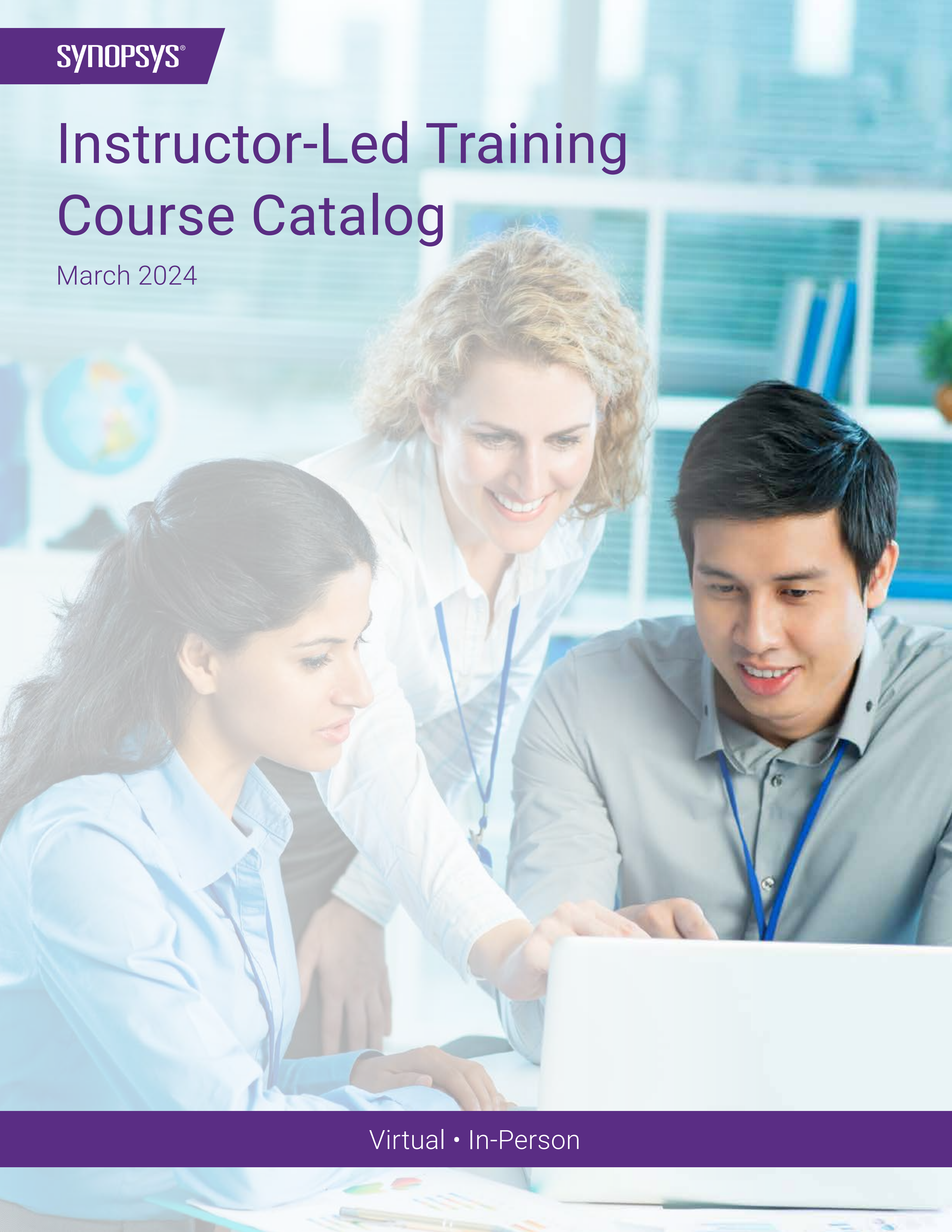
SYNOPSYS®

Instructor-Led Training Course Catalog

March 2024



Virtual • In-Person

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.
675 Almanor Ave
Sunnyvale, CA 94085

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

Table of contents

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 areas of expertise. Course owners use their experience in solving customers' challenges to inform course direction. Similarly, we use certified practicing consultants as instructors. Instructors are able to share real-life examples from previous customer interactions with the students.

What this means to you is that our courses aren't just textbook best practices; our courses have experience baked in from design through delivery.

Synopsys is also the creator and leader of the BSIMM (https://www.bsimm.com). Insights from the BSIMM, as well as from BSIMM assessors, influence both course and catalog direction.

Our Curriculum

Synopsys' curriculum is a series of complementary courses designed to meet your organization's needs. You can select the courses that best match your audience's level of experience, roles, and development platforms.

Our courses are grouped into the following software security topics:

- **Emerging Technologies:** This section covers the latest trends and innovations in cybersecurity that can help you protect your software products from current and future threats such as AI/ML.
- **Fundamentals:** Your software security journey starts here. Fundamentals courses are designed to get you started.
- **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 and Platforms:** Knowing the weaknesses in your chosen language or platform is the only way to avoid those weaknesses that lead to security vulnerabilities. Languages and 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, and 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.
- **Embedded and IOT:** Today's world is more connected than ever, which means vulnerabilities are everywhere. Learn how to apply security engineering practices and techniques in the development of embedded, Internet of Things (IoT), or other integrated systems.

Delivery Models

Virtual or In-Person: Your Choice

If you have a distributed workforce, your participants can avoid travel and time away from the office using our Virtual Instructor-Led Training (vILT). vILT is separated into shorter sessions to optimize participant engagement. vILT can be delivered over consecutive working days or on a weekly basis depending on your team's preference. Virtual training is a cost-conscious training delivery method for supporting your employees' professional development while working remotely. Our instructors are trained to engage your audience through group discussion and interactive hand-on labs designed to simulate real-world environments. Instructors can make course adjustments to better complement the needs, interests, and experience level of your participants.

If you prefer traditional instructor-led training, our certified instructors will travel to the location of your choice.

Instructor-led courses are held on your schedule in the format that works best for you.

Synopsys uses a number of training strategies to assist in participant engagement, including hand-on labs using our cloud-based VM solution, breakout groups, live demonstrations, white boarding, videos, and polling.

	VIRTUAL ILT	CLASSROOM ILT
Instructor type	Full-time security professional	Full-time security professional
Activities	Hands-on labs	Hands-on labs
Student materials	Digital	Digital
Location of students	Distributed and remote	On-site
Delivered	Globally	Globally
Travel costs	$0.00	Varies
Number of students supported	Up to 20	Up to 20
Training topics available	Comprehensive catalog	Comprehensive catalog
Training duration	4 hour sessions for multiple days	8 hours in one day

Emerging Technologies

Emerging Technologies

New in 2024

Principles of AI/ML Security

Intended Audience
- Architects
- Developers
- DevSecOps
- 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:
- Recognize industry standards and governance
- Identify AI/ML security risks at each development stage
- Apply threat modeling techniques to analyze security of AI/ML projects
- Apply threat modeling techniques to mitigate risks in AI/ML projects

Description
The Principles of AI/ML Security course aims to equip participants with an understanding of the baseline ideas for securing this greenfield technology, focusing on generative AI and its applications, industry trends, governance standards, and the risks that affect both the business and the technology itself. Participants will learn about secure integration, model explainability, and verification techniques, and they will explore common AI/ML architectures and their specific vulnerabilities. Through labs on penetration testing and threat modeling, including hands-on exercises in a prebuilt sandbox environment and a hybrid exercise focusing on Retrieval Augmented Generation/Large Language Model (RAG/LLM) architecture, attendees will gain practical experience with exploiting and mitigating risks in AI/ML projects while applying comprehensive threat modeling methodologies to secure AI/ML applications effectively.

Introduction: This section offers a comprehensive overview of artificial intelligence (AI), machine learning (ML), and generative AI (GenAI). It provides in-depth definitions and explores the various types of these technologies. Moreover, the section examines the benefits of these technologies, including their potential for increasing efficiency, productivity, and innovation. It also highlights the challenges and potential pitfalls that come with these technologies, including ethical considerations such as bias, the possibility of misuse, and explainability.

AI/ML Security Risks: This section delves into the vulnerabilities identified in AI/ML projects. The discussion encompasses issues such as bias, adversarial attacks, and training data integrity. It also highlights potential threats at different software development life cycle stages such as at design, development, and deployment. Threats range from bias to data poisoning and model manipulation. This section also touches on industry standards, including the OWASP LLM Top 10 and EU standards. Additionally, this section provides insights into Executive Order 14028 and what we should anticipate from the potential AI security standards that NIST might introduce in the upcoming year.

Industry Best Practices and Threat Modeling: This section discusses various security principles and frameworks for AI/ML projects, including secure data acquisition and management, threat modeling, secure design practices, model explainability and verification, and applying threat modeling to AI/ML. It also covers common AI/ML architectures such as image recognition, search, data classification, and specific training methodologies like RAG.

Labs
- Hands-on penetration exercise
- Hands-on threat modeling exercise

Fundamentals

Principles of Software Security

Intended Audience
- Architects
- Developers
- Managers
- QA Engineers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course Objectives for the Introduction Module
At the end of this course, you will be able to:
- Recognize the importance of software security
- Identify the obstacles that software security faces
- Understand the characteristics of a successful software security initiative
- Describe key software security activities

Course Objectives for the Requirements Module
At the end of this course, you will be able to:
- Recognize common attacks on software
- Recognize common solutions and patterns to mitigate attacks on data, functionality, and resources
- Recognize security requirements to mitigate common vulnerabilities

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. The objective of this course is to identify the obstacles that software security faces, and how teams can employ successful software security initiatives within their organization to overcome them.

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 half-day **Requirements** 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

Labs
- Security requirements for use cases:
  - Authentication
  - Authorization
  - Resource management
  - Data interpretation
  - Data in use
  - Data in motion
  - Data at rest

Attack and Defense

Intended Audience
- Architects
- Developers
- Managers
- QA Engineers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course Objectives
At the end of this course, you will be able to:
- Recognize common attacks on software
- Recognize common solutions and patterns for mitigating these attacks
- Recognize how to avoid common vulnerabilities

The Attack and Defense course aims to lay the foundation for participants' understanding of the broader cybersecurity context, and why software attacks are a crucial concern. It focuses on how the threat landscape has evolved over time, and 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 be 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

Understanding the cybersecurity landscape: This section discusses digital transformation, smart cities, and their associated threat landscape.

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

OWASP Top 10

Intended Audience

- Architects
- Managers
- QA Engineers
- Web 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 the flaws and weaknesses covered in the OWASP Top 10
- Understand how attackers exploit these flaws and weaknesses
- Understand how to protect against these issues in your applications by applying safer design patterns, coding, and testing practices

This course focuses on the most important security defects found in web applications, covering all issues in the latest OWASP Top 10 list. Each topic describes a vulnerability and provides guidance for remediation. This course also provides demonstrations and practical hands-on exercises where students learn what impact these security issues can have on web applications.

What is the OWASP Top 10?

Taxonomies provide a common vocabulary for professionals to use when discussing software security vulnerabilities. The OWASP Top Ten list is the most widely used taxonomy for web application security. The OWASP Top Ten 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.

OWASP Top 10

This is the main section and covers the 10 most critical web application security risks, as defined in the latest OWASP Top 10:

- **A01 Broken Access Control**
  - Authentication vs. authorization, privilege escalation, tampering
- **A02 Cryptographic Failures**
  - Failures related to cryptography often leading to sensitive data exposure or system compromise
- **A03 Injection**
  - Dangers of mixing data with code
  - Cross-Site Scripting resulting from unencoded, unvalidated, and untrusted user-supplied data
- **A04 Insecure Design**
  - Risks related to design flaws
  - Adding the required controls to your system to build a solid foundation for the rest of your application stack since security holes can exist in your application even before you write a single line of code
- **A05 Security Misconfiguration**
  - Misconfigured servers, lack of knowledge on installed features
  - Specific type of Server-Side Request Forgery (SSRF) attack
- **A06 Vulnerable and Outdated Components**
  - Why and how does this happen?
- **A07 Identification and Authentication Failures**
  - Broken authentication and session management
- **A08 Software and Data Integrity Failures**
  - Regarding assumptions related to software updates, critical data, and CI/CD pipelines without verifying integrity
  - Causes of deserialized vulnerabilities
- **A09 Security Logging and Monitoring Failures**
  - Secure logging and monitoring
- **A10 Server-Side Request Forgery (SSRF)**
  - Dangers of remote resources specified by user input

Labs and Demos

This course includes a variety of labs and demos for students to practice their skills.

Mobile

Defending Android

Intended Audience
- 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 the Android platform
- Understand Android security features
- Understand risks relevant to the Android platform
- Apply defensive programming techniques

The Defending Android course begins with a 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 preventing common application security risks, which are explored within the framework of the OWASP Top 10 Mobile 2024 Security Project.

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.

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

Intended Audience
- 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:
- Recognize iOS application security issues
  - Understand the iOS platform
  - Understand iOS security features
  - Understand the iOS risk landscape
- Apply defensive programming techniques

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 risk, 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 requested to have a Mac with OS version 10.15 (Catalina) or higher and running the latest version of Xcode.

Cloud Platforms

Securing Azure

Intended Audience
- Cloud Administrator
- DevOps
- Full-stack 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 cloud computing with emphasis on security considerations
- Understand the various service offerings
- Comprehend Azure security features

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.

Databases: This section discusses the various database services that Azure offers including fully managed relational, NoSQL, and in-memory databases. These span proprietary and open source engines to fit the needs of modern app developers.

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

Intended Audience
- Cloud Administrator
- DevOps
- Full-stack Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 8 hours

Course Objectives
At the end of this course, you will be able to:
- Secure a cloud application using common cloud-native technology including: AWS IAM, KMS, EC2, VPC, and S3
- Use common tooling in a hands-on fashion to secure an example cloud application
- Examine the threat model for a typical cloud application, differentiating security considerations for cloud-native vs. on-premises applications

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, EC2, S3, RDS, KMS, Serverless Lambda, 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: This section discusses cloud risks and prepares the students for the lab environment.

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).

Serverless (Lambda): This quick start section discusses common security considerations for Lambda.

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

Live demos
See common security issues found in the AWS environment, such as IAM misconfiguration, exploiting an EC2 metadata weakness, and elevating permissions via a Lambda exploit.

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
- Hard mode

Intended Audience

- Cloud Administrator
- DevOps
- Full-stack 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 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

Cloud Platforms

New in 2024

Securing Kubernetes

Intended Audience
- Architects
- DevOps
- Full-Stack Developers
- Technical Managers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 16 hours

Course Objectives
At the end of this course, you will be able to:
- Understand key risks when using Kubernetes
- List the available security controls
- Describe protections against attacks on Kubernetes
- Identify common Kubernetes security tools

Kubernetes has emerged as the leading orchestration technology used by organizations large and small for packaging and deploying microservices in applications. The Securing Kubernetes course teaches how to protect the integrity of containers running in a Kubernetes cluster. This includes both the application and the infrastructure. This course explains what Kubernetes is about, how an organization can reap the benefits of secure container deployment, and best practices.
- **A brief intro into containers:** Covers some container basics
- **Container risks and threats:** Informs about key risks such as insecure container images
- **Orchestration:** Explains how the life cycle of containers is managed
- **Kubernetes attack surface:** Discusses the attack surface, framework, and scenarios
- **Securing Kubernetes:** Details security capability models, namespaces, service accounts, authentication, and authorization
- **Networking:** Explains networking security, policies, and authentication
- **Secrets management:** Includes third-party secrets management
- **Admissions:** Discusses admission controllers, pod security admission, and gatekeeper
- **Resource consumption and availability:** Talks about resource quotes and highly available Kubernetes
- **Deployment time security:** Includes topics like container sandboxing, control groups, and dangerous capabilities
- **Runtime security:** Discusses auditing workloads
- **Logging and monitoring:** Discusses adding a cluster and auditing policy
- **Supply chain security:** Includes discussion on image signing, scanning, and verification
- **Real-world case studies**

Labs
- Analyze misconfigured and malicious container images
- Benchmark Kubernetes cluster and hardening
- Identify and fix Kubernetes misconfigured RBAC policies
- Understand SSRF in the Kubernetes world and its fix using network policies
- Explore DoS proofing the memory/CPU resources
- Understand container escape to host system

Defensive Strategies

Securing Code Using Static Analysis

Intended Audience
- DevOps
- QA Engineers
- 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:
- Recognize the importance of static analysis
- Understand where static analysis fits in your SDLC
- Apply static analysis tools in your SDLC

The Securing Code 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

Securing Open Source

Intended Audience

- Administrators
- Architects
- DevOps
- Full-stack Developers
- Managers
- QA Engineers
- Security Practitioners

Delivery Format

- Virtual Classroom

Class Duration

- 4 hours or Custom

Course Objectives

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

- Refresh your knowledge of OSS communities and utilization
- Understand the legal, security, and operational impacts of OSS
- Build a process for OSS approval and organizational compliance

Open source software (OSS) is defined as a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software to anyone and for any purpose. As the role of developers has grown more vital, so has the prominence of open source code. Today, open source libraries are the foundation for every application in every industry. It is so prevalent that many code owners are not aware of all the open source components in their software.

The Synopsys Securing Open Source instructor-led training course enables students to establish trust and limit risks to the business through education and awareness surrounding OSS obligations and implications. This course provides Application Development, Operations, Legal, Security, and DevOps teams the understanding needed to secure open source within their organization from a program and compliance-based lens. Content includes developer aides to manage OSS in your environment as well as hands-on labs and case studies covering real world open source challenges and tooling for automation and scaling to your business.

Course topics include:

- **Introduction:** Covers the definitions, historical events, individuals, and organizations that formed the open source community and foundations which we rely upon today
- **License Fundamentals:** Helps students understand and interpret the fundamentals of open source pertaining to licensing and development
- **Open Source Maturity:** Analyzes the components of an OSS program structure and discusses program implementation excellence within a Synopsys OSS framework
- **Open Source Communities:** Focuses on acceptable activities and clear policies to drive safe behaviors within the open source community while protecting intellectual property for patents and secrets
- **Third-Party Open Source:** Describes the open source supply chain and vendor compliance with commercial arrangements containing OSS
- **Metrics:** Enables students to understand and interpret data required for effective OSS metrics, and allows them to apply or articulate the OSS program value using metrics
- **Developer Practices:** Students learn about methods developers use to integrate open source components, as well as how they analyze and connect these methods in practice
- **DevOps Tooling Capabilities:** Developer patterns, how to control the source of open source, and how to intelligently orchestrate compliance are some of the topics discussed here
- **Open Source Operations:** Explains how OSS impacts and identifies unique risks and threat patterns

Labs

Black Duck scan tooling:

- Installation and configuration
- Localized scanning results

Securing Software with DevSecOps

Intended Audience

- Architects Administrators
- Architects
- DevOps
- Full-stack Developers
- Managers
- QA Engineers
- Security Practitioners

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:

- Recognize the difference between DevOps and DevSecOps
- Differentiate between Agile, CI/CD, and DevSecOps
- Understand the process to achieve DevSecOps by bringing together people, process, and technology
- Understand the process of vulnerability management using the tools in the lab

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.

This course is available with a lab selection of Open Source and/or Synopsys security testing tools on an insecure Java or Golang application.

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.

Labs

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)
- Building an application image and integrating its scan (Java pipeline only)
- Integrating a DAST scan (both pipelines)
- Pausing the CI pipeline for manual approval (both pipelines)
- Asynchronous continuous security pipeline (Java pipeline only)

Languages and Platforms

Defending C

Intended Audience
- Architects
- Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 16 hours
- 8 Hours on Request

Course Objectives

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

The Defending C course provides developers with a solid foundation in software security as it relates to the implementation of applications developed in 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.

Course sections

- Risk landscape
- Memory bugs
- Integers
- Strings and streams I/O
- Heap corruption and integrity
- Execution targets
- Secure toolchain

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
- 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
- Integer attacks: Asks students to find integer vulnerabilities in four vulnerable programs and then fix the defects
- 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 call() that contains the best practice in this course
- Bypassing ASLR: Instructs students why they should not rely on ASLR by stepping sequentially though an ASLR bypass
- Fuzzing a vulnerable parser using AFL: Students fuzz a vulnerable parser to find integer vulnerabilities

Note: Although this course is taught best as a 16-hour course, a shortened 8-hour version is available on request.

Defending C++

Intended Audience
- Architects
- Developers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 16 hours
- 8 Hours on Request

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

The Defending C++ course provides developers with a solid foundation in software security as it relates to the implementation of applications developed in 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.

Course sections
- Risk landscape
- Memory bugs
- Sequences, algorithms, and containers
- Integers
- Heap corruption and integrity
- Execution targets
- Secure toolchain
- Modern C++: C++11, C++14, C++17 core language features

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
- 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
- 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
- Exploring new: Examines wrapping due to array new and explores different behaviors under different compilers
- Bypassing ASLR: Teaches students why they should not rely on ASLR by stepping sequentially though an ASLR bypass
- Fuzzing a vulnerable parser using AFL: Students fuzz a vulnerable parser to find integer vulnerabilities

Note: Although this course is taught best as a 16-hour course, a shortened 8-hour version is available on request.

Defending Golang

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

The Defending Golang course helps you to identify security risks common to Golang applications and their impact when the vulnerability is exploited. Equipped with a variety of labs and a demo, this course provides you with best practices for secure Golang programming.

Course topics include:
- **Risk Landscape:** Compares Go to other compiled languages such as C/C++ and Java in this section and provides a mind map of defensive programming problem areas
- **Injection Vulnerabilities:** Focuses on the Cross-Site Scripting (XSS) injection attack, its types, its contexts, and mitigation
- **Files:** Discusses how attackers can use name confusion and path traversal, and explains how to protect against those attacks
- **Other Injection Vulnerabilities:** Provides a brief overview of some injection vulnerabilities such as XML/ XPath injection, HTTP header injection, LDAP injection, and buffer overflows (C/C++), and explores SQL injection (SQLi), log injection, and command injection, and their mitigation in detail
- **Working with XML:** Discusses how allowing untrusted input into XML can cause injection attacks, and how XML eXternal Entities (XXE) can be problematic
- **Concurrency:** Describes in-process concurrency such as race conditions and deadlocks, and interprocess concurrency such as name squatting and time-of-check, time-of-use (TOCTOU); the Go race detector is also introduced
- **Integers:** Provides an overview of handling of integers in Go such as integer types and ranges, also discusses wrapping and conversion errors, conversion rules, and mitigations in Go
- **Error Handling:** Examines error handling in Go and problematic ignored errors using two CVEs
- **Secure Toolchain:** Discusses Gosec, a security tool that performs static code analysis for Golang projects
- **Miscellaneous Topics:** Includes best practices for unsafe packages, external attack surface reduction, privilege reduction, and insecure configuration

Labs
Work with these labs to discover weaknesses discussed in the class in an intentionally vulnerable system and apply appropriate mitigations.

Note: The instructor chooses the labs from this main list based on student preparedness and time availability.
- Go Setup
- JS Injection (XSS)
- Directory Traversal
- Web App: Database Access
- Malicious Commands
- File Upload
- Dangerous OS Calls
- XML eXternal Entities
- Go to the races
- Gosec Tool
- Insecure Configuration: Database Access
- Insecure Configuration: Directory Listing

Defending Java Web Applications

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 the overall approach to securing web applications
- Identify security risks common to Java web applications
- Identify security vulnerabilities in Java web applications
- Apply defensive programming techniques to write secure Java web applications

This course focuses on defensive programming techniques in Java Web 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: Discusses common web application risks, typical Java Web Application risks, risks caused by the configuration of the application server, and a quick overview of OWASP Top 10 critical security risks

Access Control: Discusses privilege escalation, forceful browsing attacks, broken authentication, and parameter tampering

Secure Session Management: Explains secure session management techniques, secure Session ID generation, and secure timeouts

Secure Configuration: Describes general risks, security configuration and error handling

Input Validation: Details the risks of improper validation and output encoding, and provides examples of proper validation techniques for mitigation

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

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

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

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

Labs
- Semantic input validation
  - Processing URLs as string
  - Validating the URL strings via whitelisting
- Performing input validation
  - Web application penetrating testing utilizing Burpsuite
  - Changing values during interception to test input validation
- Using output encoding
  - Preventing cross-site-scripting (XSS) through output encoding
  - Utilizing OWASP Java encoder to encode strings
- Using parameterized queries
  - Protecting a web application from SQL injection attacks
  - Modifying SQL strings using the SessionFactory class to create secure queries

Defending JavaScript and HTML5

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 common risks related to client- and server-side JavaScript
- Identify and fix security vulnerabilities in your JavaScript code and web application configuration

The Defending JavaScript and HTML5 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 related to JavaScript and HTML5 technologies. This course also explains risks present in server-side code written in JavaScript, such as JSON injection, insecure object comparison, prototype pollution, and mass assignment.

Web Application Risk Landscape: This section analyzes browser security features and common web application risks. It also covers risks relevant to JavaScript applications, such as manipulating the DOM, JavaScript execution contexts, and DOM clobbering.

Defensive Programming for Client-Side JavaScript and HTML5: This section covers best practices that will help students write secure code when handling untrusted data in scripts. It also covers using HTML5 features such as web storage, web messaging, and cross-domain communication; iframe sandboxing; and the content security policy.

Defensive Programming for Server-Side JavaScript: This section discusses how to safely handle JSON data, perform object comparison, and avoid the risks of prototype pollution and mass assignment.

Labs
Students find, fix, and verify the remediation of a 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 C#.NET Web Applications

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

The Defending .C#.NET Web Applications course focuses on modern C#.NET 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-indepth, 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: 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

Labs
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

Attacking Strategies

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 at which they operate
- 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

Intended Audience

- Architects
- DevOps
- QA Engineers
- 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 networking basics
- Follow a standard methodology for performing network security assessments
- Recognize common network testing tools and exploitation
- Communicate findings to network administrators and management

Attacking Web Applications

Intended Audience

- Architects
- DevOps
- QA Engineers
- 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 the basics of HTTP and web security testing
- Use tools for intercepting and modifying HTTP communication
- Understand how to exploit common web vulnerabilities
- Present the result of a web security test for different target groups

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

Hackathon

Intended Audience
- Developers
- Security Champions
- Security Professionals
- Engineering
- Other technical roles

Delivery Format
- Virtual Classroom
- Traditional Classroom

Class Duration
- Custom

Course Objective
At the end of this course, you will be able to:
- Discover vulnerabilities in an application

This is a Capture The Flag (CTF) style course that provides an intentionally vulnerable environment for participants to test their exploit capabilities. Event content such as tech stacks, programming languages, exploitation categories, and duration can be tailored to the customer's needs. Synopsys offers three distinct flavors of this course:

- **Pure Hacking:** In this version, participants attack the provided application and environment while adhering to the rules of engagement. Their score for the course is determined by the number and difficulty of the vulnerabilities they successfully discover and exploit. (Level: Beginner – Expert)
- **I teach, you break:** In this version, hints are provided in the form of short lessons throughout the course. The lessons conclude by identifying areas within the application including tools, example exploits, or live walkthroughs. Participants will then practice their newly learned knowledge. (Level: Beginner – Expert)
- **Find and Fix:** In this version, participants identify the vulnerabilities and fix them. Points are awarded only for fixes. This version should include Pure Hacking, and the participants should have DevOps knowledge. (Level: Intermediate – Expert)

In addition to these three flavors, there are different support modalities that you can choose to utilize:

- **Hacking Sprint:** Similar to an instructor-led-training class, the environment and instructors are available for an eight (8) hour duration. The instructors provide both discovery and exploit advice as well as grade remediation. Depending on the number of participants and the flavor chosen, multiple instructors may be required.
- **Hacking Marathon:** The environment is left open for an extended period such as a full week or even a full month. This duration works well for security weeks or security months where participants can attack the environment at their leisure throughout the duration. Instructors set up office hours when they are available to provide advice to the students, run an "I teach, you break" session, as well as grade remediation efforts. The number of office hours needed is determined by the number of participants as well as the duration of the event.

Red Teaming

Intended Audience
- Architects
- Developers
- Security Practitioners

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- Traditional 8 or 16 hours; Virtual 8 hours

Course Objectives
At the end of this course, you will be able to:
- Understand the red teaming concept
- Understand how red teaming differs from traditional testing
- Understand the components of a red team assessment
- Explain red team assessments internally to management
- Understand the process of conducting red team assessments

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.

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

Requirements, Architecture, and Training

Champions Workshop

Intended Audience

- Security Champions

Delivery Format

- Traditional Classroom
- Virtual Classroom

Class Duration

- 32 hours with an option for an additional 8 hours of custom content

Course Objectives

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

- Understand the basics of Software Security
- Understand the roles and responsibilities of a Security Champion
- Mentor your team members on Software Security best practices

Introduction

The Champions Workshop provides Security Champions with the knowledge and skills they need to enhance the security practices of their teams and ensure security best practices are a part of the entire development process.

Benefits of a Champions Workshop

In today's fast paced software development world, security can struggle to keep pace with development. Security often becomes an afterthought, takes a back seat to functionality, or gets tacked on at the end. This tends to produce software with high vulnerability counts or insecure designs. To combat these issues, many companies are creating Security Champions Workshops in order to scale their security capabilities, shift security left int the process, and ensure security moves at the speed of development.

Champions Workshops seek to bridge the gap between security and development teams by providing enhanced security training to a subset of Developers, Engineers, Testers, and Operations Personnel. These Security Champions advance security culture, serve as a focal point for security issues on their teams, and act as a communication channel between development teams and security. This allows for a developer-focused approach to implementing secure development lifecycle processes and techniques.

How Does the Workshop Work?

The Workshop takes a modular approach to champions training, allowing each customer to assemble a course that best fits the needs of their Security Champions. Topics include Security Basics, Securing Apps and Platforms, Secure Development Practices, and Defensive Programming. Our Champions Workshop Manager will work with you to develop an educational path that best fits the needs of your company and your Security Champions.

Example Paths

DevSecOps Path

- Intro to Software Security
- Champions Roles and Responsibilities
- DevSecOps
- Securing Open-Source Software
- Threat Modeling
- Peer Code Review
- Triage and Prioritization
- Cloud Essentials
- Container Essentials

WebDev Path

- Intro to Software Security
- Champions Roles and Responsibilities
- Agile Security
- OSS Basics
- OWASP Top 10
- Attacking Web Apps
- Defending (JS/Java/C#) Web Apps

Threat Modeling

Intended Audience
- Architects
- Developers
- DevOps
- Managers
- QA Engineers
- Security Practitioners

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- 16 hours

Course Objectives
At the end of this course, you will be able to:
- Describe different techniques used for threat modeling
- Explain the Synopsys threat modeling process and methodology
- Use the Synopsys threat modeling approach for analyzing applications and systems

Introduction
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.

The lab portion of this course is available in multiple flavors:
- Microservices
- Embedded
- Customizable

Threat Modeling Introduction: This section defines a threat model and its benefits and discusses well-known threat model methodologies and approaches.

Synopsys Case Studies: Synopsys establishes a hypothetical case study which is used as a foundation for highlighting the various steps/processes of the Synopsys Approach. Using a hypothetical threat model example, this section provides a deep-dive into the execution of threat models. The following are examples of case studies provided by Synopsys:
- Classic Car Conversions
- Embedded System
- Mobile Application
- API Architecture
- Services Infrastructure

Synopsys Threat Model Process: Synopsys brings years of knowledge and experience to its threat modeling approach. Synopsys has established its own way of building diagrams, representing assets and controls, and techniques to introduce consistency in the identification of threats. This portion of the discussion breaks down the Synopsys Approach to threat modeling.

Attack Tree and Threat Traceability Matrix: Synopsys enumerates all possible attack scenarios against a given technology and documents the controls and mitigating factors against a successful compromise. These attack scenarios are documented in the form of an attack tree or traceability matrix.

Labs
This lab reinforces what was learned in the previous sections:
- Students work in independent groups to build an entire threat model for a fictitious system with a component diagram
- Even with a defined process, people come up with different threat models; these are discussed

Note: Although this course is taught best as a 16-hour course, a shortened 8-hour version is available on request.

Embedded and IoT

Embedded Systems Security

Intended Audience
- Architects
- Embedded Developers
- Managers
- Testers

Delivery Format
- Traditional Classroom
- Virtual Classroom

Class Duration
- Custom Scoped

Course Objectives
At the end of this course, you will be able to:
- Understand the fundamentals of integrating security into a given system design
- Understand the methods attackers may use to compromise a system
- Understand the process of decomposing a system to elicit security goals and objectives
- Understand decision-making related to security requirements, remediations, and threats
- Understand where to seek information on specific security guidance and technology

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.