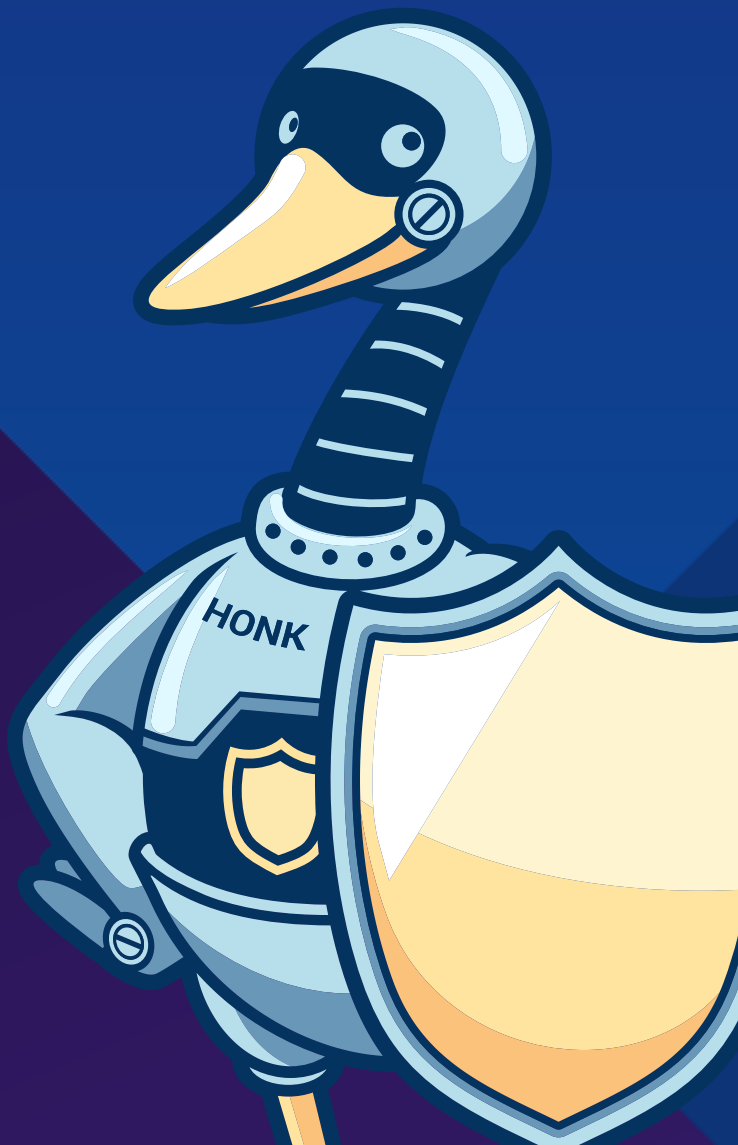
OpenSSF

OPEN SOURCE SECURITY FOUNDATION

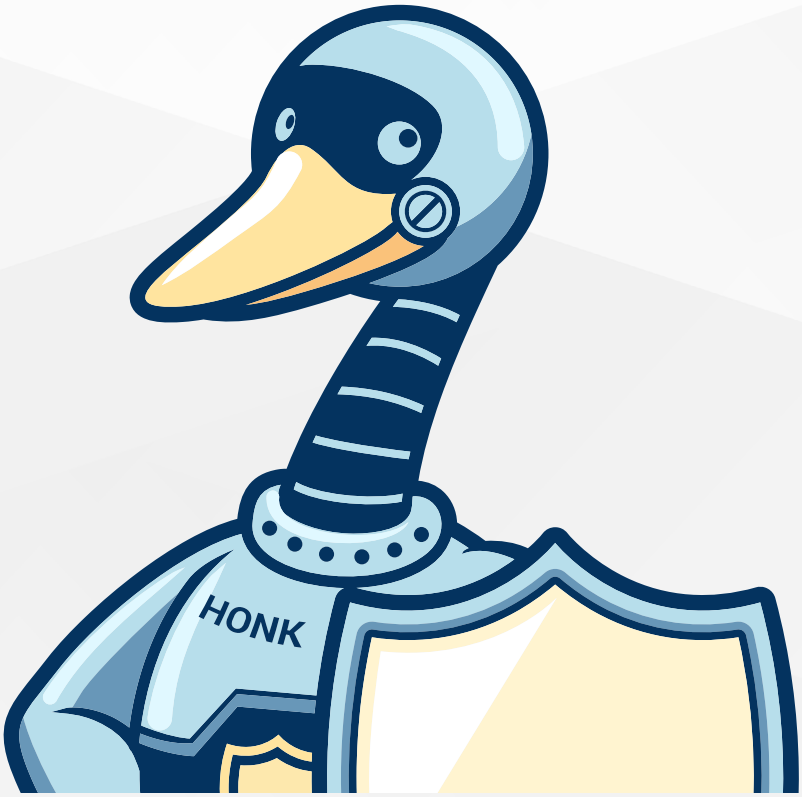
# Plan for Improving Software Developer Security Education




openssf.org

# Summary

This paper provides recommendations on how to improve the security education of software developers worldwide by expanding training materials and incentives for that training. In this report we briefly justify why secure software development education is needed and then summarize the current state of educational materials. We then discuss the OpenSSF education efforts from 2022 through 2023, including the identified need to"collect and curate content", and identify focused requirements. We conclude with a summary of OpenSSF education efforts that we propose for 2024 as well as those underway. Appendix A discusses, in more detail, many related educational materials available. Appendix B discusses some important secure software development lifecycle models.



OpenSSF
OPEN SOURCE SECURITY FOUNDATION

# Table of Contents

# Secure Software Development Education Needed

There is a need for secure software development education.

CSO Online surveys show there's a shortage of cybersecurity skills in general. In 2016, "46% of orgs said they have a problematic shortage of cybersecurity skills"; by 2021 this had grown, as "57% of [orgs] impacted by the global cybersecurity skills shortage". In 2021, there were an estimated 4 million unfilled cybersecurity positions [Dataconomy 2021]. This growth is unsustainable.

Many governments have collectively identified a key factor: software is generally not secure by design, nor is it secure by default (see the paper *Secure by Design*, October 2023). Trying to "secure the insecurable" is unlikely to ever succeed.

The fact that most software is not secure by design should be expected, because software developers typically do not know how to develop secure software. This lack of knowledge means that most software must be insecure, and is often secure than its users deserve. Here is some evidence that developers typically don't know how to develop secure software:

- No top 40 US "coding" or top 5 non-US CS schools required secure coding in 2019 [Forrester 2019].
- Of U.S. News's top 24 CS 2022 schools, only one (University of California - San Diego) requires security for undergraduates.
- One article pointedly noted, "universities don't train computer science students in security".
- 53% of software developers report that their organizations don't ensure training on securing coding [Poneman 2020]
- The third most popular answer for how to improve OSS security was providing more training to the OSS community (per the 2022 v2.0 survey "Addressing Cybersecurity Challenges in Open Source Software" by Stephen Hendrick (VP Research, The Linux Foundation) & Martin McKeay (Senior Editorial Research Manager, Snyk), question q0050mrv). The only higher-ranked items were "define best practices for secure software development" and "provide tools for analyzing and remediating vulnerabilities in the top 500 open source components" - which clearly don't conflict with training.
- One survey claimed otherwise, but it is misleading. The State of Developer-Driven Security Survey, Secure Code Warrior, 2022, found that 89% of developers reported they've received sufficient training in secure coding skills. However, what this survey really showed is that developers know so little that they think they know more than they do (an unfortunate example of the Dunning–Kruger effect). More than half of those respondents were not familiar with common software vulnerabilities, how to avoid them, and how they can be exploited. 92% said they needed more training on security frameworks, and 86% stated they found it challenging to practice secure coding. In short, they thought they knew enough, yet most knew almost nothing and did not know enough to able to do it.

This lack of knowledge impedes all software developers, whether they are developing open source software (OSS) or closed source software. A potential security advantage of OSS is that OSS can have high quality from mass peer review. However, mass peer review is only effective if the reviewers know what to look for. In practice, the knowledge needed for developing secure software is the same regardless of the licensing approach.

While there is no such thing as a silver bullet which will solve all software development challenges, there is nothing as foundational or as frequently used throughout the software and product development lifecycles as knowledge. Automation and tooling are useful to conduct certain tasks at specific stages, but the human element is present during throughout software development. Tools are valuable, but their effectiveness depends on the knowledge of their users. Defensive coding skills, threat modeling, security code review skills, and much more are always with the knowledgeable engineer, and provide omnipresent insights during all activities. Quality education and training in cybersecurity and secure software development, leading to applicable knowledge and skills, becomes a force multiplier that is always present in every task.

# Current State of Educational Materials Summary

There are a vast number of cybersecurity-related educational courses. In this document, we focus on courses specifically aimed at educating software developers about security, including design, implementation, selection of external components, verification, and countering supply chain attacks. For details, see Appendix A, "Current State of Educational Materials Landscape Details ('Collect and Curate')".

Here is a quick summary:

- The OpenSSF Secure Software Fundamentals Course is a popular, free, highly regarded course for up-to-date general information on the fundamentals for developing secure software. It's an eLearning self-paced course, so it easily scales. The course is impactful but lacks a labs feature. It could also be improved by adding pointers to other more advanced and specific materials were available.
- SAFECode has some interesting existing materials that are in need of updating to reflect current development practices. It also doesn't have a "general" course on secure software development, but focuses on specific topics.
- IBM's free Coursera course is perhaps the closest material to the OpenSSF's fundamentals course. It has a lab, but it's focused on web application development with Python, which is very specific.
- Synk Learn - Security for Developers is free and focuses on a short set of lessons to counter OWASP Top 10 vulnerabilities.
- There are courses available from various sources such as Securityjourney.com and Cydrill. These tend to cost thousands of dollars, and may be better suited for commercial entities.
- Numerous security education-focused organizations like ISC2 and SANS have both free and paid content available for practitioners, but these tend to focus on cybersecurity during operations more than application/development security. ISC2 does have the CSSLP, but it is more focused on managing secure software development, and in general asks about things at a high level. These are valuable in their areas of focus, but leave gaps.

We did not survey the many materials that focus on deployment and operation of software to attempt make them secure. We instead want to focus on making software secure by design and secure by default.

# Context: OpenSSF Education Planning Efforts in 2022-2023

To understand the context of this paper, it's important to understand the OpenSSF education planning efforts in 2022-2023.

In the wake of the Log4Shell vulnerability the OpenSSF developed a "mobilization plan" to improve security related to open source software (OSS). Steam 1 of this plan focused on the education of software developers. The OpenSSF created an OpenSSF Education Special Interest Group (SIG) which developed an education plan. This plan identifies three areas:

- **Collect & Curate Content:** This focuses on identifying high-quality available content that already exists and helping identify gaps in desired educational materials.
- **Expand Training:** This takes the data collected from [area] 1, and crafting new materials across the "Three Legged Education Stool" to ensure all types of learners have access to materials that they can relate to and learn from.
- **Reward and Incentivize Developers and Maintainers:** This seeks to promote methods to showcase developers and learners that have taken the coursework and demonstrated their skills to improve their eminence and status within the community and with employers.

When the plan was created, it was expected that large amounts of funding would be available. Both government and industry indicated that Log4Shell had many impacts, and that they would like to prevent a recurrence of anything similar. Therefore, the plan was developed including many items along with estimates of their resource costs. Unfortunately, the funding for this plan has not materialized, for a variety of reasons.

However, we believe all is *not* lost. The three areas still represent a sensible high-level strategy, and the original plan includes interesting ideas. What's needed is a *narrow focus* on lower-cost, scalable, sustainable, biggest-benefit efforts, to turn *ideas* into *practice*. The result might not be a wide variety of capabilities like the larger plan, but it could *happen.*

To do this, we must first *identify* specific requirements and narrowly *focus* on them so we can achieve results. We propose focusing especially on what can be accomplished within one year. We can then focus on collecting and curating content to meet those focused areas, and then implement a focused implementation of the three areas listed above.

# Focused Requirements

Different people have different needs, depending on what they do, what they know, what they need to know, and their resources available (time and money). Often people are grouped into "personas" to identify these differences.

Since resources are limited we propose identifying those focus areas first. This makes other steps easier. For example, identifying relevant existing resources is much harder without knowing what the specific requirements are; it's easier with a narrower focus..

For the moment, we propose prioritizing these educational materials:

| Priority | Target learner | Content | Notes |
|---|---|---|---|
| 1 | Any software developer | Improved version of "fundamentals of developing secure software" course | OpenSSF has a fundamentals course already that is popular, free, and highly regarded. However, it lacks optional labs. Labs improve learning, but using them takes time some practitioners won't have. There's also a request to increase its use of multimedia.<br><br>This needs to be free, since we want it taken by all software developers. Any cost will reduce the number of participants, and thus reduce its impact. Once created, maintenance costs are expected to be relatively low, because fundamentals rarely change. The scale is essentially "all software developers", a scale that requires online education.<br><br>The plan is to develop and release a full set of labs and at least 15 multimedia extensions for entire course by the education SIG team by 2024-09-27. |
| 2 | Manager supervising developers | What managers should expect developers to know and do to develop secure software | Explain what managers should be expecting their developers to do to develop secure software, so they can adequately hire, acquire training for, manage, and fire. We're currently thinking this would be free or at least low-cost, since there's no obvious way to broadly require this. Currently we expect this to be short (much shorter than the fundamentals) and based on Intel's contributed material.<br><br>The plan is to develop a draft and share with reviewers by 2024-06-21. |
| 3 | Software developer (specific ecosystem & specialized topics, e.g., threat modeling) | Deeper security knowledge on a specific ecosystem or topic | This would be a set of educational materials, each focused on a different ecosystem (e.g., programming language) or a specialized topic. At least some may need to have fees, since they may take more effort to maintain and there are many of them. It's unclear what specific area to focus on. To ensure that selection is data-driven, we are working with LF Research on a survey to identify the top areas to pursue and select one for this year. This is likely to have a fee to take the course; profits would go back to OpenSSF.<br><br>The plan is to complete this by 2024-12-20; we hope to complete it by 2024-10-24, though since it depends on the survey, that timeline may be too short. |

There is a rationale for this list:

1. The fundamentals for software developers are fundamental. Many things should be known by *all* developers.

2. Without management understanding and support, efforts are unlikely to be effective. Managers don't need to be able to do the work, but they must understand what needs to be done.

Ecosystem-focused and more specialized materials are helpful. However, it takes resources to develop each.

For part 1 (the fundamentals) we'll take what the current course covers as a starting point. The goal is for software developers to know:

1. **Basics:** What is security, privacy, risk management, etc.

2. **Requirements:** Common security requirements

3. **Design:** Design principles for security (including Saltzer & Schroeder)

4. **Reuse:** How to securely reuse software (especially open source software)

5. **Implementation:** Common implementation vulnerabilities (including all those identified by the OWASP Top 10 and CWE top 25) and how to systematically prevent them.

6. **Verification:** How to verify for security, including common types of tools (SAST, fuzzing, web application scanners, secret scanning) & testing (coverage, negative testing)

We could take steps to validate this list to see if these are the most important priorities that need to be addressed (or if other areas are more critical), or if there are specific kinds of information and/or formats that are preferred. The Linux Foundation could do a survey to gain insights into the views of those participating in OpenSSF.

A few quick notes:

1. **Generalize where easy.** Developing secure software is generally the same if it's OSS or not, and many developers develop OSS and closed source software. Where easily done, we should address developing software regardless. Selection of OSS is not unique to OSS either, so that knowledge is also useful to all**.**

2. **Start with standalone materials.** There's an argument for integrating "how to develop secure software" into materials on how to develop software.

3. **Start with English.** We would love for the materials to be translated into many languages, but starting with English is the obvious place to start.

4. **Education not entertainment.** We want material to be interesting, but learning is the goal. In particular, it's possible to create flashy videos that lack substance and any kind of interaction, where the "learners" learn little. The goal should always be that learners learn.

5. **Make it Sustainable.** It must be affordable to update material as new attacks and defenses are found. We also need to determine how to fund maintenance.

We suggest not focusing on these for now in OpenSSF, to conserve limited resources:

- C-suite. Until other things are in place this isn't likely to help, and their time is rare.
- Courses for researchers in developing secure software. Universities are more likely to play that role, and there aren't as many positions available.
- Operations. Many others are doing education for operations. Also, many OSS projects don't have "operations" in the traditional sense; they just release their code.

- Specialized public policy or regulatory compliance needs. The fundamentals course notes some broad regulatory issues that widely apply (e.g., GDPR). However, we are not trying to develop guidance specifically for those within a particular government, or those with specialized regulatory compliance requirements. For example, we consider out-of-scope issues like compliance with the NIST Cybersecurity Framework and NIST Privacy Framework (including NIST SP 800-53) or the US Department of Defense Cybersecurity Maturity Model Certification (CMMC).

One option is to take additional steps to countering cheating. As discussed below, the "fundamentals" course is currently focused on a simple "certificate of completion" which can be more easily circumvented, e.g., by someone taking the tests for someone else. It's not clear it's worth establishing a more complex certification process, but it might be. ISC2, a partner of OpenSSF, has experience in this, as does LF Training & Certification.

This document presents a proposed *mimimal* set of educational efforts for the OpenSSF; others are welcome. At the time of this writing, there is a discussion in the OpenSSF Education SIG about also defining minimum requirements for secure software development education (e.g., in colleges and universities). We also intend to discuss with various OpenSSF members how to encourage the use of OpenSSF educational materials within their organizations. These additional efforts are also welcome.

Given the above, here are steps that should be taken (including some already in process):

### 1. IMPROVE FUNDAMENTALS COURSE.

a. Analyzed 2023 feedback on fundamentals course: Here.

b. Add multimedia (e.g., video clips, animation, etc.)

c. Add hands-on labs.

   i. Investigate how to implement optional labs. See hands-on labs.

   ii. Create a few sample labs

   iii. Fill out labs. We'll try to get volunteers to help once we have a working template and samples, as this can be "embarrassingly parallel"

### 2. INCREASE AWARENESS OF FUNDAMENTALS COURSE AND EDUCATION NEEDS MORE GENERALLY.

a. Investigate Google ads costs.

b. Regularize the name of the fundamentals course. It has many different names, which can be confusing. "Secure Software Development Fundamentals Courses" vs ""Developing Secure Software" (LFD121)" vs. others. Adding the term "fundamentals" or making some other change may be important, so we can distinguish between it & various focused courses.

c. Blog posts on OpenSSF & Linux Foundation pointing to educational materials

d. Give talk at OSS NA on educational material/ progress if accepted (talk proposed)

e. Creating a set of "ambassadors" who would present basics about developing secure software (e.g., as a tutorial) at conferences in their local area (eliminating training costs). We could provide them with presentation materials, e.g., OpenSSF's "A Brief Introduction to Developing Secure Software". These would act as a stepping stone to the full course.

f. Try to convince OpenSSF members to add the course to their recommendations & their LMSs (they can add it to their LMS systems with SCORM Connect)

g. Work with DEI WG to find ways to "get the word out" without lots of money.

### 3. IDENTIFY IMPROVEMENT OPPORTUNITIES FOR SECURITY EDUCATION

a. LF Research Survey. We have begun working with LF Research to develop a survey to determine what people believe is needed (we intend to categorize people since different groups may perceive different needs).

b. OpenSSF Governing Board (GB) informal survey. In particular, if they aren't using our fundamentals course, why not? What else do they need in security education related to software development?

### 4. REWARD/INCENTIVIZE DEVELOPERS (INCLUDING MAINTAINERS) TO LEARN HOW TO DEVELOP SECURE SOFTWARE

a. Identify mechanisms to automatically determine developer knowledge of fundamentals - create simple documented API to determine if a project has at least 1 maintainer with evidence of this knowledge.

   i. To be potentially used by Scorecard and/or Best Practices Badge; See Scorecard issue #3534. It might not be scored by Scorecard.

   ii. E.g., LFD121 gives a Credly badge for completion (example), need to find mapping from GitHub id/ GitLab id/email address

   iii. Determining this information involves personal information about people. We have raised how to do this in a privacy-preserving way to LF legal and privacy experts.

### 5. DEVELOP/RELEASE MANAGER COURSE.

a. Current plan is to build on Intel's work (slide deck)

b. Education SIG to review (once released), if okay, convert to OpenSSF template & make adjustments (with credit)

c. Should this become an online course? Video? This is TBD; the current plan is to see what Intel can release before making those decisions.

### 6. IDENTIFY A SEQUENCE OF FOLLOW-ON COURSES & BEGIN EXECUTION OF THOSE

a. Start by identifying LF T&C existing & planned courses

b. Work with partners, e.g., ISC2, to identify others

c. Update fundamentals course to point to those follow-on materials as possible next steps

d. Later on, possibly create/extend these courses

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

Here we discuss some existing courses/materials in more detail. This section is equivalent to a quick version of the "Collect and Curate Content" step in the original plan. If we can use one or refer to one of them, that's wonderful. If not, they may serve as inspiration. For our purposes we're focused on courses not books. We're omitting retired/inactive courses as well as those not related to software development.

We note some general lists of such materials, then discuss some that appear to important or inspirational in some way. Our goal is not to make the perfect complete list; we don't have time to do that. Our goal is to help us identify enough materials so we can focus on what to do in the near term, reusing materials or learning lessons where we can.

## Lists of course materials

The OpenSSF education SIG's "Educational Materials Matrix for Developing More Secure Software - THE SPREADSHEET" lists many educational materials, especially freely-available ones. Although it's incomplete, so many options are listed that triage is needed. Many resources are listed as "Static Guide / Documentation"; we'll exclude them as being directly relevant, as we want courses with at least some interactivity (like quizzes or exercises). They are, of course, useful source materials.

Here are some other lists:

1. Coursera isn't really a course, but a platform for others to publish courses. https://www.coursera.org/courses?query=software%20security
2. https://www.g2.com/categories/secure-code-training
3. We have put together a short compendium on security training available on the LF Training Platform, including those from OpenSSF: https://docs.google.com/document/d/1GZqtm90nj14CrZQlblZIHTUbnaE3TUZETkC2IZknVr4/edit#heading=h.uy1j1kt9p485
4. Google sponsored links. We can't evaluate everything, but sponsored links on Google indicate organizations who are paying for people to find them, so they seemed worth investigating. A Google search for "courses on developing secure software" included these sponsored links that were especially relevant:
    1. Security Journey's Secure Application Development Training for Developers and Everyone in the SDLC. https://www.securityjourney.com/secure-coding-training-pricing
    2. Cydrill - https://cydrill.com/ - has online or instructor-led training, for developers and testers
    3. IBM's "Application Security for Developers and DevOps Professionals" on Coursera https://www.coursera.org/learn/application-security-and-monitoring

Below is a brief discussion of each.

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

## OpenSSF Fundamentals Course

The OpenSSF has released a free self-paced course, Secure Software Development Fundamentals, available via <https://openssf.org/training/courses/>. This is intended for software developers and covers the fundamentals. It's available on the Linux Foundation's Training & Certification platform as well as on edX. It's free and covers many topics. A known weakness is that it (currently) lacks optional exercises. Adding an optional set of exercises would substantially help it. Also note that it does not focus on management, operations, or deeply into specific ecosystems.

## OpenSSF "A Brief Introduction to Developing Secure Software" slide deck

OpenSSF's "A Brief Introduction to Developing Secure Software" is a short presentation summarizing the fundamentals course, intended for 30-40 minutes. It requires a presenter and doesn't provide enough by itself to provide listeners the necessary fundamentals.

It's best considered a "teaser" that provides some useful information to developers and encourages them to take the full fundamentals course. We envision some OpenSSF "ambassadors" presenting it at various conferences (e.g., conferences local to them). This would give software developers an idea of what they need to know and make them aware of the full free course.

## OpenSSF - other courses

The OpenSSF offers courses on specialized topics, as listed at <https://openssf.org/training/>:

- Securing Projects with OpenSSF Scorecard Course: "Securing Projects with OpenSSF Scorecard (LFEL1006) is available on the Linux Foundation Training & Certification platform and is designed with end users of Scorecard tooling in mind. This course will cover how to integrate the OpenSSF Scorecard into your software development life cycle."
- Securing Your Software Supply Chain with Sigstore Course: Securing Your Software Supply Chain with Sigstore (LFS182x) is available on the Linux Foundation Training & Certification platform and is designed with end users of Sigstore tooling in mind. "Building and distributing software that is secure throughout its entire lifecycle can be challenging, leaving many projects unprepared to build securely by default. Attacks and vulnerabilities can emerge at any step of the chain, from writing to packaging and distributing software to end users. Sigstore is one of several innovative technologies that have emerged to improve the integrity of the software supply chain, reducing the friction developers face in implementing security within their daily work."

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

## SAFECode courses

SAFECode is at <https://safecode.org/>. SAFECode was established in 2007 and provides a number of helpful freely-available education materials, including videos. In several cases there's a sequence of videos with optional quiz questions. These materials have been influential, as many have been available for some time and are of high quality.

Their videos tend to be of high quality when released, but are sometimes not well-maintained and are sometimes completely passive. For example, their "Secure Java Programming 101" video stresses that "Java mobile code" is different, yet in practice mobile code is almost never used today (that role has long been eclipsed by JavaScript). In this case (as in some other cases) it's also simply a video to watch; there are no quizzes, exercises, or a test at the end to interactively encourage learning or to verify learning occurred. Some SAFECode materials do have quizzes.

We believe a fundamental problem is that videos are very hard to edit and do peer review as a group, as well as being hard to update. For material that needs to be maintained over time, like this, it's often more sensible to emphasize creating text that's easier to update. Having sections with video of material that's unlikely to change can be useful, but video-only material is harder to update. If people want audio/video, it might be useful to further investigate using or integrating modern automated readers (possibly with AI). Note that people can also use screen readers.

We also believe this shows it's important to have quizzes or other interactive materials, and not just passive videos. In cases where SAFECode has no quizzes or any other interactions, it's too easy to turn on a video but not learn from it. SAFECode has some solid information, and has been a key mechanism for getting information to some developers. We're grateful for their efforts and hope to build on their examples.

The full catalog is at <https://safecode.org/training/>. Perhaps the most general is "Basic Practices for Secure Development of Cloud Applications – Part 1 and 2" - but oddly, there's no course for developing secure software in general. The "Security Development Lifecycle 101" is focused more on creating a lifecycle, not on what to do (e.g., for implementation). "System Hardening 101" can be useful, but isn't enough for developing secure software. The SAFECode catalog also includes a variety of helpful but highly specialized lessons on specific topics. As noted above, in many cases you can choose the versions with quizzes (we would recommend preferring the ones with quizzes where available).

It would be good to walk through the SAFECode courses and see which ones should be linked to from the OpenSSF fundamentals course (and then do so).

## ISC2

ISC2 is best known for its Certified Information Systems Security Professional (CISSP). Note that the CISSP has a different scope than is in view here. The purpose of the CISSP is to determine if an individual can "effectively design, implement and manage a best-in-class cybersecurity program" - and in practice, it focuses mostly on management and operations. Knowing some basics of developing secure software is a small part of it, but not its focus. It's not really focused on software developers. https://www.isc2.org/certifications/cissp

ISC2 Certified Secure Software Lifecycle Professional (CSSLP) <https://www.isc2.org/certifications/csslp> focuses more on how to develop secure software. Taking the test costs $599 in the US. In practice, many will choose to take education courses first, which costs more <https://www.isc2.org/register-for-exam/isc2-exam-pricing>. It can be divided into 8 domains:

- **Domain 1.** Secure Software Concepts
- **Domain 2.** Secure Software Lifecycle Management
- **Domain 3.** Secure Software Requirements
- **Domain 4.** Secure Software Architecture and Design
- **Domain 5.** Secure Software Implementation
- **Domain 6.** Secure Software Testing
- **Domain 7.** Secure Software Deployment, Operations, Maintenance
- **Domain 8.** Secure Software Supply Chain

Others report CSSLP tends to focus more on managing secure software development, and in general asks about things at a high level. For example, Jayton Birch emphasized that its focus was not on the technical knowledge required by software developers to implement secure software and Alexandr Fadeev emphasized that you had to think like a manager, not like a developer. This suggests it's widely perceived as being in-depth information for managers of software developers who need deeper process knowledge (with a certificate to prove it), not for software developers per se.

OpenSSF / ISC2 have announced in 2023 that we plan to collaborate:

https://openssf.org/press-release/2023/11/02/linux-foundation-isc2-and-openssf-collaborate-to-target-secure-code-development/ and we look forward to it!

## Securityjourney.com

Securityjourney.com includes Secure Application Development Training for Developers and Everyone in the software development lifecycle (SDLC). See https://info.securityjourney.com/secure-code-training-1. Its pricing varies, e.g., 5 Users for a 3 Year Term $2,750 / year (over $900/user if purchased this way).

They describe their approach as follows:

- **"Offensive & Defensive Approach** - Hands-on training allows developers to break applications to simulate an attacker's actions and then fix what they broke, all in the same lesson.
- **Accountability with Code Fixes** - Responsive developer training plans that integrate with your existing AppSec testing tools to identify and address vulnerabilities in your own code.
- **Live Assignments in Web-Based Sandbox** - Hands-on experiment engines provide real-world scenarios that allow developers to exploit, fix, and compete.
- **Custom, Programmatic Approach** - Customizable learning paths based on your organization's unique opportunities for improvement."

Their approach is a set of lessons (over 800) grouped by "learning path". They define many "learning paths"; users select the learning path to use. There are two kinds of learning paths, with many specific paths:

- **Role-Based Learning Paths:**
  - Business Learner
  - Web Developer (Back-End)

- Web Developer (Front-End)
- Native Developer
- Mobile Developer (iOS)
- Mobile Developer (Android)
- Data Scientist
- Tester
- DevSecOps
- Cloud Engineer
- Privacy Engineer

- **Compliance-Based Learning Paths:**
  - OWASP Learning Path
  - PCI Learning Path
  - Executive Order Learning Path

Their library of materials focused on specific programming languages & frameworks, as well as for different app/technologies.

One participant (David Russo) reports it's "non-free but good".

## Cydrill

Cydrill states that its training program "equips your developers with the secure coding skills they need to beat hackers at their own game." They have pitches for 3 roles: business leader, talent manager, and software developer.

As of 2024-01-09 they have a set of 37 courses. They have a search system allowing you to select your preferences to narrow your choices down to specific courses.

They have two delivery mechanisms:

- e-Learning (Online)
- Instructor-led (On-site or online)

They also divide up courses by:

- Audience (Developer, Tester)
- Subject (C, C#, C++, Java, Node, Python)
- Platform (ARM, Cloud, Desktop, Web)
- Special topics (Automotive, Banking & Finance, Healthcare, Machine Learning, Medical Devices, Network Security, PCI DSS)

For example, "Web Application Security" is a 3-day class for 12 participants, focusing on Java. It is instructor-led and intended for developers (as it is hands-on). It has 26 labs and 13 case studies. Pricing is 2250 EUR / person. Its outline is:

- Cyber security basics
- The OWASP Top Ten 2021
- Wrap up

Examples of courses include:

- Desktop application development in Java
- Security Testing Python Web Applications
- Secure coding in C and C++ for medical devices
- Cloud Application Security in Python for AWS
- Cloud Application Security in C# for Azure

In many cases they have similar courses swapping out one technology for another. This provides focused information, but it's relatively short-lived since it's focused on only one technology at a time without a lot of technology-independent context.

Cydril has a small ebook "People over tools: the key to real software security" that has a nice tagline: "Fixing security flaws is great. Preventing them in the first place is even better." The ebook states, "While DevSecOps represents a move in the right direction, adding security steps to testing and operations means that only the last stages of the software development lifecycle are covered: what about earlier phases?" Sadly, I think they're right that many "DevSecOps" implementations merely add some steps in testing & operations, instead of really putting security into all processes. They also state, "Ideally secure coding would be part of the curriculum of every programming course, but we are not there yet (and likely won't be, for a long time). The only way to combat this crucial blindspot is to train developers in secure coding best Practices – both in how to deal with the most common vulnerability types, and also in how to apply defensive programming techniques to write resilient code with proper input validation."

## Linux Foundation Security Workshops (SKF-based courses)

The Linux Foundation Security Workshops are really 3 courses:

- **Understanding Vulnerabilities and Security Threats (WSKF603).**
  - This is a 1-day workshop to "Understand the OWASP® Top 10 Security Threats"
- **Securing Coding Fundamentals (WSKF601)**
  - This is a 3-day workshop.
- **Advanced Secure Coding (WSKF602)**
  - This is a set of additional labs

These are all instructor-led courses, as opposed to being self-paced. In practice these are usually given in person. These workshops were originally conceived to support community colleges and similar situations, based on courses originally taught by Glenn van Tate. These focus on many labs, using the SKF framework. Instructors include Glenn van Tate (Europe) and Randall T. Vasquez (US).

Pricing/availability is not publicly listed at this time.

## IBM's "Application Security for Developers and DevOps Professionals" (Coursera)

The IBM course "Application Security for Developers and DevOps Professionals" is available via Coursera and was developed by John Rofrano. It's stated that it's approximately 17 hours and cost-free, making it similar in time length to OpenSSF's fundamentals course. Learner reviews generally ranked it highly.

It includes hands-on exercises and a final project, which is an advantage. It requires knowledge of how to program in Python and uses Python-specific exercises, which may be a challenge for those who don't use Python. It focuses on the OWASP top 10, and ignores the CWE top 25 (so many top vulnerabilities and mitigations are never discussed in this material). There doesn't seem to be much focus on designing for security (based on the outline). One potential risk is that it appears to be so focused on specific technologies (e.g., OpenSSL and Python) that users of other technologies may not learn the fundamentals of what they need to know.

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

It has 4 modules:

- **Introduction to Security for Application Development -** "In this module, you will identify how security fits into your workflow and gain a working knowledge of security concepts and terminology. You'll discover how to design for security in the Software Development Lifecycle (SDLC) and find out about a set of practices known as DevSecOps. You will also discover the OSI model, identify the necessary OSI layers for developers, and implement security measures on the four layers of application development. You will gain insights into security patterns and learn how to organize them. You will describe TLS (Transport Layer Security) and SSL (Secure Sockets Layer), identify how to keep TLS secure in the SDLC, and explore OpenSSL and its purpose. You will learn the strategies, best practices, and methodologies for getting security early into your code to protect applications against threats and vulnerabilities. Further, you'll find out how you can use tools like vulnerability scanners and threat models to mitigate security vulnerabilities. You'll also get the opportunity to add key terms like authentication, encryption, and integrity to your security vocabulary. Finally, you will also perform hands-on labs to encrypt and decrypt files using OpenSSL and scan a network environment with Nmap."
- **Security Testing and Mitigation Strategies -** "In this module, you will learn the key mitigation strategies to secure your application throughout development and production. You will also discover a range of security testing methods like static analysis, dynamic analysis, vulnerability analysis, software component analysis, and continuous security analysis. You will explore ways to perform code review and ensure runtime protection for application development. You will also perform hands-on labs based on static analysis, dynamic analysis, vulnerability scanning, and vulnerability detection."
- **OWASP Application Security Risks -** "In this module, you will learn about the Open Web Application Security Project (OWASP) and its Top 10 security concerns. You'll learn about application vulnerabilities and discover the top vulnerabilities concerning security experts and professionals. You will explore SQL injection, cross-site scripting, and storing secrets securely. You will also investigate software and data integrity failures, discover how to detect these types of vulnerabilities, and examine ways to mitigate their impact. You will also perform hands-on labs to analyze your code repository using Snyk and use the Vault Python API (hvac) to read, write, and delete key-value secrets in Vault."
- **Security Best Practices, Final Project, and Assessment -** "In this module, you will learn about coding best practices and software dependencies. You'll also explore how to secure a development environment by deciding what to store in a centralized repository and what not to store in GitHub. You will also perform hands-on labs to create HTTP security headers using flask-talisman and safely store and retrieve secrets using the pass CLI (command-line-interface). As your final project, you will check your code on GitHub for vulnerabilities in order of severity and fix the vulnerabilities. You'll apply the best practices for reducing the risk of vulnerability."

## IBM: Application Security for Developers (edX)

The IBM course "Application Security for Developers" is available via edX and was developed by John Rofrano. It's cost-free. It's stated that it's approximately 45 hours (5 weeks at 8–10 hours per week), about three times the estimate of the similarly-named Coursera course. From its external literature it appears to be a longer version of the Coursera course, as they have the same outline and seem to cover similar material (if they're the same it's not clear why the Coursera time estimate is 1/3 that of edX).

Its pros and cons are similar. The big pro are hands-on labs. The big negative is it only discusses the OWASP top 10, so many widespread top vulnerabilities are not covered at all. There doesn't seem to be much focus on designing for security (based on the outline).

Here is its syllabus:

- **Module 1 -** Introduction to Security for Application Development
  - Security By Design
  - What is DevSecOps
  - Vulnerability Scanning and Threat Modeling
  - Threat Monitoring
  - Activity: Security Concepts and Terminology
- **Module 2 -** Security Testing and Mitigation Strategies
  - Introduction to Security Testing and Mitigation Strategies
  - Static Analysis
  - Hands-on Lab: Using Static Analysis
  - Dynamic Analysis
  - Hands-on Lab: Using Dynamic Analysis
  - Code Review
  - Vulnerability Analysis
  - Hands-on Lab: Evaluating Vulnerability Analysis
  - Runtime Protection
  - Software Component Analysis
  - Hands-on Lab: Evaluate Software Component Analysis
  - Continuous Security Analysis
- **Module 3 -** OWASP Application Security Risks
  - Intro to OWASP (Top 10) Sec Vulnerabilities
  - OWASP Top 1-3
  - OWASP Top 4-6
  - OWASP Top 7-10
  - SQL Injections
  - Other Types of SQL Injection Attacks
  - Hands-on Lab: Understanding SQL Injections
  - Cross Site Scripting
  - Hands-on Lab: Cross Site Scripting
  - Storing Secrets Securely
  - Hands-on Lab: Storing Secrets Securely
- **Module 4 -** Security Best Practices
  - Code Practices
  - Hands-on Lab: Code Practices
  - Dependencies
  - Hands-on Lab: Dependencies
  - Secure Development Environment
  - Hands-on Lab: Secure Development Environment
- **Module 5 -** Final Exam

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

## Implementing DevSecOps (LFS262)

"Implementing DevSecOps" (LFS262) is an LF course for $299. It focuses on how to "implement DevSecOps practices into the software delivery pipeline using open source software". "To perform the hands-on lab exercises in this course, learners will need internet access, a web browser, Git, and a cloud provider account (e.g., Google Cloud Platform or AWS)."

This course implicitly assumes knowledge about secure software development fundamentals. It's focused on how to integrate security-related tools into a pipeline, not on how to develop secure software. Tools can't cause software to be designed securely, and tools have many false positives and false negatives. This is fine for a developer who knows the fundamentals, but a developer would still have to design software to be secure and would have to understand the tool results to be able to use them.

Outline:

- **Chapter 1.** Course Introduction
- **Chapter 2.** What Is DevSecOps?
- **Chapter 3.** Setting Up the Lab Environment
- **Chapter 4.** Building a DevOps Pipeline
- **Chapter 5.** Securing the Supply Chain with SCA
- **Chapter 6.** Static Application Security Testing (SAST)
- **Chapter 7.** Auditing Container Images
- **Chapter 8.** Secure Deployment and Dynamic Application Security Testing (DAST)
- **Chapter 9.** System Security Auditing with IAC
- **Chapter 10.** Securing Kubernetes Deployments
- **Chapter 11.** Secrets Management with Vault
- **Chapter 12.** Runtime Security Monitoring and Remediation

We should ensure that the fundamental course points to LFS262 as added material on how to implement a pipeline.

## Roadmaps

"Roadmaps" is an interesting interactive visual viewing system. It shows the structure of a set of learning materials (a "roadmap"). You can log in and click on various items to read more. To see its roadmap for cyber security, see: https://roadmap.sh/cyber-security. A similar view could be created for developing secure software.

Its source material is on GitHub: https://github.com/kamranahmedse/developer-roadmap

It's sometimes incorrectly called "open source" but it appears to be "open view" instead. See the license here: https://github.com/kamranahmedse/developer-roadmap/blob/master/license

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

## Tiny Courses

The OpenSSF education SIG has previously discussed the idea of making tiny courses (e.g., express learning courses) that are very narrowly focused (e.g., one specific process, a specific vulnerability for a specific ecosystem, etc.). Such small courses take fewer resources to make. An example are the "Express Learning" courses. In some situations they can be effective.

However, Timothy Serewicz (Director, Training Program, LF Training & Certification) identified an important problem with "tiny courses": when courses are atomized and developed this way in isolation, it's often hard for learners to find them, and they also tend to not be well-integrated. Any course must be subdivided into many smaller lessons, but it's important to have a larger unit that learners can select, or the time to find and select the course is too long and their sequence won't make sense.

This suggests that while we should divide lessons into easily-learned units, they should be considered as a larger construct, not just created in isolation.

## OWASP WebGoat

OWASP WebGoat <https://owasp.org/www-project-webgoat/> "is a deliberately insecure application that allows interested developers just like you to test vulnerabilities commonly found in Java-based applications that use common and popular open source components." It has a short set of lessons based on the OWASP Top 10. This can support learning, but it's hard to argue it's a full course itself.

## GMU Design & Implementation of Secure Software (SWE/ISA 681)

David A. Wheeler teaches the course "Design & Implementation of Secure Software" (SWE/ISA 681) at George Mason University (GMU) as an in-person graduate course. As a graduate course, it goes into more detail (e.g., exactly what happens on buffer overflows to the underlying platform, many readings of original material, etc.). If someone is going to be a graduate student learning more in depth about this topic, taking a graduate course can be helpful. However, some of the details aren't necessary for undergraduates. A simplified version of its most important parts were used as a starting point for the fundamentals course.

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

## Introduction to Cyber Security Specialization

The Coursera course "Introduction to Cyber Security Specialization" by Dr. Edward G. Amoroso is a high-level survey of a specialization in cyber security. It does not address any specifics of developing software, but rather on learning about foundational areas for those that might specialize in cyber security.

Abstract is: "Introduction to Cyber Security was designed to help learners develop a deeper understanding of modern information and system protection technology and methods. The learning outcome is simple: We hope learners will develop a lifelong passion and appreciation for cyber security, which we are certain will help in future endeavors. Students, developers, managers, engineers, and even private citizens will benefit from this learning experience. Special customized interviews with industry partners were included to help connect the cyber security concepts to live business experiences.

It's essentially 4 courses:

- Introduction to Cyber Attacks
- Cyber Attack Countermeasures
- Real-Time Cyber Threat Detection and Mitigation
- Enterprise and Infrastructure Security

## IT Security: Defense against the digital dark arts

The Coursera course "IT Security: Defense against the digital dark arts" is focused on general security concepts and applying them in operations. The target audience are those in IT support. It does not significantly cover software development, and thus is not really in scope for the kinds of educational materials we are considering.

Abstract: "This course covers a wide variety of IT security concepts, tools, and best practices. It introduces threats and attacks and the many ways they can show up. We'll give you some background of encryption algorithms and how they're used to safeguard data. Then, we'll dive into the three As of information security: authentication, authorization, and accounting."

- Understanding Security Threats
- (Cryptology)
- The 3 A's of Cybersecurity: authentication, authorization, accounting
- Securing your networks
- Defense in depth
- Creating a company culture for security
- Prepare for jobs in IT support

## SANS SEC275: Foundations: Computers, Technology, & Security

SANS' SEC275: Foundations: Computers, Technology, & Security course assumes the user knows little about computers, and focuses on teaching about computer technology, both abstractly and then drilling down into specific examples such as Linux, Python, and C. It covers a lot of material existing developers don't need (e.g., computer technology & how to program), while not covering material that's critical to developing secure software (e.g., it doesn't cover much of the OWASP top 10 nor the CWE top 25).

It starts by teaching system architectures, what is an operating system, what are containers, and then and how to use the Linux operating system command line specifically. It even teaches some basics about how to program. Its final section discusses more about security, including buffer overflows and format strings, as well as cryptography.

It does provide an introduction "starting from nothing" to computer concepts. However, it doesn't cover most concepts in developing secure software. The syllabus doesn't discuss secure design principles. While buffer overflows are covered, there's no attempt to identify nor cover all the most common kinds of vulnerabilities as identified by the OWASP Top 10 and the CWE Top 25.

Its overall syllabus is:

- **SEC275.1:** System Architecture, Operating System, and Linux
- **SEC275.2:** Search, Web, and Networking
- **SEC275.3:** Introduction to Servers and Programming
- **SEC275.4:** Security Concepts and Advanced Security Concepts

SANS reports that the "course content can be completed in 50 to 60 hours, but… Most students review course content multiple times, repeat labs and quizzes, or do the extra exercises and the average completion time is 120 to 140 hours."

It uses an online labs system with a Linux command line and IDE, with >90 labs. All can be accessed from a web browser.

The web-based course is $3,020; a separate certification is $380.

Note that SANS' "cyber aces" material has been retired, so it's not discussed here.

## SANS Security Awareness Developer Training Program

SANS "SANS Security Awareness Developer Training Program / Web Application Security Awareness Training" - there are two different names, but these appear to be the same materials with two different names. As implied by its title, it's awareness training, it's not intended to be enough by itself to enable developers to create secure software. However, it has a little more depth than you might expect for "awareness" training. It emphasizes the OWASP top 10 2021, but it does discuss buffer overflow and mobile application security.

Its datasheet explains what it covers (times rounded to nearest minute):

- OWASP Top 10 2021. They have a 49 min and 62 min verison.
- Mobile Application Security (38 min)
- Applied Interactive Modules for Web Application Security (7 min for each language. Supported languages are Node.js (JavaScript), C#, Java, Python, and PHP)
- Secure Coding Principles
  - Top design flaws (24 min)
  - Modern Approaches (Using Full Stacks, Using APIs, Cloud Developer) (17 min)
  - Threat Awareness (including Threat Modeling) (29 min)
  - Classic Issues (including Buffer Overflow) (32 min)
  - Software Development Lifecycle (SDLC) (Waterfall, Agile, DevOps) (34 min)

It notes that "Developer training from SANS Security Awareness includes over 40 modules that cover five of the most popular coding languages using both traditional and interactive trainings."

Prices do not appear to be publicly available. However, the Center for Internet Security has a "deep discount" for its members for SANS Security Awareness Developer Training. As of 2024-01-16 it noted, "The next deep-discount purchasing window for SANS Developer Training is from December 1, 2022 through January 31, 2023… $2,890 minimum order for 1 year of training for up to 10 users; $289 per user after that; $5,780 minimum order for 2 years of training for up to 10 users; $578 per user after that." It's likely individuals would pay much more, but other organizations might be able to negotiate similar discounts.

## SANS: Other materials

SANS has many materials. Examples include:

- Eric Johnson's 2015 blog post "Securing the Software Development Lifecycle" summarizes secure development and says "Learn more and sign up for a free demo at …" but the domain securingthehuman.org is no longer active.
- SANS paper "Secure Software Development and Code Analysis Tools" (2002) by Thien La. It has two parts, "Secure Programming Guidelines" (focusing on Perl, Java, and C/C++) and "Source Code Analysis Tools". This is a paper, so it's out of scope for our list of courses, and it's over 20 years old as well.

## Synk Learn - Security for Developers

"Synk Learn" has a number of lessons and learning paths; the "Security for Developers" learning path is the most relevant so we'll focus on that here. It's a short set of lessons focusing on countering implementation vulnerabilities identified by the OWASP Top 10, as well as a single lesson on secure design. Threat modeling is mentioned (in the insecure design lesson) but not covered. Broader context (requirements & risk management) and verification are not covered. Sample JavaScript code is given.

This learning path is a specific collection of lessons, as follows:

- Broken access control
- Insecure hash
- Insecure Randomness
- Cross-site scripting (XSS)
- Code injection
- Cross site request forgery (CSRF)
- Prototype pollution
- NoSQL injection attack
- SQL injection (SQLi)
- XML external entity injection (XXE)
- XPath injection
- Insecure design
- Vulnerable and outdated components
- Directory traversal
- Logging vulnerabilities
- Server-side request forgery (SSRF)

Each lesson ends with a small quiz. The JavaScript examples, in our sample, seem to be quite helpful.

# Appendix A: Current State of Educational Materials Landscape Details ("Collect and Curate")

"Security for Developers" is offered via a partnership between Snyk and New York University, Tandon School of Engineering; anyone can complete the path and download a certificate, but only NYU Tandon students can receive the industry badge.

## Member-contributed Materials

Intel has contributed a new course to the OpenSSF which is intended to educate managers of software developers. The purpose of the material, which is fundamentally a slide deck, is to help these managers understand how to manage software development to produce secure software.

The current plan is to update the deck, and use it as a basis for a short course for software managers. We currently expect it will become a set of videos, along with quiz questions and a final exam.

The group is always open to additional member contributions in this area. Member-donated material help jump-start the process of creating and augmenting material and also helps us grow the pool of contributors and collaborators towards our overall educational efforts. Most, if not all, OpenSSF members have existing security educational materials that could make good candidates for community contribution. Doing so helps those organizations lower their cost of on-going maintenance of those former internal materials by joining a cadre of community contributors that can assist with that burden.

## Potential materials

We are in discussions with organizations that might be able to share materials they've developed with either the OpenSSF or simply the world at large. We've not seen them, but we've been in long discussions with them, and hope they'll work out.

In particular, the US Navy has some materials on developing secure software, which we're told includes examples of attacks to learn from. These might be useful starting lab examples. However, at this time we have not seen them nor have they been released.

In the end, it's up to those organizations to decide if they want to receive their materials (and what process they need to confirm that). We certainly encourage contributions.

# Appendix B: Secure Software Development Lifecycle Models

Secure software development, to work well, must be embedded in the overall software development lifecycle (SDLC). Some documents, such as ISO/IEC/IEEE 12207:2017 ("Systems and software engineering: Software life cycle processes"), describe the general processes used when developing and deploying software. Some other documents specifically discuss how secure software development should be embedded in the SDLC, and are worth consulting to ensure important topics are covered. We list here a few key documents that discuss how secure software development should be embedded in the SDLC.

## NIST Special Publication 800-218

NIST Special Publication 800-218 is the Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities (February 2022). It argues that "Organizations should integrate the [Secure Software Development Framework (SSDF)] throughout their existing software development practices, express their secure software development requirements to third-party suppliers using SSDF conventions, and acquire software that meets the practices described in the SSDF." It wisely notes that "the earlier in the SDLC that security is addressed, the less effort and cost is ultimately required to achieve the same level of security. This principle, known as shifting left, is critically important [as it] minimizes any technical debt [and it can] result in software with stronger security and resiliency."

Instead of introducing new practices, it "describes a set of high-level practices based on established standards, guidance, and secure software development practice documents." This has the advantage of abstracting many different materials into a broader framework. In the document these are grouped into 4 categories:

- "Prepare the Organization (PO): Organizations should ensure that their people, processes, and technology are prepared to perform secure software development at the organization level…
- Protect the Software (PS): Organizations should protect all components of their software from tampering and unauthorized access.
- Produce Well-Secured Software (PW): Organizations should produce well-secured software with minimal security vulnerabilities in its releases.
- Respond to Vulnerabilities (RV): Organizations should identify residual vulnerabilities in their software releases and respond appropriately to address those vulnerabilities and prevent similar ones from occurring in the future."

While in theory this document is supposed to apply to any SDLC, in practice parts of this document unfortunately encourage high-risk waterfall-like approaches. E.g., the first practice is "Define Security Requirements for Software Development (PO.1)" which says "Ensure that security requirements for software development are known at all times so that they can be taken into account throughout the SDLC and duplication of effort can be minimized because the requirements information can be collected once and shared." In practice, requirements (including security requirements) are often not fully known until after the system has been disposed of. While it is wise to attempt to capture requirements, requirements are often poorly understood and change over time. Ensuring that requirements are "always" known is a hallmark of the so-called "waterfall" process; as Dr. Winston W. Royce noted in 1970, waterfall processes (such as attempting to identify all requirements before doing anything else) are in practice "risky and invite failure." Thus, applying some of these practices should be tempered. In addition, it is a summary of other documents, many much older, so newer approaches and concerns may omitted from this material.

That said, even with these limitations, this NIST document does identify important areas relevant to software development processes when considering security.

## BSIMM

Building Security In Maturity Model (BSIMM) is one of the source documents used by NIST Special Publication 800-218, but BSIMM is important enough that we list it separately. BSIMM is an analysis of a set of over 130 organizations' software security initiatives (also known as application security, product security, or DevSecOps programs). It thus provides an insight into what a number of companies have decided to do to develop secure software. While what any particular organization chooses to do might not be an effective choice, or might not be the right choice for a different organization, the result does nevertheless provide helpful insights into what organizations are doing and what is especially common.

The BSIMM software security framework is divided into 4 domains, each of which are further subdivided into 12 total practices. Those practices are further divided into activities. The domains and practices are:

1. **Governance:** Strategy & Metrics; Compliance & Policy; Training
2. **Intelligence:** Attack Models; Security Features & Design; Standards & Requirements
3. **Secure Software Development Lifecycle (SSDL) Touchpoints:** Architecture Analysis; Code Review; Security Testing
4. **Deployment:** Penetration Testing; Software Environment; Configuration Management & Vulnerability Management (CMVM)

BSIMM 2023 found that these were the most frequently observed activities (with #1 being the most common):

1. Implement security checkpoints and associated governance.
2. Create or interface with incident response.
3. Identify privacy obligations.
4. Use external penetration testers to find problems.
5. Ensure host and network security basics are in place.
6. Use automated code review tools.
7. Perform edge/boundary value condition testing during QA.
8. Perform security feature review.
9. Unify regulatory pressures.
10. Create a security portal.

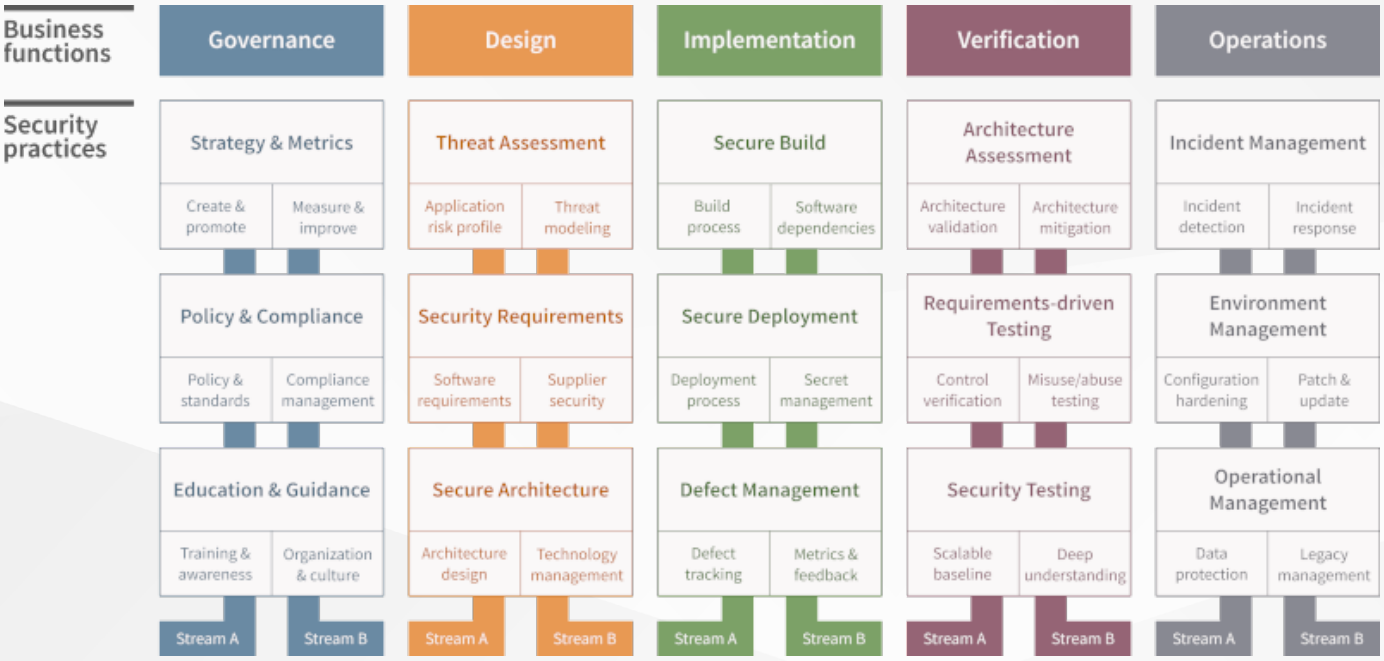
The activities with the largest growth (and thus likely to become frequently observed) were:

1. Streamline incoming responsible vulnerability disclosure.
2. Implement cloud security controls.
3. Make code review mandatory for all projects.
4. Have a research group that develops new attack methods.
5. Define secure deployment parameters and configurations.
6. Use application containers to support security goals.
7. Schedule periodic penetration tests for application coverage.
8. Identify open source.
9. Document a software compliance story.
10. Enforce security checkpoints and track exceptions.

For more information, see the BSIMM report.

## OWASP SAMM

The purpose of the OWASP [Software Assurance Maturity Model (SAMM)] is to "provide an effective and measurable way for you to analyze and improve your secure development lifecycle". It emphasizes identifying tasks specific to developing and operating secure software. The following diagram from OWASP SAMM provides a helpful visual representation:




A minor oddity of OWASP SAMM is that security requirements are labelled part of "design" in their model, instead of being considered separate as is done by practically all other models (including ISO/IEC/IEEE 15288, ISO/IEC/IEEE 12207, and many others). However, this is a minor point, as the business function "design" appears to really be about "requirements and design" (presumably this longer name didn't simply fit on the figure). A larger oddity of SAMM is that while "implementation" is identified as a business function, we have not found a security practice in SAMM that covers secure implementation of software (that is, writing source code to be secure and avoiding common types of vulnerabilities). Architecture, build, and testing are covered as security practices, but not implementation of source code. At the time of this writing we plan to use OWASP SAMM to guide a few survey questions, and supplement SAMM with a category specifically about secure implementation to address this.

## License

**David A. Wheeler**
*Director of Open Source Supply Chain Security*
*Open Source Security Foundation (OpenSSF)*

Dr. David A. Wheeler is an expert on open source software (OSS) and on developing secure software. His works on developing secure software include "Secure Programming HOWTO", the Open Source Security Foundation (OpenSSF) Secure Software Development Fundamentals Courses, and "Fully Countering Trusting Trust through Diverse Double-Compiling (DDC)". He is the Director of Open Source Supply Chain Security at the Linux Foundation and teaches a graduate course in developing secure software at George Mason University (GMU). Dr. Wheeler has a PhD in Information Technology, a Master's in Computer Science, a certificate in Information Security, a certificate in Software Engineering, and a B.S. in Electronics Engineering, all from George Mason University (GMU). He is a Certified Information Systems Security Professional (CISSP) and Senior Member of the Institute of Electrical and Electronics Engineers (IEEE). He lives in Northern Virginia.

OpenSSF

OPEN SOURCE SECURITY FOUNDATION

# Thank you!

openssf.org