# COMPLIANCE STANDARDS

UPDATED SEPTEMBER 2025

cmd+ctrl

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURITY PRINCIPLES** | | | | | | | | | |
| **AWA 101.** Fundamentals of Application Security | ✓ | ✓ | | ✓ | | | ✓ | ✓ | |
| **AWA 102.** Secure Software Concepts | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **AWA 106.** Building Secure Software: Overcoming Challenges in Application Security | ✓ | ✓ | | | | | | | |
| **AWA 107.** Building Secure Software: Foundations and Best Practices | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **AWA 108.** Building Secure Software: A Guide to Software Integration, Testing, and Deployment | ✓ | | ✓ | ✓ | ✓ | ✓ | | | |
| **ENG 110.** Essential Account Management Security | | | ✓ | | | | | | |
| **ENG 111.** Essential Session Management Security | | | ✓ | | | | | | |
| **ENG 112.** Essential Access Controls for Mobile Devices | | | ✓ | | | | | | |
| **ENG 113.** Essential Secure Configuration Management | | | ✓ | | | | | | |
| **ENG 114.** Essential Risk Assessment | | | ✓ | | | | | ✓ | |
| **ENG 115.** Essential System and Information Integrity | | | ✓ | | | | | | |
| **ENG 116.** Essential Security Planning Policy and Procedures | | | ✓ | | | | | | |
| **ENG 117.** Essential Information Security Program Planning | | | ✓ | | | | | | |
| **ENG 118.** Essential Cyber Incident Response Planning | | | ✓ | ✓ | | | | ✓ | |
| **ENG 119.** Essential Security Audit and Accountability | | | ✓ | | | | | | |
| **ENG 120.** Essential Personnel Security Policy and Procedures | | | ✓ | | | | | | |
| **ENG 121.** Essential Identification and Authentication | | | ✓ | | | | | | |
| **ENG 122.** Essential Physical and Environmental Protection | | | ✓ | | | | | | |
| **ENG 123.** Essential Secure Software Engineering Principles | | | ✓ | | | | | | |
| **ENG 124.** Essential Application Protection | | | ✓ | | | | | | |
| **ENG 125.** Essential Data Protection | | | ✓ | | | | | ✓ | |
| **ENG 126.** Essential Security Maintenance Policies | | | ✓ | | | | | | |
| **ENG 127.** Essential Media Protection | | | ✓ | | | | | | |
| **ENG 150.** Meeting Confidentiality, Integrity and Availability Requirements | | | | ✓ | | | | ✓ | |
| **ENG 151.** Fundamentals of Privacy Protection | | ✓ | ✓ | | | | | ✓ | |
| **SECURE DEVELOPMENT** | | | | | | | | | |
| **API 210.** Mitigating APIs Lack of Resources & Rate Limiting | | ✓ | ✓ | | | | | | |
| **API 211.** Mitigating APIs Broken Object Level Authorization | | ✓ | ✓ | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURE DEVELOPMENT *(Continued)*** | | | | | | | | | |
| **API 213.** Mitigating APIs Mass Assignment | | ✓ | ✓ | | | | | | |
| **API 214.** Mitigating APIs Improper Asset Management | | ✓ | ✓ | | | | | | |
| **API 351.** Securing Kubernetes in the Build and Release Stage | | ✓ | ✓ | | | | | | |
| **COD 110.** Fundamentals Secure Mobile Development | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **COD 141.** Fundamentals of Database Security | | | | ✓ | | | | ✓ | |
| **COD 152.** Fundamentals of Secure Cloud Development | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | |
| **COD 160.** Fundamentals of Secure Embedded Software Development | | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| **COD 170.** Identifying Threats to Mainframe COBOL Applications and Data | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 201.** Secure C Encrypted Network Communications | ✓ | ✓ | ✓ | ✓ | | | | | |
| **COD 202.** Secure C Run-Time Protection | ✓ | | ✓ | | | | | | |
| **COD 206.** Creating Secure C++ Code | ✓ | ✓ | | | | | | | |
| **COD 207.** Communication Security in C++ | ✓ | ✓ | ✓ | ✓ | | | | | |
| **COD 214.** Creating Secure GO Applications | ✓ | ✓ | | | | | | | |
| **COD 215.** Mitigating .NET Application Vulnerabilities (NEW) | | ✓ | ✓ | ✓ | | | | | |
| **COD 219.** Creating Secure Code SAP ABAP Foundations | ✓ | ✓ | | ✓ | | | | | |
| **COD 241.** Creating Secure Oracle Database Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 242.** Creating Secure SQL Server and Azure SQL Database Applications | | | | | | | | ✓ | |
| **COD 245.** Securing NoSQL Cloud Databases | ✓ | ✓ | ✓ | | | | | | ✓ |
| **COD 246.** PCI DSS Requirement 3: Protecting Stored Cardholder Data | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 247.** PCI DSS Requirement 3: Encrypting Transmission of Cardholder Data | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 248.** PCI DSS Requirement 6: Develop & Maintain Secure Systems & Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 249.** PCI DSS Requirement 11: Regularly Test Security Systems and Processes | | | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 251.** Defending AJAX-Enabled Web Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **COD 252.** Securing Google Platforms Applications & Data | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | |
| **COD 253.** Creating Secure AWS Cloud Applications | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | |
| **COD 254.** Creating Secure Azure Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 255.** Creating Secure Code Web API Foundations | ✓ | ✓ | ✓ | | ✓ | ✓ | | | |
| **COD 256.** Creating Secure Code Ruby on Rails Foundations | ✓ | ✓ | ✓ | | ✓ | ✓ | | | |
| **COD 257.** Creating Secure Python Web Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 258.** Creating Secure PHP Web Applications | | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 259.** Node.js Threats and Vulnerabilities | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 261.** Threats to Scripts | ✓ | ✓ | ✓ | ✓ | | | | ✓ | ✓ |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURE DEVELOPMENT *(Continued)*** | | | | | | | | | |
| **COD 262.** Fundamentals of Shell and Interpreted Language Security | ✓ | ✓ | | ✓ | | | | | |
| **COD 263.** Secure Bash Scripting | ✓ | ✓ | | ✓ | | | | | |
| **COD 264.** Secure Perl Scripting | ✓ | ✓ | | ✓ | | | | | |
| **COD 265.** Secure Python Scripting | ✓ | ✓ | | ✓ | | | | | |
| **COD 266.** Secure Ruby Scripting | ✓ | ✓ | | ✓ | | | | | |
| **COD 267.** Securing Python Microservices | ✓ | ✓ | | | | | | | |
| **COD 268.** Mitigating TypeScript Application Vulnerabilities | | ✓ | ✓ | ✓ | | | | | |
| **COD 270.** Creating Secure COBOL and Mainframe Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 283.** Java Cryptography | | ✓ | ✓ | | | | | | |
| **COD 284.** Secure Java Coding | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | |
| **COD 285.** Developing Secure Angular Applications | | ✓ | | ✓ | | | | | |
| **COD 286.** Creating Secure React User Interfaces | | ✓ | | ✓ | | | | | |
| **COD 287.** Java Application Server Hardening | ✓ | ✓ | ✓ | ✓ | | | | | |
| **COD 288.** Java Public Key Cryptogrpahy | | ✓ | ✓ | | | | | | |
| **COD 289.** Securing Java Spring APIs | ✓ | ✓ | | | | | | | ✓ |
| **COD 301.** Secure C Buffer Overflow Mitigations | ✓ | ✓ | | | | | | | |
| **COD 302.** Secure C Memory Management | ✓ | | | ✓ | | | | | ✓ |
| **COD 304.** Principles of C++ Memory Safety | | ✓ | ✓ | ✓ | | | | | |
| **COD 305.** C++ Secure Memory Management | | ✓ | ✓ | ✓ | | | | | |
| **COD 306.** C++ Memory Safety: Debugging Tools and Techniques | | ✓ | ✓ | ✓ | | | | | |
| **COD 303.** Common C Vulnerabilities and Attacks | ✓ | | ✓ | | | | | | |
| **COD 307.** Protecting Data in C++ | ✓ | ✓ | | | | | | | |
| **COD 308.** Common ASP.NET Vulnerabilities and Attacks | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| **COD 309.** Securing ASP.NET MVC Applications | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |
| **COD 310.** Securing ASP.NET Core Applications | ✓ | ✓ | ✓ | | | | | | ✓ |
| **COD 315.** Preventing Vulnerabilities in iOS Code in Swift | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 316.** Creating Secure iOS Code in Objective C | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 317.** Protecting Data on iOS in Swift | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 318.** Protecting Data on Android in Java | | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **COD 319.** Preventing Vulnerabilities in Android Code in Java | | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 321.** Protecting C# from Integer Overflows and Canonicalization Issues | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 322.** Protecting C# from SQL Injection | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURE DEVELOPMENT *(Continued)*** | | | | | | | | | |
| **COD 323.** Using Encryption with C# | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 324.** Protecting C# from XML Injection | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **COD 325.** Protecting Data in C# for .NET | ✓ | ✓ | ✓ | | | | | | ✓ |
| **COD 352.** Creating Secure JavaScript and jQuery Code | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 361.** HTML5 Security Threats | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 362.** HTML5 Built-In Security Features | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 363.** Securing HTML5 Data | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 364.** Securing HTML5 Connectivity | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| **COD 366.** Creating Secure Kotlin Applications | | ✓ | | ✓ | | | | | |
| **COD 380.** Preventing SQL Injection in Java | ✓ | ✓ | ✓ | | | | | | |
| **COD 381.** Preventing Path Traversal Attacks in Java | ✓ | ✓ | ✓ | | | | | | |
| **COD 382.** Protecting Data in Java | ✓ | ✓ | | | | | | | |
| **COD 383.** Protecting Java Backend Services | ✓ | ✓ | ✓ | ✓ | | | | | |
| **COD 384.** Protecting Java from Information Disclosure | ✓ | ✓ | ✓ | | | | | | |
| **COD 385.** Preventing Race Conditions in Java Code | ✓ | ✓ | ✓ | | | | | | |
| **COD 386.** Preventing Integer Overflows in Java Code | ✓ | ✓ | ✓ | | | | | | |
| **DES 207.** Mitigating OWASP API Security Top 10 | | ✓ | ✓ | | | | | | |
| **DES 208.** Defending Against the CSA Top 11 Threats to Cloud | | | ✓ | | | | | | |
| **DES 232.** Mitigating OWASP 2021 Injection | ✓ | ✓ | ✓ | ✓ | | | | | |
| **DES 233.** Mitigating OWASP 2021 Identification and Authentication Failures | ✓ | ✓ | ✓ | ✓ | | | | | |
| **DES 234.** Mitigating OWASP 2021 Cryptographic Failures | ✓ | ✓ | ✓ | ✓ | | | | ✓ | |
| **DES 235.** Mitigating OWASP 2021 Insecure Design | ✓ | ✓ | ✓ | | | | | | |
| **DES 236.** Mitigating OWASP 2021 Broken Access Control | ✓ | ✓ | ✓ | ✓ | | | | | |
| **DES 237.** Mitigating OWASP 2021 Security Misconfiguration | ✓ | ✓ | ✓ | ✓ | | | | | |
| **DES 238.** Mitigating OWASP 2021 Server-Side Request Forgery (SSRF) | ✓ | ✓ | ✓ | | | | | | |
| **DES 239.** Mitigating OWASP 2021 Software and Data Integrity Failures | | ✓ | | | | | | | |
| **DES 240.** Mitigating OWASP 2021 Vulnerable and Outdated Components | | ✓ | ✓ | ✓ | | | | | |
| **DES 241.** Mitigating OWASP 2021 Security Logging and Monitoring Failures | | ✓ | ✓ | ✓ | | | | | |
| **DES 250.** Secure Software Acceptance and Deployment | | | ✓ | | | | | | |
| **DES 270.** Mitigating OWASP Mobile Top 10 Risks | ✓ | ✓ | ✓ | | | | | | ✓ |
| **DES 271.** OWASP M1: Mitigating Improper Platform Usage | | ✓ | | | | | | | |
| **DES 272.** OWASP M2: Mitigating Insecure Data Storage | | ✓ | | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURE DEVELOPMENT** *(Continued)* | | | | | | | | | |
| **DES 273.** OWASP M3: Mitigating Insecure Communication | | ✓ | | | | | | | |
| **DES 274.** OWASP M4: Mitigating Insecure Authentication | | ✓ | | | | | | | |
| **DES 275.** OWASP M5: Mitigating Insufficient Cryptography | | ✓ | | | | | | | |
| **DES 276.** OWASP M6: Mitigating Insecure Authorization | | ✓ | | | | | | | |
| **DES 277.** OWASP M7: Mitigating Client Code Quality | | ✓ | | | | | | | |
| **DES 278.** OWASP M8: Mitigating Code Tampering | | ✓ | | | | | | | |
| **DES 279.** OWASP M9: Mitigating Reverse Engineering | | ✓ | | | | | | | |
| **DES 280.** OWASP M10: Mitigating Extraneous Functionailty | | ✓ | | | | | | | |
| **DES 281.** OWASP IoT1: Mitigating Weak, Guessable or Hardcoded Passwords | | ✓ | | | | | | | |
| **DES 282.** OWASP IoT2: Mitigating Insecure Network Services | | ✓ | | | | | | | |
| **DES 283.** OWASP IoT3: Mitigating Insecure Ecosystem Interfaces | | ✓ | | | | | | | |
| **DES 284.** OWASP IoT4: Mitigating Lack of Secure Update Mechanism | | ✓ | | | | | | | |
| **DES 285.** OWASP IoT5: Mitigating Use of Insecure or Outdated Components | | ✓ | | | | | | | |
| **DES 286.** OWASP IoT6: Mitigating Insufficient Privacy Protection | | ✓ | | | | | | | |
| **DES 287.** OWASP IoT7: Mitigating Insecure Data Transfer and Storage | | ✓ | | | | | | ✓ | |
| **DES 288.** OWASP IoT8: Mitigating Lack of Device Management | | ✓ | | | | | | | |
| **DES 289.** OWASP IoT9: Mitigating Insecure Default Settings | | ✓ | | | | | | | |
| **DES 290.** OWASP IoT10: Mitigating Lack of Physical Hardening | | ✓ | | | | | | | |
| **DES 361.** Mitigating LCNC (Low-Code/No-Code) Account Impersonation | | ✓ | ✓ | | | | | | |
| **DES 362.** Mitigating LCNC (Low-Code/No-Code)) Authorization Misuse | | ✓ | ✓ | | | | | | |
| **DES 364.** Mitigating LCNC Authentication and Secure Communication Failures | | ✓ | ✓ | | | | | | |
| **DES 283.** OWASP IoT3: Mitigating Insecure Ecosystem Interfaces | | ✓ | | | | | | | |
| **DES 284.** OWASP IoT4: Mitigating Lack of Secure Update Mechanism | | ✓ | | | | | | | |
| **DES 285.** OWASP IoT5: Mitigating Use of Insecure or Outdated Components | | ✓ | | | | | | | |
| **DES 286.** OWASP IoT6: Mitigating Insufficient Privacy Protection | | ✓ | | | | | | | |
| **DES 287.** OWASP IoT7: Mitigating Insecure Data Transfer and Storage | | ✓ | | | | | | ✓ | |
| **DES 288.** OWASP IoT8: Mitigating Lack of Device Management | | ✓ | | | | | | | |
| **DES 289.** OWASP IoT9: Mitigating Insecure Default Settings | | ✓ | | | | | | | |
| **DES 290.** OWASP IoT10: Mitigating Lack of Physical Hardening | | ✓ | | | | | | | |
| **DES 361.** Mitigating LCNC (Low-Code/No-Code) Account Impersonation | | ✓ | ✓ | | | | | | |
| **DES 362.** Mitigating LCNC (Low-Code/No-Code)) Authorization Misuse | | ✓ | ✓ | | | | | | |
| **DES 364.** Mitigating LCNC Authentication and Secure Communication Failures | | ✓ | ✓ | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURE DESIGN** | | | | | | | | | |
| **CYB 210.** Cybersecurity Incident Response | | | ✓ | | | | | | |
| **CYB 211.** Identifying and Protecting Assets Against Ransonmware | | | ✓ | | | | | | |
| **CYB 212.** Fundamentals of Security Information & Event Management (SIEM) | | | ✓ | | | | | | |
| **DES 101.** Fundamentals of Secure Architecture | | | ✓ | ✓ | ✓ | | | ✓ | |
| **DES 151.** Fundamentals of the PCI Secure SLC Standard | ✓ | | ✓ | ✓ | | | | | |
| **DES 200.** Fundamentals of Cryptography, Key Management and Digital Certificates | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 201.** Securing Applications and their Data with Cryptography | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 206.** Meeting Cloud Governance and Compliance Requirements | | | ✓ | | | | | | |
| **DES 209.** Authentication and Lifecycle Management | | | ✓ | | | | | | |
| **DES 255.** Securing the IoT Update Process | | ✓ | ✓ | | | | | | |
| **DES 262.** Securing Enterprise Low-Code Application Platforms | | | ✓ | | | | | | |
| **DES 305.** Blockchain Security - Protecting Existing Blockchain Assets | ✓ | ✓ | ✓ | ✓ | | | | ✓ | |
| **DES 311.** Creating Secure Application Architecture | | | ✓ | ✓ | | ✓ | | ✓ | |
| **DES 312.** Protecting Cardholder Data | | | | ✓ | | | | | |
| **DES 313.** Hardening a Kubernetes Cluster | | | ✓ | | | | | | |
| **ENG 191.** Introduction to the Microsoft SDL | | | ✓ | ✓ | ✓ | ✓ | | | |
| **ENG 192.** Implementing the MS SDL Optimization Model | | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **ENG 193.** Implementing the Agile MS SDL | | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **ENG 194.** Implementing MS SDL Line of Business | | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **ENG 195.** Implementing the MS SDL Threat Modeling Tool | | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| **ENG 205.** Fundamentals of Threat Modeling | | | | | | | | ✓ | |
| **ENG 211.** How to Create Application Security Design Requirements | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **ENG 212.** Implementing Secure Software Operatiions | ✓ | ✓ | ✓ | ✓ | | | | | |
| **ENG 251.** Risk Management Foundations | | | ✓ | | | | | | |
| **ENG 311.** Attack Surface Analysis and Reduction | | ✓ | | ✓ | | | | ✓ | |
| **ENG 312.** How to Perform a Security Code Review | ✓ | ✓ | ✓ | ✓ | | | | | |
| **ENG 320.** Using Software Composition Analysis to Secure Open-Source Components | ✓ | ✓ | ✓ | ✓ | | | | | |
| **ENG 351.** Preparing the Risk Management Framework | | | ✓ | | | | | | |
| **ENG 352.** Categorizing Systems and Information within the RMF | | | ✓ | ✓ | | | | ✓ | |
| **ENG 353.** Selecting, Implementing, and Assessing Controls within the RMF | | ✓ | ✓ | ✓ | | | | ✓ | |
| **ENG 354.** Authorizing and Monitoring System Controls within the RMF | | ✓ | ✓ | ✓ | | | | ✓ | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **INFRASTRUCTURE SECURITY** | | | | | | | | | |
| **API 250.** Controlling Access to the Kubernetes API | | ✓ | ✓ | | | | | | |
| **API 251.** Implementing Web Application and API Protection (WAAP) | | ✓ | ✓ | | | | | | |
| **CYB 251.** Securing the AI/ML Infrastructure | | | ✓ | | | | | | ✓ |
| **CYB 350.** AI Driven Incident Analysis Techniques | | ✓ | ✓ | | | | | | |
| **DES 210.** Hardening Linux/Unix Systems | ✓ | ✓ | ✓ | ✓ | | | | ✓ | ✓ |
| **DES 212.** Architecture Risk Analysis and Remediation | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 214.** Securing Infrastructure Architecture | | | ✓ | ✓ | | | | | |
| **DES 215.** Defending Infrastructure | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 216.** Protecting Cloud Infrastructure | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 217.** Securing Terraform Infrastructure and Resources | | | ✓ | | | | | | |
| **DES 218.** Protecting Microservices, Containers, and Orchestration | | | ✓ | | | | | ✓ | |
| **DES 219.** Securing Google's Firebase Platform | | | ✓ | | ✓ | | | | |
| **DES 260.** Fundamentals of IoT Architecture and Design | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **DES 261.** Securing Serverless Environments | | ✓ | ✓ | | | | | | |
| **DES 306.** Creating a Secure Blockchain Network | ✓ | ✓ | ✓ | ✓ | | | | ✓ | |
| **DES 314.** Hardening the Docker Engine | | | ✓ | | | | | | |
| **ICS 210.** ICS/SCADA Security Essentials | | | ✓ | | | | | | |
| **ICS 310.** Protecting Information and System Integrity in Industrial Control System Environments | | | ✓ | | | | | | |
| **DevSecOps** | | | | | | | | | |
| **CYB 213.** Generative AI Privacy & Cybersecurity Risk | | ✓ | ✓ | | | | | ✓ | |
| **CYB 310.** Using Cyber Supply Chain Risk Management to Mitigate Threats to IT/OT | | | ✓ | | | | | | |
| **CYB 311.** Threat Analysis with Artificial Intelligence | | | ✓ | | | | | | |
| **DSO 201.** Fundamentals of Secure DevOps | | | ✓ | ✓ | | | | | |
| **DSO 205.** Securing the COTS Supply Chain | ✓ | ✓ | ✓ | | | | | | |
| **DSO 206.** Securing the Open Source Software Supply Chain | | ✓ | ✓ | | | | | | |
| **DSO 211.** Identifying Threats to Containers and Data in a DevSecOps Framework | ✓ | ✓ | ✓ | ✓ | | | | | |
| **DSO 212.** Fundamentals of Zero Trust Security | | | ✓ | | | | | | |
| **DSO 253.** DevSecOps in the AWS Cloud | | ✓ | ✓ | | | | | ✓ | |
| **DSO 254.** DevSecOps in the Azure Cloud | | ✓ | ✓ | | | | | ✓ | |
| **DSO 256.** DevSecOps in the Google Cloud Platform | | ✓ | ✓ | | | | | ✓ | |
| **DSO 301.** Orchestrating Secure System and Service Configuration | | ✓ | ✓ | ✓ | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **DevSecOps *(Continued)*** | | | | | | | | | |
| **DSO 302.** Automated Security Testing | | | ✓ | ✓ | | | | | |
| **DSO 303.** Automating Security Updates | ✓ | | ✓ | ✓ | | | | | |
| **DSO 304.** Securing API Gateways in a DevSecOps Framework | ✓ | ✓ | ✓ | | | | | | |
| **DSO 305.** Automating CI/CD Pipeline Compliance | | ✓ | ✓ | | | | | ✓ | |
| **DSO 306.** Implementing Infrastructure as Code | | | ✓ | | | | | | |
| **DSO 307.** Secure Secrets Management | | | ✓ | ✓ | | | | | |
| **SECURITY TESTING** | | | | | | | | | |
| **ATK 201.** Fundamentals of Security Testing | | ✓ | ✓ | ✓ | | | | | ✓ |
| **CYB 250.** Cyber Threat Hunting: Tactics, Techniques, and Procedures (TTP) | | | ✓ | | | | | | ✓ |
| **CYB 301.** Fundamentals of Ethical Hacking | | | ✓ | ✓ | | | | | ✓ |
| **SDT 301.** Testing for Injection | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 302.** Testing for Identification and Authentication Failures | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 303.** Testing for Cryptographic Failures | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 304.** Testing for Insecure Design | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 305.** Testing for Broken Access Control | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 306.** Testing for Security Miscconfiguration | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 307.** Testing for Server-Side Request Forgery | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 308.** Testing for Software and Data Integrity Failures | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 309.** Testing for Vulnerable and Outdate Components | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 310.** Testing for Security Logging and Monitoring Failures | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| **SDT 311.** Testing for Integer Overflow or Wraparound | ✓ | ✓ | ✓ | ✓ | | | | | |
| **SDT 312.** Testing for Path Traversal | ✓ | | | | | | | | |
| **SDT 313.** Testing for Cross Site Request Forgery | ✓ | | | | | | | | |
| **SDT 314.** Testing for Unrestricted Upload of File with Dangerous Type | ✓ | ✓ | | | | | | | |
| **SDT 315.** Testing for Incorrect Permission Assignment for Critical Resource | ✓ | ✓ | | | | | | | |
| **SDT 316.** Testing for Use of Hard-Coded Credentials | ✓ | | | | | | | | |
| **SDT 317.** Testing for Improper Control of Generation of Code ("Code Injection") | ✓ | ✓ | | ✓ | | | | | |
| **SDT 318.** Testing for Insufficiently Protected Credentials | ✓ | ✓ | | ✓ | | | | | |
| **SDT 319.** Testing for Out-of-bound Read | ✓ | ✓ | | ✓ | | | | | |
| **SDT 320.** Testing for Out-of-bounds Write | ✓ | ✓ | | ✓ | | | | | |
| **SDT 321.** Testing for Uncontrolled Resource Consumption | ✓ | ✓ | | ✓ | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SECURITY TESTING** *(Continued)* | | | | | | | | | |
| **SDT 322.** Testing for Improper Privilege Management | ✓ | ✓ | | ✓ | | | | | |
| **SDT 323.** Testing for Improper Input Validation | ✓ | ✓ | | ✓ | | | | | |
| **SDT 324.** Testing for Improper Restriction of Operations within the Bounds of a Memory Buffer | ✓ | ✓ | | ✓ | | | | | |
| **SDT 325.** Testing for NULL Pointer Dereference | ✓ | ✓ | | ✓ | | | | | |
| **SDT 326.** Testing for Use After Free | ✓ | ✓ | | ✓ | | | | | |
| **TST 101.** Fundamentals of Security Testing | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ |
| **TST 202.** Penetration Testing Fundamentals | | | ✓ | ✓ | | | | | |
| **TST 205.** Performing Vulnerability Scans | ✓ | | ✓ | | | | | | |
| **TST 206.** ASVS Requirements for Developers | | ✓ | | ✓ | | | | | |
| **TST 301.** Infrastructure Penetration Testing | ✓ | | ✓ | ✓ | | | | ✓ | |
| **TST 302.** Application Penetration Testing | ✓ | | ✓ | ✓ | | | | ✓ | |
| **TST 303.** Penetration Testing for Google Cloud Platform | | | ✓ | | | | | | |
| **TST 304.** Penetration Testing for AWS Cloud | | | ✓ | | | | | | |
| **TST 305.** Penetration Testing for Azure Cloud | | | ✓ | | | | | | |
| **TST 351.** Penetration Testing for TLS Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **TST 352.** Penetration Testing for Injection Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **TST 353.** Penetration Testing for SQL Injection | | ✓ | | | | | | | |
| **TST 354.** Penetration Testing for Memory Corruption Vulnerabilities | ✓ | | ✓ | | | | | | |
| **TST 355.** Penetration Testing for Authorization Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **TST 356.** Penetration Testing for XSS | ✓ | ✓ | | | | | | | |
| **TST 357.** Penetration Testing for Hardcoded Secrets | ✓ | | ✓ | | | | | | |
| **TST 358.** Penetration Testing Wireless Networks | ✓ | | ✓ | | | | | | |
| **TST 359.** Penetration Testing Network Infrastructure | ✓ | | ✓ | | | | | | |
| **TST 360.** Penetration Testing for Authentication Vulnerabilities | ✓ | | ✓ | | | | | | |
| **LEARN LABS** | | | | | | | | | |
| **LAB 111.** Identifying Server-Side Request Forgery | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 113.** Identifying Cryptographic Failures | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 114.** Identifying Cookie Tampering | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 115.** Identifying Reflective Cross-Site Scripting (XSS) | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 116.** Identifying Forceful Browsing | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 117.** Identifying Hidden Form Field | ✓ | ✓ | ✓ | | | | | | ✓ |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **LEARN LABS** *(Continued)* | | | | | | | | | |
| **LAB 118.** Identifying Weak File Upload Validation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 119.** Identifying Persistent Cross-Site Scripting (XSS) | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 120.** Identifying XML Injection | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 121.** Identifying Vulnerable and Outdated Components | | ✓ | ✓ | | | | | | ✓ |
| **LAB 122.** Identifying Insecure APIs | | ✓ | ✓ | | | | | | ✓ |
| **LAB 123.** Identifying Vertical Privilege Escalation | | ✓ | ✓ | | | | | | ✓ |
| **LAB 124.** Identifying Horizontal Privilege Escalation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 125.** Identifying Buffer Overflow | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 126.** Identifying Information Leakage | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 127.** Identifying Security Logging and Monitoring Failures | ✓ | ✓ | | | | | | | |
| **LAB 128.** Identifying Unverified Password Change | ✓ | ✓ | | | | | | | |
| **LAB 129.** Identifying Error Message Containing Sensitive Information | ✓ | ✓ | | | | | | | |
| **LAB 130.** Identifying Generation of Predictable Numbers or Identifiers | ✓ | ✓ | | | | | | | |
| **LAB 131.** Identifying Improper Restriction of XML External Entity Reference | ✓ | ✓ | | | | | | | ✓ |
| **LAB 132.** Identifying Exposed Services | | | | | | | | | ✓ |
| **LAB 133.** Identifying Exposure of Sensitive Information Through Environmental Variables | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 134.** Identifying Plaintext Storage of a Password | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 135.** Identifying URL Redirection to Untrusted Site | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 136.** Identifying Improper Neutralization of Script in Attributes in a Web Page | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 137.** Identifying Improper Authorization | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 138.** Identifying Authorization Bypass Through User-Controlled Key | ✓ | ✓ | ✓ | | | | | | |
| **LAB 139.** Identifying Use of a Key Past its Expiration Date | ✓ | ✓ | ✓ | | | | | | ✓ |
| **SKILL LABS** | | | | | | | | | |
| **LAB 201.** Defending Java Applications Against Canonicalization | ✓ | | ✓ | | | | | | |
| **LAB 202.** Defending Python Applications Against Canonicalization | ✓ | | ✓ | | | | | | |
| **LAB 203.** Defending C# Applications Against Canonicalization | ✓ | | ✓ | | | | | | |
| **LAB 204.** Defending Node.js Applications Against Canonicalization | ✓ | | ✓ | | | | | | |
| **LAB 205.** Defending Java Applications Against XPath Injection | | ✓ | ✓ | | | | | | |
| **LAB 206.** Defending Python Applications Against XPath Injection | | ✓ | ✓ | | | | | | |
| **LAB 207.** Defending Node.js Applications Against XPath Injection | | ✓ | ✓ | | | | | | |
| **LAB 208.** Defending C# Applications Against XPath Injection | | ✓ | ✓ | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS** *(Continued)* | | | | | | | | | |
| **LAB 211.** Defending Java Applications Against Credentials in Code Medium | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 212.** Defending Python Applications Against Credentials in Code Medium | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 213.** Defending Node.js Applications Against Credentials in Code Medium | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 214.** Defending C# Applications Against Credentials in Code Medium | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 215.** Defending Java Applications Against Business Logic Error for Input Validation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 216.** Defending Python Applications Against Business Logic Error for Input Validation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 217.** Defending Node.js Applications Against Business Logic Error for Input Validation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 218.** Defending C# Applications Against Business Logic Error for Input Validation | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 220.** Defending Against Hard-Coded Secrets (HTML5) | ✓ | ✓ | | | | | | | |
| **LAB 221.** Defending C# Against SQL Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 224.** Defending Java Applications Against Forceful Browsing | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 225.** Defending Python Applications Against Forceful Browsing | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 226.** Defending Node.js Applications Against Forceful Browsing | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 227.** Defending C# Applications Against Forceful Browsing | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 222.** Defending Python Against SQL Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 223.** Defending Node.js Against SQL Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 228.** Defending Java Applications Against Weak AES ECB Mode Encryption | ✓ | ✓ | | | | | | | |
| **LAB 229.** Defending Java Applications Against Weak PRNG | ✓ | ✓ | | | | | | | |
| **LAB 230.** Defending Java Against Cross-Site Scripting (XSS) | ✓ | ✓ | | | | | | | |
| **LAB 231.** Defending Python Against Cross-Site Scripting (XSS) | ✓ | ✓ | | | | | | | |
| **LAB 232.** Defending C# Against Cross-Site Scripting (XSS) | ✓ | ✓ | | | | | | | |
| **LAB 233.** Defending Node.js Against Cross-Site Scripting (XSS) | ✓ | ✓ | | | | | | | |
| **LAB 234.** Defending Java Applications Against Parameter Tampering | ✓ | ✓ | ✓ | | | | | | |
| **LAB 235.** Defending Java Applications Against Plaintext Password Storage | ✓ | ✓ | ✓ | | | | | | |
| **LAB 236.** Defending Java Applications Against Sensitive Information in Error Messages | ✓ | ✓ | | | | | | | |
| **LAB 237.** Defending Java Against SQL Injection | ✓ | ✓ | | | | | | | |
| **LAB 238.** Defending C# Applications Against Weak AES ECB Mode Encryption | ✓ | ✓ | ✓ | | | | | | |
| **LAB 239.** Defending C# Applications Against Weak PRNG | ✓ | ✓ | ✓ | | | | | | |
| **LAB 240. Defending** Java Against ExternalXML Entity Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **LAB 241. Defending** C# Against ExternalXML Entity Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **LAB 242. Defending** Node.js Against ExternalXML Entity Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS** *(Continued)* | | | | | | | | | |
| **LAB 243. Defending** Python Against ExternalXML Entity Vulnerabilities | ✓ | ✓ | ✓ | | | | | | |
| **LAB 244. Defending** Java Against Security Misconfiguration | ✓ | ✓ | ✓ | | | | | | |
| **LAB 245.** Defending Node.js Applications Against Plaintext Password Storage | ✓ | ✓ | ✓ | | | | | | |
| **LAB 246.** Defending Node.js Applications Against Weak AES ECB Mode Encryption | ✓ | ✓ | ✓ | | | | | | |
| **LAB 247.** Defending Node.js Applications Against Weak PRNG | ✓ | ✓ | ✓ | | | | | | |
| **LAB 248.** Defending Node.js Applications Against Parameter Tampering | ✓ | ✓ | ✓ | | | | | | |
| **LAB 249.** Defending Python Applications Against Plaintext Password Storage | ✓ | ✓ | ✓ | | | | | | |
| **LAB 250.** Defending C# Applications Against Parameter Tampering | ✓ | ✓ | ✓ | | | | | | |
| **LAB 251.** Defending C# Applications Against Plaintext Password Storage | ✓ | ✓ | ✓ | | | | | | |
| **LAB 252.** Defending Python Applications Against Weak AES ECB Mode Encryption | ✓ | ✓ | ✓ | | | | | | |
| **LAB 253.** Defending Python Applications Against Weak PRNG | ✓ | ✓ | ✓ | | | | | | |
| **LAB 254.** Defending Python Applications Against Parameter Tampering | ✓ | ✓ | ✓ | | | | | | |
| **LAB 260.** Defending C# Applications Against Sensitive Information in Error Messages | ✓ | ✓ | | | | | | | |
| **LAB 261.** Defending Python Applications Against Sensitive Information in Error Messages | ✓ | ✓ | | | | | | | |
| **LAB 262.** Defending Node.js Applications Against Sensitive Information in Error Messages | ✓ | ✓ | | | | | | | |
| **LAB 263.** Defending Java Applications Against Sensitive Information in Log Files | ✓ | ✓ | | | | | | | |
| **LAB 264.** Defending Python Applications Against Sensitive Information in Log Files | ✓ | ✓ | | | | | | | |
| **LAB 265.** Defending Node.js Applications Against Sensitive Information in Log Files | ✓ | ✓ | | | | | | | |
| **LAB 266.** Defending C# Applications Against Sensitive Information in Log Files | ✓ | ✓ | | | | | | | |
| **LAB 267.** Defending Java Applications Against Deserialization of Untrusted Data | ✓ | ✓ | | | | | | | |
| **LAB 268.** Defending Python Applications Against Deserialization of Untrusted Data | ✓ | ✓ | | | | | | | |
| **LAB 269.** Defending Node.js Applications Against Deserialization of Untrusted Data | ✓ | ✓ | | | | | | | |
| **LAB 270.** Defending C# Applications Against Deserialization of Untrusted Data | ✓ | ✓ | | | | | | | |
| **LAB 271.** Defending Java Applications Against SSRF | ✓ | ✓ | | | | | | | |
| **LAB 272.** Defending Python Applications Against SSRF | ✓ | ✓ | | | | | | | |
| **LAB 273.** Defending Node.js Applications Against SSRF | ✓ | ✓ | | | | | | | |
| **LAB 274.** Defending C# Applications Against SSRF | ✓ | ✓ | | | | | | | |
| **LAB 275.** Defending Java Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 276.** Defending Python Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 277.** Defending Node.js Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 278.** Defending C# Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS** *(Continued)* | | | | | | | | | |
| **LAB 279.** Defending Java Applications Against Dangerous File Upload | ✓ | ✓ | ✓ | | | | | | |
| **LAB 280.** Defending Python Applications Against Dangerous File Upload | ✓ | ✓ | ✓ | | | | | | |
| **LAB 281.** Defending Node.js Against Dangerous File Upload | ✓ | ✓ | ✓ | | | | | | |
| **LAB 282.** Defending C# Applications Against Dangerous File Upload | ✓ | ✓ | ✓ | | | | | | |
| **LAB 283.** Defending Java Applications Against RegEx DoS | ✓ | ✓ | ✓ | | | | | | |
| **LAB 284.** Defending Python Applications Against RegEx DoS | ✓ | ✓ | ✓ | | | | | | |
| **LAB 285.** Defending Node.js Applications Against RegEx DoS | ✓ | ✓ | ✓ | | | | | | |
| **LAB 286.** Defending C# Applications Against RegEx DoS | ✓ | ✓ | ✓ | | | | | | |
| **LAB 287.** Defending Java Applications Against Null Pointer Dereference | ✓ | ✓ | ✓ | | | | | | |
| **LAB 288.** Defending C# Applications Against Null Pointer Dereference | ✓ | ✓ | ✓ | | | | | | |
| **LAB 289.** Defending Java Applications Against Path Traversal | ✓ | ✓ | ✓ | | | | | | |
| **LAB 290.** Defending Python Applications Against Path Traversal | ✓ | ✓ | ✓ | | | | | | |
| **LAB 291.** Defending Node.js Applications Against Path Traversal | ✓ | ✓ | ✓ | | | | | | |
| **LAB 292.** Defending C# Applications Against Path Traversal | ✓ | ✓ | ✓ | | | | | | |
| **LAB 293.** Defending Java Applications Against Integer Overflow | ✓ | ✓ | ✓ | | | | | | |
| **LAB 294.** Defending C# Applications Against Integer Overflow | ✓ | ✓ | ✓ | | | | | | |
| **LAB 301.** Defending Java Applications Against Open Redirect | ✓ | ✓ | | | | | | | ✓ |
| **LAB 302.** Defending Python Applications Against Open Redirect | ✓ | ✓ | | | | | | | ✓ |
| **LAB 303.** Defending C# Applications Against Open Redirect | ✓ | ✓ | | | | | | | ✓ |
| **LAB 304.** Defending Node.js Applications Against Open Redirect | ✓ | ✓ | | | | | | | ✓ |
| **LAB 305.** Defending Java Applications Against Weak Password Reset | ✓ | ✓ | | | | | | | ✓ |
| **LAB 306.** Defending Python Applications Against Weak Password Reset | ✓ | ✓ | | | | | | | ✓ |
| **LAB 307.** Defending C# Applications Against Weak Password Reset | ✓ | ✓ | | | | | | | ✓ |
| **LAB 308.** Defending Node.js Applications Against Weak Password Reset | ✓ | ✓ | | | | | | | ✓ |
| **LAB 309.** Defending TypeScript Applications Against Unrestricted Upload of File with Dangerous Type | ✓ | ✓ | | | | | | | ✓ |
| **LAB 314.** Defending TypeScript Applications Against SSRF | ✓ | ✓ | | | | | | | ✓ |
| **LAB 316.** Defending TypeScript Applications Against Hard-coded Credentials | ✓ | ✓ | | | | | | | ✓ |
| **LAB 320.** Defending TypeScript Applications Against Code Injection | ✓ | ✓ | | | | | | | ✓ |
| **LAB 325.** Defending TypeScript Applications Against CSRF | ✓ | ✓ | | | | | | | ✓ |
| **LAB 326.** Defending TypeScript Applications Against Path Traversal | ✓ | ✓ | | | | | | | ✓ |
| **LAB 327.** Defending C Applications Against Path Traversal | ✓ | ✓ | | | | | | | ✓ |
| **LAB 328.** Defending C++ Applications Against Path Traversal | ✓ | ✓ | | | | | | | ✓ |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS** *(Continued)* | | | | | | | | | |
| **LAB 329.** Defending Go Applications Against SSRF | ✓ | ✓ | ✓ | | | | | | |
| **LAB 333.** Defending Go Applications Against Hard-coded credentials | ✓ | ✓ | ✓ | | | | | | |
| **LAB 338.** Defending Go Applications Against CSRF | ✓ | ✓ | ✓ | | | | | | |
| **LAB 339.** Defending Go Applications Against Path Traversal | ✓ | ✓ | ✓ | | | | | | |
| **LAB 340.** Defending C Applications Against Use After Free | ✓ | ✓ | ✓ | | | | | | |
| **LAB 341.** Defending C ++ Applications Against Use After Free | ✓ | ✓ | ✓ | | | | | | |
| **LAB 342.** Defending TypeScript Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 343.** Defending GO Applications Against Command Injection | ✓ | ✓ | ✓ | | | | | | |
| **LAB 344.** Defending TypeScript Applications Against Incorrect Authorization. | ✓ | ✓ | ✓ | | | | | | |
| **LAB 345.** Defending GO Applications Against Incorrect Authorization. | ✓ | ✓ | ✓ | | | | | | |
| **LAB 346.** Defending TypeScript Applications Against Deserialization of Untrusted Data. | ✓ | ✓ | ✓ | | | | | | |
| **LAB 347.** Defending C Applications Against Null Pointer Dereference. | ✓ | ✓ | ✓ | | | | | | |
| **LAB 348.** Defending C++ Applications Against Null Pointer Dereference | ✓ | ✓ | | | | | | | ✓ |
| **LAB 349.** Defending TypeScript Applications Against SQL Injection | ✓ | ✓ | | | | | | | ✓ |
| **LAB 350.** Defending Go Applications Against SQL Injection | ✓ | ✓ | | | | | | | ✓ |
| **LAB 351.** Defending TypeScript Applications Against Cross-Site Scripting | ✓ | ✓ | | | | | | | ✓ |
| **LAB 352.** Defending Go Applications Against Cross-Site Scripting | ✓ | ✓ | | | | | | | ✓ |
| **LAB 353.** Defending TypeScript Applications Against Improper Authentication | ✓ | ✓ | | | | | | | ✓ |
| **LAB 354.** Defending Go Applications Against Improper Authentication | ✓ | ✓ | | | | | | | ✓ |
| **LAB 355.** Defending C Applications Against Stack-based Buffer Overflow | ✓ | | | | | | | | ✓ |
| **LAB 356.** Defending Python APIs from Broken Object Level Authorization | | ✓ | | | | | | | |
| **LAB 357.** Defending Python APIs from Broken Authentication | | ✓ | | | | | | | |
| **LAB 358.** Defending Python APIs from Broken Object Property Level Authorization | | ✓ | | | | | | | |
| **LAB 359.** Defending Python APIs from Unrestricted Resource Consumption | | ✓ | | | | | | | |
| **LAB 360.** Defending Python APIs from Broken Function Level Authorization | | ✓ | | | | | | | |
| **LAB 361.** Defending Python APIs from Unrestricted Access to Sensitive Business Flows | | ✓ | | | | | | | |
| **LAB 362.** Defending Python APIs from Server Side Request Forgery | | ✓ | | | | | | | |
| **LAB 363.** Defending Python APIs from Security Misconfiguration | | ✓ | | | | | | | |
| **LAB 364.** Defending Python APIs from Improper Inventory Management | | ✓ | | | | | | | |
| **LAB 365.** Defending Python APIs from Unsafe Consumption of APIs | | ✓ | | | | | | | |
| **LAB 366.** Defending Python AI Applications from Prompt Injection | | ✓ | | | | | | | |
| **LAB 367.** Defending Python AI Applications from Sensitive Information Disclosure | | ✓ | | | | | | | |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS *(Continued)*** | | | | | | | | | |
| **LAB 368.** Defending Python AI Applications from Supply Chain Compromise | | ✓ | | | | | | | |
| **LAB 369.** Defending Python AI Applications from Data and Model Poisoning | | ✓ | | | | | | | |
| **LAB 370.** Defending Python AI Applications from Improper Output Handling | | ✓ | | | | | | | |
| **LAB 371.** Defending Python AI Applications from Excessive Agency | | ✓ | | | | | | | |
| **LAB 372.** Defending Python AI Applications from System Prompt Leakage | | ✓ | | | | | | | |
| **LAB 373.** Defending Python AI Applications from Vector and Embedding Weaknesses | | ✓ | | | | | | | |
| **LAB 374.** Defending Python AI Applications from Misinformation | | ✓ | | | | | | | |
| **LAB 375.** Defending Python AI Applications from Unbounded Consumption | | ✓ | | | | | | | |
| **LAB 376.** Defending Node.js AI Applications from Prompt Injection | | ✓ | | | | | | | |
| **LAB 377.** Defending Node.js AI Applications from Improper Output Handling | | ✓ | | | | | | | |
| **LAB 378.** Defending Node.js AI Applications from System Prompt Leakage | | ✓ | | | | | | | |
| **LAB 379.** Defending Node.js AI Applications from Misinformation | | ✓ | | | | | | | |
| **LAB 380.** Defending Node.js AI Applications from Unbounded Consumption | | ✓ | | | | | | | |
| **LAB 610.** ATT&CK: File and Directory Permissions Modification | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 611.** ATT&CK: File and Directory Discovery | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 612.** ATT&CK: Testing for Network Services Identification | | | ✓ | | | | | | ✓ |
| **LAB 613.** ATT&CK: Testing for Vulnerability Identification Using Vulnerability Databases | | | ✓ | | | | | | ✓ |
| **LAB 615.** ATT&CK: Updating Vulnerable Java Web Application Server Software | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 616.** ATT&CK: Host Vulnerability Scanning | | | ✓ | | | | | | ✓ |
| **LAB 617.** ATT&CK: Testing for Plaintext Secrets in Files | | | ✓ | | | | | | ✓ |
| **LAB 618.** ATT&CK: Log Analysis | | | ✓ | | | | | | ✓ |
| **LAB 619.** ATT&CK: Exfiltration Over C2 Channel | | | ✓ | | | | | | ✓ |
| **LAB 620.** ATT&CK: Exploitation of Remote Services (Advanced) | | | ✓ | | | | | | ✓ |
| **LAB 621.** ATT&CK: Password Cracking | ✓ | ✓ | | | | | | | ✓ |
| **LAB 622.** ATT&CK: Exploiting Windows File Sharing Server with External Remote Services | | ✓ | | | | | | | ✓ |
| **LAB 623.** ATT&CK: Exploiting Vulnerable Java Web Application Server Software | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 625.** ATT&CK: Exploit Public-Facing Application (Advanced) | | | ✓ | | | | | | ✓ |
| **LAB 624.** ATT&CK: Exploiting Java Web Application Server Misconfiguration | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 626.** Using an Exploit Framework for SQL Injection | ✓ | ✓ | ✓ | | | | | | ✓ |
| **LAB 627.** Using an Exploit Framework for Port Scanning. | | | ✓ | | | | | | ✓ |
| **LAB 628.** Using an Exploit Framework for SMB Version Scanning. | | | ✓ | | | | | | ✓ |

| COURSE TITLE | CWE | OWASP | NIST* | PCI | ISO | NERC | HIPAA | GDPR | MITRE |
|---|---|---|---|---|---|---|---|---|---|
| **SKILL LABS *(Continued)*** | | | | | | | | | |
| **LAB 629.** Using an Exploit Framework for SNMP Scanning. | | | ✓ | | | | | | ✓ |
| **LAB 630.** ATT&CK: Exploiting Java SQL Injection to Extract Password Hashes | ✓ | ✓ | | | | | | | ✓ |
| **LAB 631.** ATT&CK: Network Service Discovery | ✓ | ✓ | | | | | | | ✓ |
| **LAB 632.** ATT&CK: Network Share Discovery | ✓ | ✓ | | | | | | | ✓ |
| **LAB 633.** Using an Exploit Framework for Web Application Scanning | | | ✓ | | | | | | ✓ |
| **LAB 634.** ATT&CK: Create Account | ✓ | ✓ | | | | | | | ✓ |
| **LAB 635.** ATT&CK: Unsecured Credentials | ✓ | ✓ | | | | | | | ✓ |
| **LAB 636.** ATT&CK: Data from Local System | | | | | | | | | ✓ |
| **LAB 637.** ATT&CK: Valid Accounts | | | | | | | | | ✓ |
| **LAB 638.** Using Mimikatz | | | ✓ | | | | | | ✓ |
| **LAB 639.** Using an Exploit Framework via Command Line Interface | | | ✓ | | | | | | ✓ |
| **LAB 640.** ATT&CK: Search Victim-Owned Websites | | | | | | | | | ✓ |
| **LAB 641.** ATT&CK: Password Policy Discovery | | | | | | | | | ✓ |
| **LAB 642.** ATT&CK: Permission Groups Discovery | | | | | | | | | ✓ |
| **LAB 643.** Response: Detecting a Malicious Windows Service | | | | | | | | | ✓ |
| **LAB 644.** Response: Detecting Malware in the Windows Startup Folder | | | | | | | | | ✓ |
| **LAB 645.** Response: Detecting Malware in the Registry Run Keys | | | | | | | | | ✓ |
| **LAB 646.** Response: Detecting a JSP Backdoor | | | | | | | | | ✓ |
| **LAB 647.** Response: Investigating an Antimalware Alert | | | | | | | | | ✓ |
| **LAB 648.** Response: Detecting a Malicious Windows Scheduled Task | | | | | | | | | ✓ |

*Our NIST courses that map to 800-53 and 800-171 publications. To understand how courses map to specific requirements, please contact us.

## ABOUT CMD+CTRL SECURITY

CMD+CTRL Security is a pioneer in software security training. For over two decades, organizations of all sizes, from mid-sized to Global 100 companies, have relied on our training solutions to transform their software security. Our role-based modules, skill labs, and hands-on cyber ranges are designed to build skills that stick. Visit cmdnctrlsecurity.com to learn how we can help you launch a best-in-class training program.

cmd+ctrl