General disclaimer

This document presents details about the training offerings from Synopsys at the time of its creation. Synopsys has used reasonable efforts to ensure that the information provided in this document is accurate and up-to-date, but details and offerings are subject to change.

This document contains confidential information about Synopsys and its businesses. Copies of this document may only be provided, and disclosure of the information contained in it may only be made, with written prior agreement from Synopsys.

Ownership and disposal

The information contained in this document is owned by Synopsys. The recipient shall dispose of the data as confidential waste and/or return the document to Synopsys upon request.

The Synopsys difference

Synopsys helps development teams build secure, high-quality software, minimizing risks while maximizing speed and productivity. Synopsys, a recognized leader in application security, provides static analysis, software composition analysis, and dynamic analysis solutions that enable teams to quickly find and fix vulnerabilities and defects in proprietary code, open source components, and application behavior.

For more information about the Synopsys Software Integrity Group, visit us online at www.synopsys.com/software.

Synopsys, Inc.
185 Berry Street, Suite 6500
San Francisco, CA 94107 USA

U.S. Sales: 800.873.8193
International Sales: +1 415.321.5237
Email: sig-info@synopsys.com
Table of contents

**Introduction** .................................................. 4
**Our curriculum** ................................................. 4

**Fundamentals** .................................................. 5
- Principles of Software Security ................................ 6
- Attack and Defense .............................................. 7
- OWASP Top 10 .................................................. 8

**Authentication and Authorization** ................. 9
- The Securing APIs Using OAuth 2.0 ...................... 10

**Mobile** ......................................................... 11
- Defending Android .............................................. 12
- Defending iOS .................................................. 13
- Securing Mobile Platforms 101 ........................... 14
- Securing Mobile Platforms 201 ........................... 15

**Cloud Platforms** .............................................. 16
- Securing Azure .................................................. 17
- Securing AWS .................................................. 18
- Securing Containers With Docker ......................... 19

**Defensive Strategies** ......................................... 20
- Securing Software With DevSecOps ....................... 21

**Languages and Platforms** ................................ 22
- Defending COBOL ............................................... 23
- Defending C# ASP.NET ......................................... 24
- Defending C/C++ ................................................ 25
- Defending GoLang .............................................. 26
- Defending HTML ................................................ 27
- Defending Java EE ............................................... 28
- Defending Java SE .............................................. 29
- Defending JavaScript (No Framework) .................... 30
- Defending JavaScript Angular ............................... 31
- Defending JavaScript AngularJS ............................ 32
- Defending JavaScript React ................................... 33
- Defending Python With Django ............................. 34
- Defending Python With Flask ............................... 35
- Defending .NET Core .......................................... 36

**Attacking Strategies** ......................................... 37
- Attacking Code Using Static Analysis .................... 38
- Attacking Networks ........................................... 39
- Attacking Web Applications ................................ 40
- Red Teaming .................................................... 41

**Requirements, Architecture, and Training** .. 42
- Threat Modeling Microservices ............................. 43
- Threat Modeling Multitier ................................... 44

**Embedded and IoT** ........................................... 45
- Embedded Systems Security ................................. 46

**Webinars** ....................................................... 47

**Product Training** ............................................. 50
- Coverity Server Administration Essentials .............. 51
- Coverity Build Administration Essentials ............... 52
- Coverity Essentials for End Users ......................... 53
- Defensics Essentials ........................................... 54
- Black Duck Essentials ......................................... 55
Introduction
Synopsys broad range of software security products and professional services affords us the unique position to create, maintain, and deliver the best software security training for our customers. Our instructional design process puts practicing consultants in charge of courses in their respective area of expertise. Course owner’s use their experience solving customers’ challenges to inform course direction. Similarly, we use certified practicing consultants as instructors. Instructors are able to share with the students real-life examples from previous customer interactions.

What this means to you, our courses aren’t just text-book best practices, courses have experience baked in from design through delivery.

Synopsys is also the creator and leader of the BSIMM (https://www.bsimm.com) and BSIMM community. Insights from the BSIMM, as well as from BSIMM assessors, influence both course and catalog direction. Feedback from the community is swiftly incorporated to ensure our courses are relevant for development and security teams.

Our curriculum

• **Fundamentals**: Your software security journey starts here. Fundamentals courses are designed to get you started.
• **Authentication and Authorization**: Securely using modern federated authentication and authorization frameworks frees your application team to focus on functionality while shifting some risk to third parties. Learn security best practices for using frameworks such as OAuth and SAML.
• **Cloud Platforms**: Just because your application makes use of cloud providers for hosting doesn't automatically mean the application is deployed securely. The settings and options for deployment can be daunting for developers and operations new to the cloud environment. Securely configuring the deployment is vital to the security of your customers and data.
• **Defensive Strategies**: Software development and deployment is happening at a blistering pace. To ensure software is not being sent out the door with security defects processes must be put in place to ensure the software is tested thoroughly as the software winds its way through the development process. Courses in defensive strategies ensure that the latest best practices are being deployed in your environment.
• **Attacking Strategies**: Understanding how adversaries look for weaknesses in our software is key to building security in. These courses are designed to help you put on your proverbial malicious hat.
• **Languages & Platforms**: Knowing the weaknesses in your chosen language or platform is the only way to avoid those weaknesses that lead to security vulnerabilities. Languages & Platforms highlight those weaknesses then show you how to avoid them using industry best practices.
• **Mobile**: Your application will be downloaded and installed on thousands of devices, are you sure you’ve implemented the correct security features? Explore what features should be enabled and when to secure your mobile applications.
• **Requirements, Architecture, & Training**: The earliest stages of the SDLC are requirements, architecture, and training. Courses in this category are designed to help you catch security problems early, when they are easiest and cheapest to fix.
• **Webinar**: Topics in this section are delivered to a wider audience. Duration is about one hour in length.
• **Product Training**: Learn how to use Synopsys products with a live instructor guiding your way.
The Securing Web Services course is intended for developers, engineers, and architects who work with backend web services APIs which may not necessarily have a User Interface (UI) or a UI component. This course examines web services concepts and then takes a deep dive into several web services technologies such as WS-Security, Security Assertion Markup Language (SAML), and OAuth. This course also covers risks inherent to web services and how to properly threat model web services. Web service security is examined from the perspective of the message, the channel, and the service itself. The lab component of this course allows students to gain an understanding of and practice with some of the real-world security issues inherent to web services.

The lab is intentionally written with a programming language and framework that are popular but with which most developers are not familiar: Python and Flask. This allows students to focus on secure design and secure coding concepts without being too concerned with the implementation details of a particular language.

This is a comprehensive and stand-alone course on web services. Many concepts taught in this course are covered in depth in other courses, such as Threat Modeling and OAuth. If you are building a multi-day curriculum for web service developers, please reach out to the Synopsys training team for advice on course selection.
The Principles of Software Security course provides the foundation to inspire developers and other team members to start taking security seriously. This modular course can be delivered as a full-day offering, or depending on your needs, can be focused on one of the primary modules as a half-day course.

**Introduction module**
The half-day Introduction module first identifies current software security problems, and then addresses the issues by explaining how to infuse software security into the development process early on. This module elucidates the Synopsys concept of “Building Security In” as opposed to relying solely on traditional security and testing practices.

- Basic software security concepts: Topics include a software security vocabulary, obstacles to software security, how to build security in, and the importance of a software security initiative (SSI)
- Fundamentals of a SSI: SSI scope, goals, engineering and guidance, vendor management, software security groups (SSGs), strategy, training, compliance, and metrics
- Software security engineering: Three pillars of risk management, touchpoints, and knowledge, security standards, and training, and how to integrate this learning with your Waterfall or Agile development approach

**Labs**
- Security hurdles in an ever-connected world of malicious actors
  - Think like an attacker by considering data, network, and functionality of the device
- Identify the best defect discovery techniques
  - Scenario: Starting down the road
  - Scenario: We can do more!
  - Scenario: Building security in

**Requirements module**
The Requirements half-day module focuses on introducing important cost-saving software security requirements early in the software development life cycle. Students learn the details of and the causes behind secure coding errors and mistakes in this data-centric module, and how these software security defects are exploited. They will also learn the practices that help prevent the most common mistakes.

- Essential use cases: Access control requirements for authentication and authorization
- Resource management: Ways to protect resources and prevent attacks such as denial of service and resource management guidelines
- Data life cycle: Data protection at every stage of data interpretation (data in use, data at rest, and data in motion), as well as data input, processing, and output; improper input validation; input validation approaches and guidance; log injection; output encoding; safe error handling; protecting the cache; masking sensitive data; and encryption

**Lab**
- Security requirements for use cases:
  - Authentication
  - Authorization
  - Resource management
  - Data interpretation
  - Data in use
  - Data in motion
  - Data at rest
**Attack and Defense**

The Attack and Defense course provides software builders and testers an in-depth look at standard attacks and their corresponding defenses. Students successfully completing this course are empowered to solve tricky problems securely in their own environment by mapping them to known problems and tried-and-tested solutions.

This course introduces common attacks that can happen to most applications. These attacks are also seen in different contexts such as web, embedded, thick client, or mobile, and their standard solutions are discussed in the classroom. Students are then guided to apply this knowledge to identify attacks and design defenses for a model application throughout the labs.

**Protecting data**
This section examines the life cycle stages of data, identifies common attacks for each stage, and explains how to handle common use cases securely.

- Data at rest (online and offline attacks): Exploring common ways of storing data, and the associated attacks targeted to reveal otherwise inaccessible information
- Data in motion: Exploring common communication implementations between components, and attacks that can be performed to eavesdrop on, replay, or modify data
- Data interpretation: Exploring the difference between control and data planes, how they require different approaches of interpretation, and examples of resulting attacks and countermeasures
- Data in use: Exploring attacks on the underlying software, hardware stacks, and physical world environments that can leak data

**Access control**
This section discusses authentication and authorization, and looks at common methods of identifying a system user and ways of hijacking that identity. It also examines the controls used to split and combine permissions to achieve business goals while following the principle of least privilege. And it includes a discussion about the importance of keeping audit logs.

**Resource management**
This section highlights the importance of software performance considerations in the context of intentional misuse and abuse. How much stress can a malicious user put on the system? Does that user always require a rich pool of resources to do so?

**Open source software**
Risks from open source software are discussed in this section, including:

- Open source software use
- Common attacks
- Standard defenses
- Common pitfalls

**Labs**

- Password hash cracking: Students run a password hacking program called John the Ripper (JtR).
- SSL scan: Students use a free tool called Qualys SSL Scan to test the security strength of an SSL certificate used to encrypt communication of a website.
- Intercept HTTP request/response: Students examine one of the most important tools of web security testing: the local HTTP proxy.
- Session ID entropy: Students look at the entropy of the session ID in the Bank of Insecurities
The OWASP Top 10 course focuses on the most important security defects found in web applications, including all issues in the latest OWASP Top 10 list. Each issue includes a description of the vulnerability as well as practical guidance for remediation. This course also provides demonstrations and practical hands-on exercises examining what impact these security issues can have on web applications.

- **A1:2017 Injection:** The dangers of mixing data with code
- **A2:2017 Broken Authentication:** Broken authentication and session management
- **A3:2017 Sensitive Data Exposure:** Resulting from improper classification or lack of authorization
- **A4:2017 XML External Entities (XXE):** Specific type of server-side request forgery (SSRF) attack
- **A5:2017 Broken Access Control:** Authentication vs. authorization, privilege escalation, tampering
- **A6:2017 Security Misconfiguration:** Misconfigured servers, lack of knowledge of installed features
- **A7:2017 Cross-Site Scripting (XSS):** Resulting from unencoded, unvalidated, and untrusted user-supplied data
- **A8:2017 Insecure Deserialization:** Causes of deserialized vulnerabilities
- **A9:2017 Using Components with Known Vulnerabilities:** Why and how does this happen?
- **A10:2017 Insufficient Logging and Monitoring:** Secure logging and monitoring

**References and next steps**
- BSIMM: Measure your development practices
- Secure software development life cycle
- Coding standards
- Security unit tests
- Security integration tests
- Automated security analysis

**Labs**
Challenges and exercises:
- **Getting started**
  - Hands-on with labs and tools, challenges, and learning
  - Accessing Skytap
  - Starting and navigating OWASP Juice Shop
  - Learning Burp Suite Community Edition
- **Injection**
  - SQL injection
  - NoSQL injection
- **Broken access control**
  - Force-browse to the administration section of the website
  - Access someone else's basket
- **XSS**
  - Reflected DOM XSS
  - Stored XSS without using the user interface
Authentication and Authorization
The Securing APIs Using OAuth 2.0

The Securing APIs Using OAuth 2.0 course teaches students how to control access to APIs using OAuth tokens. The course explains each OAuth grant type, the intended use case, and common attacks. Architectural design flaws as well as common implementation bugs related to OAuth authorization servers, resource servers, clients, and tokens are all discussed in this course. The course also highlights the dangers of using plain OAuth for authentication and explains the need for technologies like OpenID/Connect. The content covered in the course is reinforced with instructor-led demos and hands-on group lab exercises.

API risk landscape
This section defines APIs, their security capabilities according to Gartner, and threats according to OWASP. It includes multiple scenarios in which organizations have failed to secure APIs and explains how to secure access to services.

The need for OAuth 2.0
This section describes the need for OAuth 2.0 in the interconnected world of applications and APIs, provides a little historical perspective, and discusses problems present before OAuth 2.0.

Delegated access with OAuth 2.0
This section includes various example scenarios and conceptual views of how OAuth 2.0 delegates access to applications on the user’s behalf.

Overview of OAuth 2.0 grant types
This section provides a high-level overview of four distinct grant types or “flows” that determine access to clients and avoid potential security issues.

Delegated access from a web application: Authorization code grant
Delegated access from a single-page application: Implicit grant
Delegated access from a back-end application: Client credentials
- Resource owner password grant
Delegated access from a mobile application: Proof key for code exchange

Extensions: JWT and OpenID Connect
This section describes the following OAuth 2.0 extensions:
- JSON web tokens (JWT) as access tokens
- OpenID Connect to perform authentication

Phishing for OAuth tokens
This section explains how hackers phish for OAuth tokens. It also includes pointers on how to defend your system against such attacks.

Lab
This lab reinforces what was learned:
- Students work in independent groups to build an entire threat model for a fictitious system with a component diagram
- Even with a defined process, students come up with and discuss different threat models
Mobile
Defending Android

The Defending Android course begins with foundational overview of the Android platform, its architecture, and the security model, and then builds on that to discuss Android-specific risks. The core of the course is defensive programming techniques for the Android platform and common application security risks. Defensive programming techniques are explored within the framework of the OWASP Mobile Top 10 Security Project. The risk itself, code that implements the risk, and time-tested techniques for addressing the risks are all examined.

The Android platform, architecture, and security model
Students learn about the Android operating system, application runtime, application components, kernel-level security, application-level security, and the Trusted Execution Environment.

The OWASP Mobile Security Project
This section teaches students about the software risks in the Android platform based on the OWASP Mobile Security Project.

Defensive programming techniques for Android
For each risk examined in this section, students will be able to recognize affected code, understand how to remediate the risk, and make the changes in the code to mitigate the risk. There are several knowledge-check quizzes in this section. Application risks specific to the Android platform include permissions, intents, activities, broadcast receivers, content providers, services, logging, web views, and file handling.

Common application security risks
The risks reviewed in this section are common application risks that can be applicable not only to the Android platform but to applications outside of it as well. Examples, knowledge checks, and a lab follow this section. Topics include: improper authentication, client-side injection, SQL injection, cross-site scripting (XSS), lack of server-side controls, denial of service, and eavesdropping.

Labs
Each lab is followed by a question and answer session.
• Attacks and defense on a malicious application
  – Exploiting the content provider with remediation
  – Exploiting the broadcast receiver with remediation
  – Exploiting the exported activity with remediation
• Android platform-specific defensive programming addressing M2 and M5
  – Extract sensitive information stored by the application
  – Identify information leakage risks in the logs
  – Discover risks for surrounding data and credential storage
• Application security defensive programming for M6, M8, M9, and M10
  – Reverse-engineer the insecure application, tamper with the authorization mechanism, and compile the application again
  – Discover different client-side injection risks
Defending iOS

The Defending iOS course begins with an overview of the iOS platform, the securities that are built in, and how they have evolved over the many iterations of the iOS operating system. Next, the course walks through common iOS application security concerns and discusses how best to mitigate or remediate such issues. The course looks at the risk, the code that implements the issues, and code examples for the issues and remediation steps.

The iOS platform, architecture, and security model

- **Architecture**
  - Describe the iOS operating system: XNU kernel, architecture
  - Describe the iOS application runtime: Development components
  - Describe the layers of the iOS SDK
  - Identify the components of iOS applications: Application package ("ipa" file), application types, components, storage, and interprocess communication
  - Describe iOS storage and IPC choices

- **Security controls**
  - Describe OS-level security: UNIX security, sandboxing, FairPlay DRM, code signing, keychain services, and touch ID
  - Describe application-level security: Address space layout randomization, nonexecutable data pages, stack canaries, iOS SDK built-in protections, privacy controls, and local authentication framework

- **Risk landscape**
  - Common iOS application security risks
  - Reverse engineering: Tools, and defenses
  - Jailbreaking: Anti-jailbreaking controls
  - Attacks against touch ID

Defensive programming for common iOS application risks

- Insecure handling of URL schemes
- Insecure network communication
- Data leakage
- Weak authentication/authorization
- Weak cryptography
- Buffer overflows
- Improper input validation and data representation

Labs

- Data leak risks
  - Information leakage—logging exercise
  - Credentials stored in plist
  - Credentials stored in SQLite database

- Crypto
  - Hard coded keys
  - Keychain

Note: Students are required to have a Mac.
Mobile Securing Mobile Platforms 101

The Securing Mobile Platforms 101 course is aimed at users who are new to mobile, new to mobile security, or have had only brief exposure to such topics. It covers areas including common mobile vulnerabilities and mitigating controls, different app architecture types, and more.

This course contains three modules.

Module 1: Mobile-first AppSec
As we continue to see new platforms, cross-platform frameworks, and architectures, mobile-first AppSec techniques need to continue to evolve to keep up. This module takes a new look at applying our existing AppSec knowledge and processes to this new mobile landscape.

This module includes discussions on these topics:
- Introduction to mobile-first development
- Mobile application architectures, threat models, and AppSec considerations
- Testing and tools, security assessment, and trends

Module 2: Android and iOS platform security overview
This module explores Android and iOS architectures and ecosystems, how applications are installed, how they exist on devices, and what features exist on the platforms. This module includes discussions on the following topics:
- Various application and platform security controls that exist on both platforms to enable safer development and runtime environment
- Application origin, components, features, and inter-app communication methods
- Application package structure for both Android and iOS platforms
- Implementation concerns regarding some components and frameworks

Module 3: Introduction to secure development for Android and iOS
This module examines common security risks in iOS and Android applications, with the objectives of both recognizing bad code and writing secure code. Topics include:
- iOS: Secure development overview of common and platform-specific risks with knowledge checks and code examples
- Android: Secure development overview of common and platform-specific risks with knowledge checks and code examples

Intended Audience
- QA Engineers
- Security Practitioners
- Architects
- Mobile Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course objectives
At the end of this course, you will be able to:
- Understand how various parts of the Android and iOS platforms operate including application security controls
- Identify various language and system frameworks available on Android and iOS platforms
- Understand the layout of application packages for both platforms
- Recognize how applications and extensions can communicate at runtime
- Recognize mobile application security issues for both platforms
- Understand defensive programming techniques
The Securing Mobile Platforms 201 course is aimed at users with some understanding of mobile platforms who want to gain an understanding of advanced security concepts. It places particular emphasis on applications that are highly sensitive or are considered business-critical. This course also delves into tampering of mobile apps, jailbreaking, rooting, the use of debuggers, and more.

This course contains three modules.

**Module 1: Mobile payments**
This module contains the following topics:
- Introduction and the mobile payments ecosystem: Mobile banking, P2P, POS terminals, and proximity payments
- Mobile payments architecture: Major system components, communication, API, personalization, and feature comparison of known systems
- Publicly known vulnerabilities and takeaways from these: Forensic security analysis, PIN storage and verification, relay attacks, and more
- Additional attacks and countermeasures

**Module 2: Protecting client-side code**
This module contains the following topics:
- Common attacks: Examines attackers’ goals, their approach, and the techniques they may use while targeting mobile applications
- Protection techniques and countermeasures: Describes in detail various protection techniques such as anti-debugging and asset encryption
- Implementation concerns: Looks at certain caveats and things you should have in mind while implementing protection in your application
- Technologies: Reviews some of the most popular protection suites in the market

**Module 3: Protecting business-critical applications**
This module contains the following topics:
- What is a business-critical app? Includes typical behaviors and attributes that help us determine whether an application is business-critical, and recaps the attacker’s typical process
- Attacks against business-critical apps: Provides an overview of modern tampering techniques and detections
Cloud Platforms
Securing Azure

In the Securing Azure course, students learn how to secure Azure infrastructure as a service (IaaS). This course initially presents a brief overview of the Azure infrastructure offerings, and then dives into how to secure them. In hands-on labs, students learn how to spot an insecure Azure configuration and fix it.

Introduction to Azure cloud and security considerations
This section covers:
• Cloud service, deployment, and shared responsibility models
• Risks, security capabilities and security considerations

Identity and access management
This section describes how identity and access management are implemented in Azure. Some topics include security considerations, general best practices, Azure AD hardening, privileged identity, management, monitoring, and policy.

Networking
This section discusses how to implement and secure virtual networking resources. Some topics include: security considerations, isolation of resources, protection of data in transit, access control, rules, virtual network service endpoints, firewalls, and monitoring.

Storage
This section explores how a storage account in Azure enables access to its storage solutions. Some topics include security considerations, encryption for data at rest and in transit, data plane security, management plane security, availability, logging and monitoring, and key vault.

Compute services
This section examines compute offerings and security considerations within Azure. Some topics include security considerations, VM access management, application identity, disk encryption, policies for virtual machine, image management, monitoring, and Azure virtual machine security.

Labs
• Initial login: Students set up their lab environment in this lab.
• Identity and access management: This exercise explores how the services running within the virtual machine can execute actions on Azure's management plane. Students also identify the role and permissions assigned to the virtual machine and understand its security implications.
• Networking: In this exercise, students examine a network security misconfiguration and fix it.
• Storage: This lab allows students to explore how Azure storage can be accessed using methods such as storage account keys and shared access signature, and learn the security implications of each method.
Securing AWS

Cloud computing has grabbed the world’s attention not only for its pervasive, on-demand, convenient usage, but for its ability to be vulnerable to data breaches and novel forms of attack. Since most software uses the cloud in various shared capacities (development, hosting, or integration with third-party code), threats from hackers are inevitable. This hands-on workshop equips students to understand this new landscape of converged infrastructure and shared services, its existing and emerging threats, and provides them with secure mitigation methods.

The Securing AWS course is an introductory course, covering Amazon core services, such as IAM, KMS, EC2, S3, and VPCs, with a focus on security. This course enables students to identify areas for cross-pollination between development and operations that enhance application, infrastructure, and network security.

Introduction and threat model
This section discusses cloud risks, Juice Shop lab architecture, threat modeling introduction, and a Juice Shop threat model.

Identity and access management
This section covers a quick start and common security considerations for identity and access management (IAM), including root account security and general IAM best practices.

Virtual private cloud
This section discusses multiple levels of security that you can use to protect your network.

Elastic compute cloud
This section covers a quick start and common security considerations for elastic compute cloud (EC2).

Key management service
This section covers a quick start and common security considerations for key management service (KMS).

Relational database service
This section covers a quick start and common security considerations for relational database service (RDS).

Simple storage service security
This section covers a quick start and common security considerations for simple storage service security (S3).

Labs
- Initial login: Students set up their lab environment in this lab
- Exploring IAM roles: Students explore the functionality of IAM roles, and peer under the hood to understand some of their security implications
- Encrypting data with KMS: Students use a customer-managed customer master key to encrypt and decrypt sensitive data hosted on the workstation server
- RDS network hardening: Students identify and correct a security RDS misconfiguration
- Securing S3 buckets: Students access S3 objects, configure permission, and try to secure them
Securing Containers With Docker

Containers have changed the way applications are being deployed. "Containerization" has gained traction over the years because it easily enables an application team to build, package, and distribute a microservice or an application across different environments. Docker has emerged as the leading container technology for packaging and deploying these services or applications.

However, as always, security is a challenge that organizations face when it comes to deploying containers securely. Container security refers to protecting the integrity of the containers—the application as well as the infrastructure it uses. The Securing Containers With Docker course features hands-on labs, best practices, and instructions that will enable students to harden the container runtime and the container host. The sections of this course are the following:

- Introduction: Overview, function, and value
- A closer look at Docker: Setting up your own Docker images
- Security controls provided by Docker: Safeguards to containers
- Attacks against your containers: Common and uncommon attacks
- Open source tools: Streamlining the security of your Docker containers
- Understanding the risks: Technical risks and procedural challenges

Labs
The three labs in this course cover the following:

- Docker basic commands
- Run a remote image
- Working with images
- Run a local image
- Security controls
- User namespaces
- No new privileges
- Control groups
- Capabilities
- Apparmor
- Seccomp
- Hacklab
- Sharing is caring?
- Fixing the vulnerability
- The great escape!
- Fixing the hole
- Volumes of vulnerabilities
- Fixing the vulnerability
- Hard mode!

Intended Audience
- DevSecOps
- Cloud Architects
- Developers
- Security Practitioners

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course objectives
At the end of this course, you will be able to:

- Understand how containers work and differ from virtualization
- Understand the key risks when using containers
- List the available security controls
- Learn how to protect against container attacks
- Understand Docker security best practices
- Learn about some common container security tools
Securing Software With DevSecOps

Securing Software With DevSecOps is an introductory course aimed at understanding DevSecOps key concepts, roles, benefits, challenges, and deployment. Differences between Agile, continuous integration/continuous delivery (CI/CD), and DevSecOps are explored in this course. DevSecOps sample pipeline demos and case studies enrich the course to make it a complete learning experience for students.

DevOps
• This section refreshes students’ DevOps knowledge and lays the foundation for the DevSecOps section. Along with a short history of DevOps, this section describes the mindset and various entities that need to collaborate to achieve the common goal of company excellence and competitive advantage. It also elaborates on CI and CD, and how they compare against Agile and DevOps.
• DevSecOps
• This deep-dive section first defines DevSecOps and explains its benefits, key concepts, and culture. Then it details DevSecOps challenges in three categories (people, process, and technology) as a precursor to the next problem-solving section.
• Achieving DevSecOps
• This section explores a three-pronged approach to achieving the best business results using DevSecOps.
• Case studies and DevSecOps challenges
• This section integrates all the stages of building a complete DevSecOps pipeline. The case studies are presented in reference to the three-pronged approach described above, illustrating how DevSecOps and automation helped clients achieve DevSecOps transformation.
• The challenges (people, process, technology) are discussed here as well, along with possible solutions that were applied.

Culture of automation and CI/CD
• This section focuses on how CI/CD and automation are integral to DevSecOps, enabling processes and people to be brought together via technology. This section also focuses on pipeline implementation in the context of several DevSecOps goals that organizations want to achieve.

Lab
The Java pipeline or Go pipeline lab exercises showcase basic automation use cases and tool integration. This course covers a sample language, either Java or Go, and a set of tools for the pipeline. These activities use cloud-based virtual machines. Lab activities include:
• Building the source code to generate a WAR (for Java pipeline) or GoLang (for Go pipeline) artifact
• Integrating a SAST scan (both pipelines)
• Integrating an SCA scan (both pipelines)
• Integrating a DAST scan (both pipelines)
• Pausing the CI pipeline for manual approval (both pipelines)
• Building an application image and integrating its scan (Java pipeline only)
Languages and Platforms
Defending COBOL

Software in any development and execution environment is subject to intrusion. This is certainly true of the Common Business-Oriented Language (COBOL) mainframe environment. The Defending COBOL course has two main sections. The first section looks at security risks to your company and how your code might contribute to that risk. The second section is the core of this course; it explores COBOL programming best practices and how to avoid or mitigate the vulnerabilities described in the taxonomy of vulnerabilities. It includes examples of bad code snippets, mitigation for each vulnerability, and secure coding examples to avoid the vulnerability.

**Section 1: Foundations of COBOL security**
- COBOL ecosystem: Application processing and architecture
- Modernization of COBOL and the associated risk in a changing environment
- Typical COBOL system assets
- COBOL security myths
- Cost of (in)security

**Section 2: COBOL defensive programming techniques**
- Secure input validation and data representation
- Improper input validation and output encoding
- Standard attacks on data interpretation
- Approaches to input validation
- Different types of injection attacks
- Secure and bad code examples
- Secure database access
- Authentication
- Authorization
- Secure logging practices
- Insecure and secure logging practices
- Secure error handling
- Leaking sensitive information
- Failing to clean up after failure
- Security principles and requirements
- Authentication using RACF and SAML models
- Authorization
- Confidentiality
- Cryptography

**Languages and Platforms**

**Intended Audience**
- Architects
- Developers

**Delivery Format**
- Virtual Classroom

**Course Duration**
- 4 hours

**Course objectives**
At the end of this course, you will be able to:
- Comprehend the cost of insecurity
- Understand COBOL security myths
- Comprehend defensive programming techniques
- Understand security principles
Defending C# ASP.NET

The Defending C# ASP.NET course explores in detail the root cause and remediation of vulnerabilities that apply to ASP.NET C# web applications. It also discusses activities that can be performed during the software development life cycle (SDLC) that help detect and prevent vulnerabilities, such as penetration testing and code review.

Course sections
- Overview of web application vulnerabilities: Introduces critical web application vulnerabilities in .NET
- Input validation: Examines its use as the first line of defense against injection attacks and other attacks
- Handling output: Describes why output encoding is used in addition to input validation or when input validation is not possible
- Using SQL safely: Investigates SQL injection and the approaches to creating SQL statements
- Authentication: Explains the flaws in the "security through obscurity" concept and .NET authentication systems, as well as the authentication methods, benefits, and drawbacks
- Session management: Shows what an attacker can do with a session token including session fixation, prediction, and brute-forcing, and what you can do about it
- Access control: Demonstrates the importance of avoiding excessive client-side trust and why access control is more than just authentication
- Defending against CSRF: Examines what is targeted by an attacker and provides common solutions
- Protecting the view state: Looks at how to preserve page and control values between round trips
- Deserialization: Describes what the C#.NET deserialization risks are and are not, and the various mitigation strategies
- Secure configuration: Explains the importance of recognizing and dealing with flaws in the system configuration and environment

Lab
This course has one lab, which incorporates the most important best practices discussed in the course.
- Performing input validation
- Using output encoding
- Using parameterized queries and stored procedures
- Protecting against CSRF
- Securing .NET serialization

Languages and Platforms

Intended Audience
- Architects
- Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course objectives
At the end of this course, you will be able to:
- Recognize web application vulnerabilities
- Understand OWASP Top 10 security risks
- Recognize flaws in the configuration and environment
- Identify tools available to handle risks
- Write secure code
Defending C/C++

The Defending C/C++ course provides developers with a strong foundation in software security as it relates to the implementation of applications developed in C or C++. This course includes detailed examples and focuses on the correct way to think through security problems by providing structured theory, demonstrations, technical deep-dives, and illustrated explanations. This course emphasizes the habit of building security in with proven programming practices, and explains common security-related problems in detail so that students can avoid them in their own work.

Note: The course can be delivered either as a C flavor or C++ flavor, but not both. Students can pick sections of relevance to them for a 1-day or a 2-day class. The 2-day class includes all course sections and labs.

Course sections
- Risk landscape
- Runtime and environment
- The C++ object model (only in the C++ flavor)
- Control plane versus the data plane
- Files, streams, and file I/O
- Stack corruption
- Memory integrity
- Sequences, algorithms, and containers (only in the C++ flavor)
- Integers
- Strings (only in the C flavor)
- Heap corruption and integrity
- Secure toolchain
- C++11, C++14, C++17 core language features (only in the C++ flavor)
- Patterns and antipatterns

Labs
- Getting to know your environment: Introduces the lab layout and the toolset provided on the virtual machines including editors, compiling, debugging, and scripting
- Risk landscape: Asks students to predict the outcome of a build, and run a simple program that gets sizing wrong
- Output encoding: Teaches the correct way to perform output encoding using the standard facilities
- Trusting input: Asks students to find, exploit, and fix a vulnerability on the server side of a vulnerable client-server program
- Stack-based buffer overflow and memory integrity: Introduces a sample application that is vulnerable to a buffer overflow
- (C++) preventing overflows using an input iterator adapter: Tasks students with writing an iterator adapter that can be used with the standard algorithms (e.g., std::copy) to prevent buffer overflows
- Integer attacks: Asks students to find integer vulnerabilities in four vulnerable programs and then fix the defects
- (C) strings attack and defense: Asks students to find and fix the vulnerabilities discussed in the course
- Writing a safe wrapper for calloc(): Asks students to write a safe wrapper for calloc() that contains the best practice in this course
- Compiler protections in-depth: Explores FORTIFY_SOURCE semantics and the stack-protector for glibc/GCC
- Bypassing ASLR: Teaches students why they should not rely on ASLR by stepping sequentially though an ASLR bypass

Intended Audience
- Architects
- Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 or 16 hours

Course objectives
At the end of this course, you will be able to:
- Identify security risks common to C/C++ applications
- Identify the impact to an application when a vulnerability is exploited
- Understand how to apply best practice C/C++ programming techniques
- Understand how best practices prevent common vulnerabilities
- Identify how C/C++ applications build configuration
- Identify how the production runtime environment can be used to further reduce risk

Languages and Platforms
Defending GoLang

The Defending GoLang course focuses on developing secure systems software and web applications in GoLang. It covers the generic risks in highly performant, concurrent systems software, the specific risks for GoLang developers, and the specific remedies available. This course also explains risks in developing web applications and web services in GoLang. Using real vulnerabilities from production open source projects, this course demonstrates the real-world complexity of security risks and remediation.

Defending GoLang is divided into two sections: Secure systems (back end) and Secure web development (front end). The topics from these sections are referenced with labs that allow students to discover those weaknesses in an intentionally vulnerable system as well as apply appropriate mitigations.

Secure systems (back end)
This section explores common weaknesses from an attacker’s perspective and then describes the best practices to keep those weaknesses from becoming vulnerabilities.
• Injection
• Process initialization safety
• Buffer overread
• Path traversal
• Concurrency flaws
• Unsafe library calls
• Integers
• Error handling

Labs
• Anonymous connections
• Directory traversal
• Command injection
• File validation
• Patching dangerous OS calls, authentication, directory traversal, command injection, and dangerous OS calls

Secure web development (front end)
This section focuses on the most important topics from the OWASP Top 10 as they apply to GoLang applications.
• A3:2017 sensitive data exposure
• A4:2017 XML external entities (XXE)
• A7:2017 Cross-site scripting (XSS)
• A8:2017 Insecure deserialization
• A9: 2017 Using components with known vulnerabilities

Labs
• SQL injection
• DOM-based XSS
• Stored XSS
• Insecure configuration
• XXE
• Patch SQL injection, DOM-based XSS, stored XSS, insecure configuration, and XXE

Intended Audience
• Architects
• Developers

Delivery Format
• Traditional Classroom
• Virtual Classroom

Class Duration
• 8 hours

Course objectives
At the end of this course, you will be able to:
• Identify security risks common to GoLang applications
• Identify the impact to the application when a vulnerability is exploited
• Understand how to apply best practices to GoLang programming

Languages and Platforms
Defending HTML

As HTML5 gains relevance in enterprises for its rich features and enhanced online experience, developers need to consider the vulnerabilities that their new code may introduce into their organizations’ web infrastructure. The Defending HTML5 course addresses this concern by identifying risks in the new HTML5 landscape and explaining how to mitigate them with defensive programming strategies and best practices.

The HTML5 risk landscape
Topics discussed in the first half of this section include:
• Web application attack surface
• Bugs and flaws
• Underlying protocol for browser communication (HTTP)
• Browser security architecture
• Traditional and mobile browser security architecture
Topics discussed in the second half of this section include:
• Same origin policy issues
• Document object model
• Common web security issues

Defensive programming techniques for HTML5
This section covers best practices that will help students write secure code when using these new HTML5 features:
• Web storage
• Web messaging
• Iframes sandbox
• Content security policy
• Cross-origin resource sharing policy
This section also encourages students to think carefully about what protection strategy to use when rendering user-submitted HTML markup.

Labs
This course includes five labs to reinforce learning. Students are requested to find, fix, and verify the vulnerability in the demo applications during these labs.
• Storing data securely: The application in this lab, a prototype of an online radio station, has a storage-related information disclosure vulnerability for students to fix.
• Cross-domain communication: In this lab, students work with a new technology: cross-domain messaging using the web messaging API. As in the previous lab, students modify the original code to make it withstand cross-domain attacks.
• Iframes in a sandbox (XSS): This lab helps students understand the correct usage of the iframe sandbox attribute. This attribute was introduced in HTML5 to provide extra security restrictions on iframes that are loaded from the parent domain, as well as from a different domain.
• Using content security policy: This lab helps students understand the correct usage of content security policy (CSP) headers. This header was introduced in HTML5 to provide extra security restrictions on the domains in which resources are loaded to reduce the impact of cross-site scripting.
• Using cross-origin resource sharing securely: In this lab, students use cross-origin resource sharing on an HTML5 demo ecommerce site to securely share user content with another website.
Defending Java EE

The Defending Java EE course focuses on using defensive programming techniques in Java EE applications against common web vulnerabilities. It discusses an approach to identify security risks and vulnerabilities, apply defensive programming techniques, and securely configure web applications. This course also provides demonstrations and practical hands-on exercises where students learn how to identify security vulnerabilities in the code and fix them using best practices discussed in the course.

Recognizing risks in enterprise Java web applications
This section discusses common web application risks, the typical Java EE risks, and the risks caused by the configuration of the application server. This section also provides a quick overview of OWASP Top 10 critical security risks.

Input validation
Details the risks of improper input validation and output encoding.

HTML output encoding
Discusses encoding contexts within a browser, and output encoding usage in HTML, a URL, JavaScript strings, Spring, Struts, and JSF.

Authentication
Discusses common broken authentication issues with reference to session management.

Secure session management
Describes secure session management techniques, secure Session ID generation, and secure timeouts.

Enforcing access control
Describes privilege escalation, forceful browsing attacks, and parameter tampering are some of the topics.

Preventing cross-site request forgery (CSRF)
Discusses reasons why CSRF is possible, and common approaches for prevention versus secure approaches.

Using databases
Explores SQL injection and includes examples of vulnerable code and details of how to avoid insecure API usage.

Handling XML
Optional section discusses common XML pitfalls with associated mitigation, and shows code vulnerable to XML, XXE, or XPath injection.

Java deserialization
Includes a brief history, types of deserialization, and usage, methods, and examples, with key mitigation techniques.

Miscellaneous topics
Covers general risks, miscellaneous security configurations, and error handling.

Lab
In the practical hands-on lab, students identify vulnerabilities in the code by investigating the application functionality and fixing vulnerabilities using secure coding best practices.

• Semantic input validation
• Performing input validation
• Using output encoding
• Using parameterized queries
Defending Java SE

The Java Standard Edition platform comes with a comprehensive feature set that allows developers to create and deploy Java applications on desktops, servers, and embedded devices. The Defending Java SE course examines security aspects of the desktop and client-server software architectures and aims to eliminate improper trust assumptions that developers commonly make. The core of this course teaches secure coding techniques designed to mitigate security vulnerabilities that affect software in general, as well as issues specific to the Java platform.

Risk landscape
A programming language risk landscape describes threat landscape faced by a particular programming language. In other words, it identifies the features of the language that merits attention. There will be some overlap with other languages, but it is also the specific aspects of the Java language that could be misused or abused that this course will cover.

Theory
In this section we lay the foundation for understanding injection attacks and some basic principles.

Handling XML
In this section, students learn how to protect their code from XML attacks by recognizing vulnerable code, secure configuration and threat mitigation.

Java deserialization
This section introduces vulnerabilities related to deserialization and strategies for mitigating them.

Secure file access
In this section, students learn to prevent attacks on files and file systems.

Java cryptographic architecture
The Java cryptographic architecture (JCA) provides a set of cryptographic services. This section includes an overview of "bad" cryptography, an in-depth discussion of SecureRandom, and other details on key generation and storage.

Java secure sockets extension
This section covers the use of TLS and HTTPS in Java programs.

Labs
- Defending against XML attacks: Students execute and defend against XEE and XEE.
- Defending against deserialization attacks: Students execute a deserialization attack using yoserial and then explore different mitigation techniques.
- Secure file access: This lab demonstrates an escalation of privilege by executing a path injection attack. Students must first find the vulnerability and then fix the issue.
- Certificate pinning: Students learn a safe way to perform certificate pinning.
Defending JavaScript
(No Framework)

The Defending JavaScript (No Framework) course addresses the questions of secure development in front-end and back-end JavaScript. This course helps students understand generic web application risks as well as specific risks involved in manipulating JavaScript in the Document Object Model (DOM) on the client side, bypassing browser controls such as same origin policy and sandboxing, sending AJAX requests, and using client-side frameworks and libraries. This course also explains risks present in server-side code written in JavaScript, and includes different types of injections, framework misconfigurations, cross-site request forgery, and input validation.

Note: Node.js is the server-side framework used with all JavaScript course flavors. Express.js is optional and is based on the customer’s requirement.

Web application risk landscape
Analyzes browser security features and web application risks.

JavaScript risk landscape
Introduces features and risks relevant to JavaScript applications such as manipulating the DOM and JavaScript execution contexts.

Defensive programming: Client-side HTML5 risks
Covers best practices that will help students write secure code when using the new HTML5 features of web storage, web messaging, Iframes sandbox, content security policy (CSP), and the best protection strategy to use when rendering user-submitted HTML markup.

Defensive programming techniques for HTML5: CORS
Offers a deep dive into cross-origin resource sharing (CORS).

Defensive programming techniques: Other HTML5 technologies
Discusses security risks arising from other HTML5 technologies such as geolocation functionality and the WebSocket API, and how best to defend against them.

Writing secure JavaScript
Introduces generic notions of defensive programming that relate to JavaScript applications.

Defensive programming with server-side JavaScript frameworks: Node.js
This section provides a brief overview of Node.js framework, as well as coding best practices on how to avoid security defects in Node.js applications:

- Input validation and output encoding
- Regular expression DoS (ReDoS)
- Unsafe deserialization
- Disabling certificate checks

Labs
Students find, fix, and verify the vulnerability in the demo applications during labs.

- Storing data securely: Find a storage-related information disclosure vulnerability and fix
- Cross-domain communication: Modify the original code to make it withstand cross-domain attacks
- Iframes in a sandbox (XSS): Correct usage of the iframe sandbox attribute
- Using content security policy: Correct usage of content security policy headers
- Using cross-origin resource sharing securely: Use cross-origin resource sharing on an HTML5 demo ecommerce site to securely share user content with another website
Defending JavaScript Angular

The course addresses the questions of secure development in front-end and back-end JavaScript. This course helps students understand generic web application risks as well as specific risks involved in manipulating JavaScript in the Document Object Model (DOM) on the client side, bypassing browser controls such as same origin policy and sandboxing, sending AJAX requests, and using client-side frameworks and libraries. This course also explains risks present in server-side code written in JavaScript, and includes different types of injections, framework misconfigurations, cross-site request forgery, and input validation.

Note: Node.js is the server-side framework used with all JavaScript course flavors. Express.js is optional, and is based on the customer’s requirement.

Web application risk landscape
Analyzes browser security features and web application risks.

JavaScript risk landscape
Introduces features and risks relevant to JavaScript applications such as manipulating the DOM and JavaScript execution contexts.

Defensive programming with client-side JavaScript
Introduces client-side JavaScript risks and known issues.

• Automatic output encoding and sanitizing
• Cross-site scripting vulnerabilities unique to Angular
• Leveraging Angular’s CSRF protection
• Client-side concerns such as browser storage, content security policy, and third-party components

Writing secure JavaScript
Introduces generic notions of defensive programming that relate to JavaScript applications.

Defensive programming with server-side JavaScript frameworks: Common issues
Discusses common server-side framework issues such as handling JSON data securely on server, vulnerable code and writing secure code, exploits, prototype pollution and mass assignment with examples.

Defensive programming with server-side JavaScript frameworks: Node.js
This section provides coding best practices on how to avoid security defects in Node.js applications:
• Process spawn with untrusted data
• Input validation and output encoding
• Secure logging practices

Labs
• Securing the client side: Students work on an application written in Angular and Node.js that has security defects, find the vulnerabilities, modify code, and verify fixes
• Securing the server side: Students work on the server-side vulnerabilities in the same sample application, exploit defects, and fix them
Defending JavaScript AngularJS

The Defending JavaScript AngularJS course addresses the questions of secure development in front-end and back-end JavaScript. This course helps students understand generic web application risks as well as specific risks involved in manipulating JavaScript in the Document Object Model (DOM) on the client side, bypassing browser controls such as same origin policy and sandboxing, sending AJAX requests, and using client-side frameworks and libraries. This course also explains risks present in server-side code written in JavaScript, and includes different types of injections, framework misconfigurations, cross-site request forgery, and input validation.

Note: Node.js is the server-side framework used with all JavaScript course flavors. Express.js is optional, and is based on the customer’s requirement.

Web application risk landscape
Analyzes browser security features and web application risks.

JavaScript risk landscape
Introduces features and risks relevant to JavaScript applications such as manipulating the DOM and JavaScript execution contexts.

Defensive programming with client-side JavaScript
Introduces client-side JavaScript risks and known issues.

Defensive programming with client-side JavaScript frameworks: AngularJS
This section briefly discusses the AngularJS framework, its security issues, and secure coding best practices:
• Automatic output encoding and sanitizing HTML content
• Strict contextual escaping (SCE)
• Cross-site scripting via template injections
• Angular and content security policy support
• Storing sensitive data
• AngularJS’s CSRF protection
• Sanitizing translation content

Writing secure JavaScript
Introduces generic notions of defensive programming that relate to JavaScript applications.

Defensive programming with server-side JavaScript frameworks: Common issues
Discusses common server-side framework issues such as handling JSON data securely on server, vulnerable code and writing secure code, exploits, prototype pollution and mass assignment with examples.

Defensive programming with server-side JavaScript frameworks: Node.js
This section provides coding best practices on how to avoid security defects in Node.js applications:
• Process spawn with untrusted data
• Input validation and output encoding
• Secure logging practices

Labs
• Securing the client side: Students work on an application written in AngularJS and Node.js that has security defects, find the vulnerabilities, modify code, and verify fixes
• Securing the server side: Students work on the server-side vulnerabilities in the same sample application, exploit defects, and fix them
Defending JavaScript React

The Defending JavaScript React course addresses the questions of secure development in front-end and back-end JavaScript. This course helps students understand generic web application risks as well as specific risks involved in manipulating JavaScript in the Document Object Model (DOM) on the client side, bypassing browser controls such as same origin policy and sandboxing, sending AJAX requests, and using client-side frameworks and libraries. This course also explains risks present in server-side code written in JavaScript, and includes different types of injections, framework misconfigurations, cross-site request forgery, and input validation.

Note: Node.js is the server-side framework used with all JavaScript course flavors. Express.js is optional, and is based on the customer’s requirement.

Web application risk landscape
Analyzes browser security features and web application risks.

JavaScript risk landscape
Introduces features and risks relevant to JavaScript applications such as manipulating the DOM and JavaScript execution contexts.

Defensive programming with client-side JavaScript
Introduces client-side JavaScript risks and known issues.

Defensive programming with client-side JavaScript frameworks: React
This section discusses the React framework, its security issues, and secure coding best practices:
• React security properties
• Contextual encoding
• Handling RAW HTML
• Steam remote code execution
• Client-side XSS sanitizers
• Preventing XSS when using server-side rendering
• Vulnerable third-party components and outdated React versions

Writing secure JavaScript
Introduces generic notions of defensive programming that relate to JavaScript applications.

Defensive programming with server-side JavaScript frameworks: Common issues
Discusses common server-side framework issues such as handling JSON data securely on server, vulnerable code and writing secure code, exploits, prototype pollution and mass assignment with examples.

Defensive programming with server-side JavaScript frameworks: Node.js
This section provides coding best practices on how to avoid security defects in Node.js applications:
• js overview
• Process spawn with untrusted data
• Input validation and output encoding
• Secure logging practices

Labs
• Securing the client side: Students work on an application written in React, MongoDB, and Node.js that has security defects, find the vulnerabilities, modify code, and verify fixes
• Securing the server side: Students work on the server-side vulnerabilities in the same sample application, exploit defects, and fix them
Defending Python With Django

Python is an extensively used general-purpose, high-level programming language designed to support rapid development, prototyping, and fast iterative development. The Defending Python With Django course covers common security vulnerabilities in both Python as well as in common Python frameworks.

Django is a popular Python web framework that leverages a model, view, template architecture. Django offers developers prebuilt functionality within the core components or as extendable third-party plugins.

Python overview and popularity
• Python background
• Python 2 or Python 3?
• Well-known websites using Python

Frameworks
• Well-known open source Python frameworks and selection criteria
• Similarities and differences between Django and Flask

Defending Django
• Overview, architecture, and project layout
• Security features in Django core
• Django plugins
• Configuring Django securely
• Exploiting and mitigating vulnerabilities

Common vulnerabilities
• Cross-site scripting (XSS)
• Broken authentication
• Cross-site request forgery (CSRF)
• Clickjacking
• Injection attacks
• Dangerous Python libraries
• Path manipulation
• Unsafe deserialization
• Information exposure
• Security misconfiguration
• Dependency management
• Regular expression denial of service (ReDoS)

Labs
Students exploit and remediate problem situations provided in the following lab exercises:
• Authentication: Students prevent unauthenticated access to the application functionality, and view, search, create and delete tasks
• Authorization: Students prevent unauthorized access to the application functionality, and view, search, and create tasks
• CSRF: This exercise exploits a cross-site request forgery vulnerability within the application to attempt to forge state change on the application server
• XSS: This exercise demonstrates a cross-site scripting exploit in the application in which a user can inject malicious client-side code (such as JavaScript) into the application
• SQLi: The exercise exploits a SQLi vulnerability in which the application processes untrusted user input in the context of a dynamic SQL query
• Unsafe deserialization: The exercise exploits an unsafe deserialization vulnerability in which the application deserializes an untrusted cookie value resulting in remote code execution
Defending Python With Flask

Python is an extensively used general-purpose, high-level programming language designed to support rapid development, prototyping, and fast iterative development. The Defending Python With Flask course covers common security vulnerabilities in both Python as well as in common Python frameworks.

Flask is a popular Python web framework that leverages a model, view, and template architecture. Flask offers developers prebuilt functionality within the core components or as extendable third-party plugins.

**Python overview and popularity**
Python background, Python 2 or Python 3? And Wellknown websites using Python.

**Frameworks**
Well-known open source Python frameworks and selection criteria and similarities and differences between Django and Flask.

**Defending Flask**
Secure configuration and exploitation and mitigation of vulnerabilities Common vulnerabilities:
• Cross-site scripting (XSS)
• Broken authentication
• Cross-site request forgery (CSRF)
• Clickjacking
• Injection attacks
• Unsafe deserialization
• Regular expression denial of service (ReDoS)

**Labs**
Students exploit and remediate problem situations provided in the following lab exercises:
• XSS: This exercise demonstrates cross-site scripting exploits in the application in which a user can inject malicious client-side code (such as JavaScript) into the application, and shows how to fix it
• Open URL redirect: This exercise demonstrates how an untrusted input can redirect the user to a different application, and shows how to fix it
• Authentication: This exercise accesses post-authentication pages without authenticating to the application, and shows how to fix it
• CSRF: This exercise exploits a cross-site request forgery vulnerability within the application to attempt to forge state change on the application server
• Clickjacking: The exercise focuses on finding vulnerable clickjacking instances and shows how to fix them
• Server-side template injection (SSTI): The exercise exploits the various implications of template injection
• File upload: This exercise focuses on fixing the file upload functionality in the application
• Path traversal: The goal of this exercise is to prevent path traversal
• Python library vulnerability: This exercise identifies the dangerous Python libraries used, and then shows how to fix them
• Instructor demonstration of unsafe deserialization RCE: The exercise exploits an unsafe deserialization vulnerability in which the application deserializes an untrusted cookie value resulting in remote code execution
• Flask configuration: This exercise reviews the Flask configuration file and then shows how to fix it

**Intended Audience**
- Architects
- Developers

**Delivery Format**
- Traditional Classroom
- Virtual Classroom

**Class Duration**
- 8 hours

**Course objectives**
At the end of this course, you will be able to:
• Recognize how common OWASP vulnerabilities affect Python applications
• Write secure Python code
• Securely configure popular Python frameworks
• Manage Python dependencies

**Languages and Platforms**
Defending .NET Core

The Defending .NET Core course focuses on modern .NET core secure development with an emphasis on microservices, service-oriented architecture, and cloud-first applications.

In addition, this course teaches modern attacker techniques and how to defensively write code to prevent these vulnerabilities in your applications. This course discusses activities that you can perform during the software development life cycle (SDLC) to detect and prevent vulnerabilities.

Overview of web application vulnerabilities
Introduces the critical web application vulnerabilities in .NET.

Secure design patterns
Covers design patterns for security: principle of least privilege, defense-in-depth, and more.

Test-driven development for security
Explores how to write unit tests to assert for security.

Improving code quality for security by leveraging OSS
Demonstrates how to leverage open source tools to improve the security posture of the SDLC.

Security in .NET core
This section includes a detailed discussion of web application vulnerabilities and how to defend against them. Topics include:
• Input validation: Examine its use as the first line of defense against injection attacks and other attacks
• Handling output: Learn why output encoding is used in addition to input validation or when input validation is not possible
• Using SQL safely: Investigate SQL injection and the approaches to creating SQL statements
• Authentication: Learn the flaws in the "security through obscurity" concept and .NET authentication systems as well as the authentication methods, their benefits, and drawbacks
• Securing JWT and sessions: See what an attacker can do with a session token, including session fixation, prediction, and brute-forcing, and what you can do about it, as well as how JWTs are handled securely
• Access control: See the importance of avoiding excessive client-side trust and why access control is more than just authentication
• Defending against CSRF: Examine what is targeted by an attacker and the common solutions
• Deserialization: Learn what the C#.NET deserialization risks are and are not, and the various mitigation strategies
• Secure configuration: Learn the importance of recognizing and dealing with flaws in the system configuration and environment

Lab
Lab exercises focus on the most important best practices discussed in the course:
• Performing input validation
• Using output encoding
• Using parameterized queries and stored procedures
• Protecting against CSRF
• Securing NET serialization

Languages and Platforms

Intended Audience
• Architects
• Developers

Delivery Format
• Traditional Classroom
• Virtual Classroom

Class Duration
• 8 hours

Course objectives
At the end of this course, you will be able to:
• Understand secure data processing controls including input validation and output encoding
• Examine how to prevent injection vulnerabilities
• Examine how to securely manage cookies and JWTs
• Understand test-driven security
• Understand environment hardening
Attacking Strategies
Attacking Code Using Static Analysis

The Using Static Analysis course focuses on the static analysis process and tools that can be used to test and attack a web application. It explains static analysis techniques, compares manual and automated code reviews, and discusses the implementation of static analysis in your software development life cycle (SDLC). This course also provides demonstrations and practical hands-on exercises in which students learn how to identify common vulnerabilities using code review and how to use common static analysis tools.

Introduction to static analysis
This section introduces static analysis, the need for it, its history, and types of static analysis.

Advantages and limitations of static analysis
Discusses static analysis pros and cons, false positives and false negatives, languages, frameworks, and third-party code.

Where does static analysis fit in?
Defines vulnerabilities and discusses where static application security testing (SAST) fits in.

Important static analysis concepts
Explains input validation and output encoding, proper use of APIs, technologies, and methods.

Static analysis types
This section details the pros and cons of each type of static analysis.

Static analysis common steps
Covers topics including code review cycle, establishing goals, understanding context, and source code and configuration.

Manual static analysis
Explains how to conduct manual static analysis, along with its advantages and pitfalls.

Tools and tool types
This section discusses tool types and available tools.

Deployment types
Covers topics such as centralized static analysis effort and considerations, developer desktops and considerations, build servers and considerations, and CI/CD pipeline and considerations.

Running tools
This section discusses tool flow, and simple and in-depth static analysis tools.

Triage
Explains how to triage findings, tackle a large number of findings, and understand impacts to triage.

Reporting
Discusses how to Report results and defects.

Fix the code
This section explains how to use findings and fix code.

Labs
The following labs are included in the course:
- Manual code review
- Desktop static analysis
- Configure and scan JavaSec using Coverity
- Results triage
Attacking Networks

The Attacking Networks course is aimed at helping students understand the security posture of a network and how best to uncover its vulnerabilities. The first part of this course introduces students to network security testing and then discusses a structured approach for performing tests using tools. The second part of the course is dedicated to software exploits, advanced testing skills, and post-exploitation activities. The final part of the course explains how to document and communicate findings from an assessment. Labs are performed throughout the course to tie concepts to the real world.

Introduction to network security testing
This section explains what network security testing is and how it differs from other testing types. Topics covered include:

- Network basics: Fundamentals of networking covering how networks work on the protocol layer
- Network security devices: Traditional and more advanced devices and the layers they operate at
- Rules for network security testing: Guidelines for not causing disruptions during testing

Network security testing process
This section details a structured approach to network security testing to ensure that all five steps are covered in the limited time frame available for the test.

Exploitation and post-exploitation
Exploits are some of the most common network security issues. Vulnerabilities in code allow attackers to compromise systems. This section gives an overview of various software exploits and how they are used in the fourth step of the network security testing process. Post-exploitation activities are a variety of techniques carried out after initial compromise. Advanced techniques used to gain additional access inside the network as well as to access sensitive information are also detailed in this section.

Communicating findings
The real value of a network security test comes when the findings are communicated in a clear and effective way to responsible entities for proper mitigation and correction measures to be taken. Being able to write a defect report that targets the right group is therefore one of the most important skills for a network security tester. This section explains the dos and don'ts of this valuable fifth step in the process.

Labs
Labs are chosen from this list to match audience needs:

- Wireshark: Observe an OSI model in action
- Nmap: Discover hosts and listening services
- Metasploit: Introduction to MSF and exploitation
- Netcat: Network swiss-army knife
- Password cracking: Going from hash to plaintext
- Communicating findings: Evaluate risks, document defects, and communicate to management
- Additional compromises: Find other vulnerabilities
Attacking Web Applications

The Attacking Web Applications course explains how to test for security issues in web applications. It defines what web security testing is and how it differs from other forms of testing, describes what the testing process looks like, and gives specific guidance on how to test for some of the most important risks in web applications.

Introduction to web security testing
Covers the fundamentals of web security testing.

Web security testing process
Covers the methodology for web security testing including how to develop a test strategy, test plan, test case specifications, execute, document and retest.

HTTP
Covers HTTP basics, including HTTP requests and responses, URL encoding, RESTful web services, session management, cookies, same origin policy, document object model, intercepting traffic, and local proxies.

Testing for OWASP Top 10
Details how to identify and test for some of the most important OWASP Top 10 security risks in web applications.

Communicating findings
Covers how to rank risks and communicate security findings to various stakeholders. Topics covered include test deliverables, audience analysis, defect reports, evaluating risks, and disclosing vulnerabilities.

Labs
Labs for this course include:
• Intercept HTTP request/response
• Set up a local proxy
• Configure it to capture http traffic from the browser
• Intercept proxies
• SQL injection
• SQL injection from form input
• Challenge on enumerating secret question answers
• Cross-site scripting (XSS)
• Reflected XSS
• Stored XSS
• XSS and client-side tampering
• Communicating findings
• Evaluate risks
• Write a defect report
• Communicate the defect and its risk to management
Red Teaming

Red teaming is a goal-based assessment approach that allows organizations to gain insight into how their security posture is when faced with a real threat. This hands-on Red Teaming course introduces students to the concepts of red teaming and how it's different from traditional vulnerability testing. The course also includes guidance for the organization on creating and maintaining its own internal red teams. Students in this course are introduced to physical, social, and electronic testing methods that can be utilized during red team engagements.

**What is red teaming?**
Students learn how to emulate adversaries to provide depth during an assessment

**Thinking maliciously**
Students think like an attacker, ask questions about trust, and analyze potential assumptions and possible attacks with the ultimate objective of knowing the enemy.

**What is social engineering?**
This section provides a behind-the-scenes look on social engineering. Topics covered include:

**What does a red team look like?**
This section discusses key roles, requirements of a leader and participants, their skill sets, and organizational placement.

**Physical bypass techniques**
This section discusses shims, bump keys, and under the door tools.

**RFID cloning**
Discusses RFID (radio frequency identification) badges, their frequencies, and tools for cloning.

**Scoping a red team assessment**
Explains factors to consider when scoping the assessment, its length, staffing, and limitations.

**Putting together a red team playbook**
This section examines the red team playbook.

**Phases of a red team assessment**
Covers the five phases of a red team assessment, from reconnaissance to report writing, along with their pitfalls.

**Getting organizational buy-in**
Students craft a mission statement, write goals, and learn how to sell their red teaming effort.

**Report writing**
Covers what students should include in their report, how to present attack scenarios and threat findings, and how to provide good remediation advice.

**Labs**
Tools that aid students during the reconnaissance and exploitation phase of an assessment are used the exercises below:
- GooFile: An open source tool which discovers files with a given extension on a target domain
- theHarvester: A specialized tool for discovering corporate email addresses
- Maltego: An extremely powerful open source intelligence discovery tool
- Social engineering toolkit (SET): A framework used to automate several facets of an email-based phishing attack
- Nmap: The de facto standard for port scanners
- Metasploit: An open source exploitation framework
- Metasploitable: A known vulnerable host with many possible avenues for compromise
Mobile Threat Modeling Microservices

Discovering weaknesses in the design of a system is the specific goal of threat modeling. Synopsys’ threat modeling approach can reveal security issues not fully addressed by the traditional methods of penetration testing and secure code review. Organizations benefit from this software design analysis because you can perform it without code to discover potential vulnerabilities early in the development cycle.

This lab-reinforced Threat Modeling Microservices course provides a deep dive into system threat models with topics such as decomposing and modeling the system, inventorying services, infrastructure, data stores, modeling attack possibilities, evaluating and interpreting the threat model, and creating a traceability matrix.

Threat Modeling Microservices brings to light potential weaknesses in the design of microservices-oriented architecture. This course highlights common problem patterns and solutions in both the services and supporting infrastructure.

Introduction
A brief, initial section introduces the company, the instructor, and invites students to talk about their expectations and current knowledge level. It establishes objectives and topics for this course. After opening remarks, this section explains the following threat modeling topics:

- Software security initiative (SSI) capabilities
- Threat modeling definition
- Vocabulary
- Approaches to threat modeling
- Types

Synopsys threat model process
Synopsys brings years of knowledge and experience to its threat modeling approach. Conceptually, threat modeling is a simple process, but it takes time and repetition to get better at creating threat models. This section breaks down the following topics:

- Threat modeling process
- Threat model types and their characteristics
- Threat model stencil

System threat models
This section provides a deep dive into system threat models with the following topics:

- Decomposing and modeling the system
- Inventorying services, infrastructure, and data stores
- Gaining understanding from interviews
- Understanding different types of models from development and infrastructure
- Identifying assets, controls, and threat agents
- Modeling attack possibilities
- Evaluating and interpreting the threat model
- Creating a traceability matrix

Lab
System threat model
This lab reinforces what was learned in the course through a series of exercises:

- Students work in independent groups to build an entire threat model for a fictitious microservices-oriented system with a component diagram
- Even with a defined process, students come up with and then discuss the different threat models
Threat Modeling Multitier

Discovering weaknesses in the design of a system is the specific goal of threat modeling. Synopsys’ threat modeling approach can reveal security issues not fully addressed by the traditional methods of penetration testing and secure code review. Organizations benefit from this software design analysis because you can perform it without code to discover potential vulnerabilities early in the development cycle.

This lab-reinforced Threat Modeling Multitier course provides a deep dive into system threat models with topics such as decomposing and modeling the system, inventorying services, infrastructure, and data stores, modeling attack possibilities, evaluating and interpreting the threat model, and creating a traceability matrix.

Introduction
After opening remarks, this section explains the following threat modeling topics:
• Software security initiative (SSI) capabilities
• Threat modeling definition
• Vocabulary
• Approaches to threat modeling
• Types

Synopsys threat model process
Synopsys brings years of knowledge and experience to its threat modeling approach. Conceptually, threat modeling is a simple process, but it takes time and repetition to get better at creating threat models. This section breaks down the following topics:
• Threat modeling process
• Threat model types and their characteristics
• Threat model stencil

System threat models
This section provides a deep dive into system threat models with the following topics:
• How to decompose and model the system
• Different types of models from development and infrastructure
• Identifying assets, controls, and threat agents
• Modeling attack possibilities
• Evaluating and interpreting the threat model
• Creating a traceability matrix

Lab
System threat model
This lab reinforces what was learned in the course through a series of exercises:
• Students work in independent groups to build an entire threat model for a fictitious system with a component diagram
• Even with a defined process, students come up with and discuss the different threat models
Embedded and IoT
Embedded Systems Security

The Embedded Systems Security course provides an introduction to security engineering for professionals who develop embedded, Internet of Things (IoT), or other integrated systems. Course content is geared toward those students who have a firm understanding of the principles of designing, engineering, or developing non-IT systems and seek to understand the influence of security as a stakeholder in design.

Students are provided with a base understanding of cyber security as it relates to various systems and the processes that should be present within their engineering life cycles. The course takes the approach of understanding risks and vulnerabilities typically present in these systems, and outlining processes and techniques to assist in developing software and embedded systems to minimize cyber security risk.

Nomenclature and concepts
Outlines the nomenclature and standardized vocabulary used throughout the course.

Common vulnerabilities
Describes commonly seen vulnerabilities and risks observed in both software and embedded systems.

Understanding the system of interest
Assists students with the identification of the system function and composition of embedded or integrated systems with respect to their influence on security analysis.

Embedded systems attack taxonomy
Outlines the common taxonomy of exploring and testing an embedded system.

Common embedded attack patterns
Examines several attack taxonomy elements.

Tenets of embedded systems security
Presents an overview of security tenets for embedded systems.

Security in the systems development life cycle
Covers the security-related engineering processes and software development life cycle touchpoints to integrate security as a stakeholder in design.

Threat modeling for embedded systems
Examines the avenues that may be exploited or that pose risk to a proposed system.

Risk assessment for embedded systems
Introduces the processes and techniques for risk assessment related to cyber security issues.

Nontechnical mitigating controls
Discusses nontechnical mitigating controls for addressing security risks in embedded systems.

Standards references
Explains various standards references.

Labs
In a lab environment using instructor-provided tools, students are provided with hands-on exercises demonstrating some of the attack methods described in the course.

Note: Optional hands-on labs are available for classes with sufficient technical experience to complete lab-based exercises. A background in implementation, coding, or technical engineering is recommended for this material.
Software Security 101
This webinar introduces software security fundamentals, their importance, and business impact. It elaborates on activities that are leveraged to build and maintain secure software. It also includes a high-level overview of some of the more common web application vulnerabilities, their causes, and how they can be prevented.

AI in Security
Artificial intelligence (AI) is becoming a standard tool in the modern age. Almost every field of human endeavor seems to be employing AI to increase its effectiveness, and cyber security is no different. Regardless of hat color, security experts are adapting AI-based techniques to reach their goals. This webinar explores a few of the more popular security techniques in use today by cyber security experts, and what type of tasks they are benefiting, as well as a few drawbacks that prevent them from being more effective.

Big Data Security
There's a lot of talk surrounding Hadoop, Redis, Cassandra, MongoDB, and other "big data" databases. Whatever happened to an old-fashioned Oracle or MySQL database? Why do we need these big data databases and how secure are they? You might hear that all this complexity could be offloaded to the cloud, but what security considerations does your organization still need to be responsible for? In this webinar, we seek to demystify big data and most importantly, we talk about security considerations that your organization needs to be aware of, especially as more workflows are being shifted to the cloud.

Open Source Security
Over the last 20 years, open source has evolved from the fringes of software development to become the core of most modern applications, operating systems, and cloud infrastructures. According to Gartner and Forrester, modern applications are often composed of as much as 90% open source code. The benefits of open source are clear: reduced development time and costs, accelerated innovation, and in theory improved code quality and security. However, as highly publicized vulnerabilities and exploits at Equifax and other companies show, lack of visibility and control of open source can leave software and organizations exposed to security and quality risks, as well intellectual property and legal risks related to compliance with open source licenses.

In this webinar, Synopsys explains how organizations get blindsided by open source security, quality, and compliance issues, and provides insight into best practices for managing these risks without slowing the pace of development.

Biometrics
In our daily lives, we regularly have to verify that we are who we say we are. Conventionally, this authentication process is achieved by using something we know, such as a username and password, or something we have, such as a passport or driving license. Recently, this authentication process has moved toward using something we are; this is known as biometric authentication. This webinar focuses on some of the popular biometric modalities including face, fingerprint, voice, and eye. The aim of this session is to provide an introduction to biometrics, analyze the recent developments/attacks, and examine the adoption of biometrics in financials.
Securing Containers
Containers have changed the way applications are deployed. In the past few years, “containerization” gained a lot of traction because it makes it easy for an application team to build, package, and distribute a microservice or an application across different environments. Docker has emerged as the leading container technology for packaging and deploying services or applications. However, security is a challenge when it comes to deploying containers securely. Container security refers to the protection of the integrity of the containers, including both the application and the infrastructure it uses.

This webinar explains what container security is all about, and how an organization can reap the benefits of containerized applications in a secure manner. It also covers best practices for securing the deployments at scale.

OWASP Top 10 Essentials
Taxonomies provide a common vocabulary for professionals to use when discussing software security vulnerabilities. The OWASP Top 10 list is the most widely used taxonomy for web application security. The OWASP Top 10 covers the most critical web application security defects. It is created by security experts from around the world who have shared their expertise to produce this list. This webinar focuses on the basics of each of the Top 10 vulnerabilities.

SAML Fundamentals
SAML allows multiple applications, APIs, or cloud providers to use a central identity provider to handle user logins, thus giving an organization centralized control over user onboarding, access control, and logging. However, given its key role in controlling access to an organization's resources, incorrect use of SAML can have serious security consequences. This one-hour webinar covers common security issues in SAML implementations and how to avoid them.

Annual Security Awareness
Software is everywhere and connected. We’ve seen the things connected to the internet evolve from computers, cell phones, and tablets, to microwaves, diapers, and egg cartons. The influx of connected devices has led to an exponential increase of automation in our daily lives. Hackers have noticed this as well. Each time we add yet another connected device to the network, we increase the likelihood of a breach of our personal and sensitive information.

Hackers have an ever-expanding toolkit to get to our data. There certainly exist tools of the trade that make many highly technical attacks easier. Yet sometimes the attack isn't technical at all. The attacker may prey on our emotions by appearing friendly, helpful, lost, confused, and even threatening to get us to reveal sensitive information.

As systems users, we can protect ourselves. In the webinar, learn how to be aware of the dangers, understand how the attackers accomplish their goals, and discover what you can do to thwart them, so you can prevail in keeping your systems safe.
Product Training
Coverity Server Administration Essentials

Course Description
This course provides DevOps and server administrators the understanding they need to install, configure, and manage a Coverity Connect server. This course includes a strong emphasis on practical hands on labs.

The course describes how to validate your Coverity configuration and monitor the server’s status. This course includes hands on labs demonstrating how to add users and groups, and configure Role Based Access Controls to ensure users get only the permissions they need. In addition, we will discuss how to organize projects and streams on the system, proper system backups, and other key features of the server. Students will also practice creating custom views that can be used in reports and for automatic issue notifications.

After completing the course, students will be able to:
• Complete Coverity connect installation and manage upgrades
• Setup SCM and JIRA integrations
• Create and organize Projects and Streams
• Setup components
• Create and customize triage stores
• Add or import users and groups
• Configure permissions for users
• Customize issue categorization
• Facilitate the use of the Coverity desktop tools by end users

Intended Audience
• DevOps
• Server Administrators
• Security teams

Delivery Format
• Traditional Classroom
• Virtual Classroom

Class Duration
• 12 hours
Coverity Build Administration Essentials

Course Description
This course provides DevOps and build masters the understanding they need to set up, build and analyze projects with Coverity. This course includes a strong emphasis on practical hands on labs, both code-based and language-independent.

The course describes how to plan and implement the onboarding of a project into Coverity. This includes demonstrating configuration options appropriate for standard compilers, tools, and processes. Students will learn how to customize the analysis of a project so that it uses only the checkers that are most relevant to them. Students will also learn how to specify advanced checkers options to achieve even more precise results. The course will cover common failures and appropriate remediation steps, as well as best practices to maximize user adoption.

After completing the course, students will be able to:
• Configure Coverity to work with build environment and compilers
• Implement best practices to optimize build performance and ensure proper analysis
• Resolve common build and analysis failures, including false positives and missed defects, through function modeling
• Configure the Coverity Analysis Engine to provide appropriate analysis coverage for different languages and development processes
• Integrate with source control and issue tracking systems, such as Subversion, Git, and Jira
Coverity Essentials for End Users

Course Description
This course provides Coverity End Users the understanding they need to maximize the value they will receive from Coverity. Students will learn how to efficiently inspect, investigate, and triage vulnerabilities identified by Coverity.

This course starts with a short overview of what Coverity is and how it works. The course then covers how to use custom views to help prioritize, examine and triage Coverity findings. The course ends with a demonstration of how the Coverity desktop tools can be used to increase productivity.

After completing the course, students will be able to:
• Use Quick filtering
• Create and share custom views with advanced filtering
• Examine issues
• Understand events
• Issue instances/occurrences
• Triage issues
• Understand historical audit trails
• Use Coverity desktop plugins
• Create custom notifications

Intended Audience
• Developers
• Security Team members
• DevOps
• Team leads

Delivery Format
• Virtual Classroom

Class Duration
• 2 hours
• BD Essentials
Defensics Essentials

Course Description
This course provides developers and test engineers the understanding to set up, configure, and run fuzz testing with Defensics. This course provides several hands-on lab exercises where the student is able to test the skills they learn in various lessons.

The course walks through the basics of fuzz testing concepts, continues with setting up a Defensics environment, and proceeds to tackle different test scenarios with varying protocols. The course teaches the importance of Secure SDLC and emphasizes fuzzing guidelines to be followed when executing tests in any environment.

After completing the course, students will be able to:
• Acquire and install Defensics and various test suites
• Run and automate fuzz test execution through the command line
• Configure different protocol test suites for testing
• Understand the results after successful test runs
• Gather the results and provide meaningful data to developers
• Use file fuzzing to exercise non-network protocol implementations

Intended Audience
• Developers
• Test Engineers
• Security Engineers

Delivery Format
• Traditional Classroom
• Virtual Classroom

Class Duration
• 12 hours
Black Duck Essentials

Course Description
This course provides teams with the understanding needed to set up and build out a Black Duck environment, so they can quickly get results and begin managing their open source software usage. This course includes a strong emphasis on practical hands on labs, and walks participants through fundamental Black Duck tools as well as the latest features.

The course is constructed around scenarios in which participants complete various configuration steps. These include organizing projects and versions in the Black Duck interface, running scans with Synopsys Detect, working with the Bill of Materials, and applying policy rules effectively. Participants will learn how to optimize their scan configuration to increase the relevance of results. They will also learn best practices around editing the Bill of Materials, and remediating security and license risks.

After completing the course, students will be able to:
- Create Black Duck projects in an organized, scalable way
- Implement best practices to optimize scanning with Synopsys Detect
- Use adjustment features at the project and version level to manage scan results in the Bill of Materials
- Effectively use remediation tools, including the security tab and license terms, to triage security and license risks
- Implement best practices for policy management across projects
- Generate reports that track relevant project details, along with Notices files for versions

Intended Audience
- Development Security Operations
- License Compliance Owners
- Architects
- Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 12 hours