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

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Our Curriculum</td>
<td>5</td>
</tr>
<tr>
<td>Delivery Models</td>
<td>6</td>
</tr>
<tr>
<td><strong>Fundamentals</strong></td>
<td>7</td>
</tr>
<tr>
<td>Principles of Software Security</td>
<td>8</td>
</tr>
<tr>
<td>Attack and Defense</td>
<td>9</td>
</tr>
<tr>
<td>OWASP Top 10</td>
<td>10</td>
</tr>
<tr>
<td>Webinars</td>
<td>11</td>
</tr>
<tr>
<td><strong>Authentication and Authorization</strong></td>
<td>12</td>
</tr>
<tr>
<td>Webinars</td>
<td>13</td>
</tr>
<tr>
<td><strong>Mobile</strong></td>
<td>14</td>
</tr>
<tr>
<td>Defending Android</td>
<td>15</td>
</tr>
<tr>
<td>Defending iOS</td>
<td>16</td>
</tr>
<tr>
<td><strong>Cloud Platforms</strong></td>
<td>17</td>
</tr>
<tr>
<td>Securing Azure</td>
<td>18</td>
</tr>
<tr>
<td>Securing AWS</td>
<td>19</td>
</tr>
<tr>
<td>Securing Containers With Docker</td>
<td>20</td>
</tr>
<tr>
<td>Webinar</td>
<td>21</td>
</tr>
<tr>
<td><strong>Defensive Strategies</strong></td>
<td>22</td>
</tr>
<tr>
<td>Securing Open Source</td>
<td>23</td>
</tr>
<tr>
<td>Securing Software with DevSecOps</td>
<td>24</td>
</tr>
<tr>
<td>Webinars</td>
<td>25</td>
</tr>
</tbody>
</table>
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 activities:

- **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 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.
- **Product Training:** Learn how to use Synopsys products with a live instructor guiding your way. If you are looking for Synopsys product training, please ask your sales representative.
Delivery Models

Virtual, In-Person, or Webinar: 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.

Webinars allow you to reach a large number of participants in a short amount of time. Typically around an hour in length with no attendance cap, use webinars to introduce new topics to your entire workforce or meet an annual training requirement.

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

<table>
<thead>
<tr>
<th></th>
<th>VIRTUAL ILT</th>
<th>CLASSROOM ILT</th>
<th>WEBINAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instructor type</td>
<td>Full-time security professional</td>
<td>Full-time security professional</td>
<td>Full-time security professional</td>
</tr>
<tr>
<td>Activities</td>
<td>Hands-on labs</td>
<td>Hands-on labs</td>
<td>Demonstrations</td>
</tr>
<tr>
<td>Student materials</td>
<td>Digital</td>
<td>Digital</td>
<td>On request</td>
</tr>
<tr>
<td>Location of students</td>
<td>Distributed and remote</td>
<td>On-site</td>
<td>Distributed and remote</td>
</tr>
<tr>
<td>Delivered</td>
<td>Globally</td>
<td>Globally</td>
<td>Globally</td>
</tr>
<tr>
<td>Travel costs</td>
<td>$0.00</td>
<td>Varies</td>
<td>$0.00</td>
</tr>
<tr>
<td>Number of students supported</td>
<td>Up to 20</td>
<td>Up to 20</td>
<td>20-200</td>
</tr>
<tr>
<td>Training topics available</td>
<td>Comprehensive catalog</td>
<td>Comprehensive catalog</td>
<td>Tailored by request</td>
</tr>
<tr>
<td>Training duration</td>
<td>4 hour sessions for multiple days</td>
<td>8 hours in one day</td>
<td>1-2 hours</td>
</tr>
</tbody>
</table>
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

### Labs

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

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.

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.
Authentication and Authorization
Webinars

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.

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.
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 requested to have a Mac with OS version 10.15 (Catalina) or higher and running the latest version of Xcode.
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
- Hard mode
Webinar

Intro to Container Security

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.
Defensive Strategies
Securing Open Source

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

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)
- 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)
- Asynchronous continuous security pipeline (Java pipeline only)
Mobile Webinars

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.
Languages and Platforms
Defending C

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
• Execution targets
• Integers
• Strings and streams I/O
• Heap corruption and integrity
• 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 calloc() 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.
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
- Execution targets
- Sequences, algorithms, and containers
- Integers
- Heap corruption and integrity
- 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

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 name confusion and path traversal which is a type of injection vulnerability.
- **Other Injection Vulnerabilities**: Explores log injection and command injection in more detail and identifies other examples of injection vulnerabilities such as SQL injection (SQLi), XML/XPath injection, HTTP header injection, LDAP injection, and Buffer overflows (C/C++)
- **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).
- **Integers**: Includes wrapping and conversion errors, integer type and ranges, conversion rules, and mitigations.
- **Error Handling**: Examines errors in Go and ignored errors with the help of 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

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.
Defending 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
- Performing Input Validation
- Using Output Encoding
- Using Parameterized Queries

---

**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.

---

**Languages and Platforms**
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 frameworks: Angular: This section introduces the Angular framework, describes its security issues, and discusses secure coding best practices:
- 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 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
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 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 C#.NET Web Applications

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-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

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 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

**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

**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 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
Hackathon

This is a Capture The Flag (CTF) style course that provides an intentionally vulnerable environment for participants to test their exploit capabilities. 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.

- **I teach, you break:** In this version, “hints” are provided in the form of short 15-minute lessons throughout the day. The lessons conclude by identifying areas within the application for the participants to practice their newly learned knowledge.

- **Find and Fix:** In this version of the course, participants are awarded points for discovering and exploiting vulnerabilities similar to the Pure Hacking version above. Once a vulnerability is discovered, participants are encouraged through the awarding of significantly more points to provide remediation for the vulnerabilities by checking in updated code to a forked repository. Once the remediated code is checked-in, a host looks at the provided code and awards points based on the fix. Remediation emphasis can be increased by weighting the points for remediation such that it would be nearly impossible to “win” the Hacking event without providing remediation advice.

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

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 in 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
Webinar

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.
Requirements, Architecture, and Training
Champions Workshop

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 into 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

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

New in 2023
Requirements, Architecture, and Training
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:
- Multi-tier
- 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

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.