Create Certificate in T-SQL
- SQL Server Security: Administrator Checklist
- Database Programming Best Practices
- SQL Server Installation
 - Authentication

- Authorization
- Best Practices for Database Authorization
- Auditing and Intrusion Detection
- How to Enable Auditing
- Database Security Auditing Tools:
 - AppDetective
 - NGSSquirrel
 - AuditPro

Module XIX: Secure Network Programming

- Basic Network Concepts:
 - Network
 - Protocols
 - Client Server Model
- Basic Web Concepts
- Network Programming
- Benefits of Secure Network Programming
- Network Interface
- How to Secure Sockets:
 - Server Program
 - Client Program
- Ports
- UDP Datagram and Sockets
- Internet Address
- How to connect to secure websites
- URL Decoder

- Reading Directly from a URL
- Content Handler
- Cookie Policy
- RMI Connector
- .Net : Internet Authentication
- Network Scanning Tool: ScanFi www.securecentral.com
- Network Programming Best Practices

Module XX: Windows Socket Programming

- Introduction
- Windows NT and Windows 2000 Sockets Architecture
- Socket Programming
- Client-Side Socket Programming
 - The Socket Address Structure
 - The Socket Address Structure: Code Analysis
- Initializing a Socket and Connecting
- Server-Side Socket Programming
- Creating a Server
- Winsock 2.0
- Winsock Linking Methods
- Starting a Winsock 2 API
- Accepting Connections:
 - AcceptEx
- WinSock: TransmitFile and TransmitPackets
- Grabbing a Web Page Using Winsock
- Generic File – Grabbing Application

- Writing Client Applications
- TCP Client Application Sample Code
- Writing Server Applications
- TCP Server Application Sample Code
- Winsock Secure Socket Extensions
 - WSADeleteSocketPeerTargetName
 - WSAImpersonateSocketPeer
 - WSAQuerySocketSecurity
 - WSAREvertImpersonation
 - WSASetSocketPeerTargetName
 - WSASetSocketSecurity Function
- SOCKET_SECURITY_SETTINGS
- Case Study: Using WinSock to Execute a Web Attack
- Case Study: Using Winsock to Execute a Remote Buffer Overflow
- MDACDos Application

Module XXI: Writing Shellcodes

- Introduction
- Shellcode Development Tools
- Remote Shellcode
- Port Binding Shellcode
- FreeBSD Port Binding Shellcode
- Clean Port Binding Shellcode
 - Clean Port Binding Shellcode: sckcode
- Socket Descriptor Reuse Shellcode
 - Socket Descriptor Reuse Shellcode in C

- Socket Descriptor Reuse Shellcode: Sample Code
- Local Shellcode
- execve
- Executing /bin/sh
- Byte Code
- setuid Shellcode
- chroot Shellcode
 - Breaking of chroot jails in Traditional Way
 - Breaking Out of Chroot Jails on Linux Kernels
- Windows Shellcode
- Shellcode Examples
- Steps to Execute Shell Code Assembly
- The Write System Call
 - Linux Shellcode for “Hello, world!”
 - The Write System Call in FreeBSD
- execve Shellcode in C
 - FreeBSD execve jmp/call Style
 - FreeBSD execve Push Style
 - FreeBSD execve Push Style, Several Arguments
- Implementation of execve on Linux
- Linux Push execve Shellcode
- System Calls
 - The Socket System Call
 - The Bind System Call
 - The Listen System Call
 - The Accept System Call

- The dup2 System Calls
- The execve System Call
- Linux Port Binding Shellcode
- Compile, Print, and Test Shellcode
- Reverse Connection Shellcode
- Socket Reusing Shellcode
- Linux Implementation of Socket Reusing Shellcode
- Reusing File Descriptors
- setuid Root
 - setuid Root: Executing the Program
 - setuid Root: System calls used by the program
- Using ltrace utility
- Using GDB
- Assembly Implementation
- SysCall Trace
- RW Shellcode
- Encoding Shellcode
- Decoder Implementation and Analysis
- Decoder Implementation Program
- Results of Implementation Program
- OS-Spanning Shellcode
- Assembly Creation

Module XXII: Writing Exploits

- Introduction

- Targeting Vulnerabilities
 - Remote and Local Exploits
 - A Two-Stage Exploit
- Format String Attacks
 - Example of a Vulnerable Program
- Using %n Character
- Fixing Format String Bugs
 - Case Study: xlockmore User-Supplied Format String Vulnerability CVE-2000-0763
- TCP/IP Vulnerabilities
- Race Conditions
 - File Race Conditions
 - Signal Race Conditions
- Case Study: 'man' Input Validation Error
- Writing Exploits and Vulnerability Checking Programs
 - Writing Exploits and Vulnerability Checking Programs Sample Code
- Stack Overflow Exploits
 - Memory Organization
 - Stack Overflows
 - Finding Exploitable Stack Overflows in Open-Source Software
 - Finding Exploitable Stack Overflows in Closed-Source Software
- Heap Corruption Exploits
 - Doug Lea Malloc
 - Freed Dmalloc Chunk
 - Vulnerable Program Example
 - Figures: Fake Chunk, Overwritten Chunk
- Case Study: OpenSSL SSLv2 Malformed Client Key Remote Buffer Overflow Vulnerability CAN-

2002-0656

- Exploitation
- Exploitation Sample Code
- The Complication
- Improving the Exploit
- Integer Bug Exploits
- Integer Wrapping
- Program: Addition-Based Integer Wrapping
- Multiplication-Based Integer Wrapping
- Bypassing Size Checks
 - Signed Size Check Without Integer Wrapping
- Using the Metasploit Framework
- Determining Attack Vector
- Finding the Offset: Overwriting the Return Address
- The First Attack String
- Overwriting EIP with a Known Pattern
- Selecting a Control Vector
- Finding a Return Address
- Selecting the Search Method in the Metasploit Opcode Database
- Search Method in Metasploit Opcode Database
- Using the Return Address
 - Inserting the Return Address
 - Verifying Return Address Reliability
- Nop Sleds: Increasing Reliability with a Nop Sled
- Choosing a Payload and Encoder

- Listing Available Payloads
- Determining Payload Variables
- Generating the Payload
- msfencode Options
- List of Available Encoders
- Choosing a Payload and Encoder: msfencode Results
- msfweb Payload Generation
- Setting msfweb Payload Options
- msfweb Generated and Encoded Payload
- Integrating Exploits into Framework

Module XXIII: Programming Port Scanners and Hacking Tools

- Port Scanner
 - Working of a Simple Port Scanner
 - Prerequisites for Writing a Port Scanner
 - Port Scanner in C++
 - Port Scanner in C#
 - Building a Simple Port Scanner in VC++
 - Port Scanner in Java
 - Example JavaScript Port Scanner
 - Port Scanner in ASP.Net
 - Port Scanner in Perl
 - Port Scanner in PHP
 - UDP Port Scanning in PHP
 - Port Scanner in XML
- Coding for Ethereal

- libpcap
 - Capturing Packets
- Packet Capturing Example
- Saving Captured Packets to a File
- The *wiretap* Library
- Adding a new file format to the *wiretap* library
- *wtap* Struct
- Setting up a New Dissector
- Programming the Dissector
- Adding a *tap* Module
- Coding for Nessus
 - Nessus Attack Scripting Language (NASL)
 - Writing Personal-Use Tools in NASL
 - Programming in the Nessus Framework
 - Porting to and from NASL
 - Porting to NASL
 - Porting from NASL
- Extending Metasploit
 - Metasploit Framework (MSF)
 - *msfiweb* Interface
 - Selecting the Exploit Module
 - *msfconsole* Interface
 - Using *msfconsole* Interface
 - Steps Involved in Executing an Exploit under *msfconsole*
 - *msfcli* Interface

- Using *msfcli* Interface
- Updating the MSF
- Writing Snort rules
 - Writing Basic Rules
 - The Rule Header
 - Rule Options
 - Writing Advanced Rules: Perl-Compatible Regular Expressions (PCRE)
 - Byte_test and Byte_jump
 - Optimizing Rules
 - Testing Rules
 - Writing Detection Plugins
- Netcat Source Code

Module XXIV: Secure Mobile phone and PDA Programming

- Mobile Phone Programming
- Different OS Structure in Mobile Phone
 - Symbian Operating System
 - Guidelines for Securing Symbian OS
 - PalmOS
 - PalmOS Vulnerabilities
 - HotSync Vulnerability
 - Creator ID Switching
 - Windows Mobile
 - Calling Secure Web Services
 - Security Practices for Windows Mobile Programming
- Comparison of Common Programming Tasks

- PDA Programming
 - PDA Security Issues
 - Security Policies for PDAs
 - PDA Security Products
 - PDA Security Vendors
- Java 2 Micro Edition(J2ME)
- J2ME Architecture
- J2ME Security Issues
 - CLDC Security
- Mobile Information Device Profile (MIDP)
 - MIDP Security
- Programming the BlackBerry With J2ME
- Security and Trust Services API (SATSA) for J2ME: The Security APIs
- Certificate Enrollment in SATSA
 - Generating a Private Key and Certificate Signing Request in SATSA
 - Requesting the Signed Certificate (Verifying the CSR)
 - Storing a Certificate into the Certificate Local Store
- Data Integrity with Message Digests
 - Generating a Message Digest
 - Verifying a Message Digest
- Authentication With Digital Signatures
 - Signing a byte Array for Authentication Purposes
 - Verifying a Digital Signature using SATSA
- Data Confidentiality - Using Ciphers for Data Encryption
 - Using Cipher to Encrypt Data using a Symmetric Encryption

- Using Cipher to Decrypt Data using a Symmetric Encryption
- Security Issues in Bluetooth
 - Security Attacks in Bluetooth Devices
- Bluetooth security
 - Bluetooth Security : Key Management
 - Tool: Bluekey
 - Tool: BlueWatch
 - Tool: BlueSweep
 - Tool: Bluediving
 - Tool: Smartphone Security Client
 - Tool: BlueFire Mobile Security Enterprise Edition
- Mobile Phone Security Tips
 - Defending Cell Phones and PDAs Against Attack
- Antivirus Tools for Mobile Devices
 - F-Secure Antivirus for Palm OS

Module XXV: Secure Game Designing

- Game Designing Introduction
- Type of Games
 - Console Games
 - Mobile Games
 - Online Games
 - Off-line Games
 - Wii Games
- Threats to Online Gaming
- Game Authoring Tools

- The 2D Shooter Game Creator
- Multimedia Fusion
- Adventure Game Studio
- Game Maker
- FPS Creator
- Stagecast Creator
- RPG Maker XP
- The Scrolling Game Development Kit
- Visual3D.NET
- Game Engine
- Best Practices for Secure Game Designing

Module XXVI: Securing E-Commerce Applications

- Purpose of Secure E-Commerce Application
- E-Business Concepts: Secure Electronic Transaction (SET)
 - Working of SET
- Secure Socket Layer (SSL)
 - SSL Certificates
 - VeriSign SSL Certificates
 - Entrust SSL Certificates
- Digital Certificates
- Digital Signature
 - Digital Signature Technology
 - Digital Signature Algorithm
 - Signature Generation/Verification

- ECDSA, ElGamal Signature Scheme
- HACKER SAFE® Certification
 - HACKER SAFE Technology
- Guidelines for Developing Secure E-Commerce Applications

Module XXVII: Software Activation, Piracy Blocking and Automatic Updates

- Software Activation: Introduction
 - Process of Software Activation
 - Software Activation: Advantages
 - Activation Explained
 - Online License Management Server
 - Activation Policies
 - Policy Control Parameters
- Piracy
 - Impacts of piracy
 - Piracy Blocking
 - Digital Right Management (DRM)
 - Software Piracy Protection Strategies
 - Copy protection for DVD
 - Application Framework –DVD Copy Protection System
 - Content Protection During Digital Transmission
 - Watermark System Design Issues
 - Economic Costs
 - False Positives Rate
 - Interaction with MPEG compression
 - Detector Placement

- Copy Generation Management
- Tool: Crypkey
- EnTrial Key Generation
- EnTrial Distribution File
- EnTrial Product & Package Initialization Dialog
- Windows Automatic Updates
 - Options for Setting up Windows Automatic Updates on XP
 - Automatic Updates Option on AVG Antivirus
 - Automatic Updates for Internet Explorer
 - Automatic Updates for Mozilla Firefox

Module XX VIII: Secure Application Testing

- Software Development Life Cycle (SDLC)
- Introduction to Testing
- Types of Testing
 - White Box Testing
 - Types of White Box Testing
 - Dynamic White-Box Testing
 - Integration Test
 - Regression Testing
 - System Testing
 - Black Box Testing
 - Load Testing
 - Strategies For Load Testing
 - Functional Testing

- Testing Steps
 - Creating Test Strategy
 - Creating Test Plan
 - Creating Test Cases and Test Data
 - Executing, Bug Fixing and Retesting
- Classic Testing Mistakes
- User Interface Errors
- What Makes a Good User Interfaces
- Use Automatic Testing and Tools
- Generic Code Review Checklist
- Software Testing Best Practices
- Testing Tools
 - QEngine
 - WinRunner
 - LoadRunner
- Real Time Testing

Module XXIX: Writing Secure Documentation and Error Messages

- Error Message
 - Common Error Messages
 - Error Messages: Categories
 - Characteristics of a Good Error Message
- Error Message in a Well-designed Application
- Example of Good Error Message
- Reasons for Different Perspectives for Error Messages
- Error Message Usability Checklist

- Guidelines For Creating Effective Error Messages
- Best Practices while Designing Error Messages
- Error Messages: Examples
- Security Issues in an Error Message
- Security Precautions in Documentation

© 2007 EC-Council. All rights reserved.

This document is for informational purposes only. EC-Council MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY. EC-Council logo is registered trademarks or trademarks of EC-Council in the United States and/or other countries.