SECURE SOFTWARE DEVELOPMENT LIFECYCLE

FUNDAMENTALS

Codrut Andrei

https://www.linkedin.com/in/codrut-andrei/

SUMMARY

The Secure Software Development Lifecycle Fundamentals is a brief introduction to SDLC, its methods, available resources, and it is primarily written with the developer in mind. All of the materials included in this paper are meant to assist developers in learning how to write better code, create applications that are more secure, and build a more secure tomorrow.

Developers can help the company shorten the time to market while maintaining a high degree of quality and security for their applications, by offering them training and education on secure coding and sharing information about security standards and requirements. Automating mundane chores and enhancing teamwork can all be achieved by incorporating these techniques and technologies into daily operations.

In today's market, every business depends to varying degrees on software, and any application that is accessible over the internet has a sizable attack surface that requires security.
Protecting our enterprises is a difficult undertaking since there are more zero-day vulnerabilities, more entities that profit from erroneous configurations, and a lack of administrative and technological controls.

Cross-site scripting (XSS) and SQL injections are two of the most common instances of code flaws that lead to online application vulnerabilities. Visibility of our attack surface is a positive step, but the real problem has to be solved. The greatest method to demonstrate dedication to developing security and ensuring that our clients have faith in us as a reliable business partner is to assist our developers with safe coding best practices and trainings to prevent data breaches, financial loss, and reputational harm.

About the author

With over 14 years of experience in the technology and information security industry and a career evolution from technical to management, I have designed and implemented enterprise-grade security programs and services, developed of a team of highly skilled professionals, and matured the organization information security capabilities through leadership, strategy, planning and execution.

Mentoring, coaching, raising security awareness, giving back to the community, and doing my part to make the world a better and safer place best reflect my personality, values, and aspirations.

INTRODUCTION

MORE SECURITY, LESS DELAYS

A Secure SDLC strategy is made to put security needs and procedures into effect without slowing down software development. In a market that moves more quickly than ever, speed is crucial, and when combined with quality and security, it is a surefire prescription for success for any organization.

ACROSS THE BOARD

Secure SDLC seeks to increase security from the requirements to the final product by focusing on any stage and any aspect of the process. To ensure that everyone understands the ultimate objective and works toward attaining quality and security, security criteria should be included during the planning/requirements phase.

CONTINUAL IMPROVEMENT

Speaking the same language and being aware of our procedures, tools, and structure will help our teams collaborate better both internally and externally, leading to quicker deployments and more secure solutions.

WHY SECURE SDLC?

WHAT IS SOFTWARE DEVELOPMENT LIFECYCLE?

Is a framework defining tasks performed at each step in the software development process. The lifecycle defines a methodology for improving the quality of software and the overall development process. SDLC is the structure followed by a development team within the software organization. It aims to produce quality software that exceeds customers expectations, meets deadlines and cost estimates.[1]

In software engineering, it is the process for planning, designing, developing, testing, deploying, and supporting an information system.

The development process is enhanced by the security component of Secure SDLC, which is built on top of the already-existing framework. For instance, while defining the functional requirements during the planning stage, we will also include the security needs.

BENEFITS

- Facilitates effective planning and delivery of high-quality and secure products
- It supplies an easy-to-follow framework to design and build applications
- Reduces unnecessary costs and delays
- Enables team collaboration and continuous improvement
- Adds a risk-based approach to software development

FLAVORS

- Software Development Lifecycle
- Systems Development Lifecycle
- Secure Development Lifecycle
- Secure Systems/Software Development Lifecycle
- Secure Software Development Lifecycle

Every organization has a distinct SDLC that was implemented with business objectives and needs in mind. The title is not as important as the practices followed and what it enabled the business to do.

[1] Phoenix NAP, https://phoenixnap.com/blog/software-development-life-cycle

SECURE SDLC AND APPSEC

Different approaches have been employed over time to aid in overcoming the difficulties in software development. The methods for creating quicker and more secure apps changed along with technology and the industry. The most popular and widely utilized approaches are outlined below:

Waterfall – "Is a breakdown of a project into linear sequential phases …" Wikipedia

Agile – "Agile practices include requirements discovery and solutions improvement through adaptive planning, evolutionary development, early delivery, continual improvement, and flexible responses to changes …" Wikipedia

DevOps – "Is a set of practices that combines software development and IT Operations. It aims to shorten the systems development lifecycle …" Wikipedia

DevSecOps – "Make everyone accountable for security with the objective of implementing security decisions and actions at the same scale and speed as development and operations decisions and actions" Forcepoint

Typically, there are 5 to 8 phases in a software development lifecycle. It differs from business to company and is more likely to be implemented in accordance with the demands, resources, and activities of the particular organization.

Requirements – Defining the functional and design requirements of the application and what will be necessary to do to meet those items. It may also be called the Analysis phase.

Design – Outline the design features to implement to meet the functional, security, and privacy requirements defined earlier.

Build – Application coding and code documentation. Also called the Code phase.

Test – Static & Dynamic Analysis of the code and Security Evaluation.

Release – exposing the web application for use.

While businesses have begun to shift certain security testing responsibilities earlier in the development lifecycle in recent years, there is still a gap between the start of a project and the build phase.

THE GAP

The principle of Shift-Left is to take a task that was traditionally done at a later stage, like security, and perform it earlier. While security testing was done mostly at the end of every project, with the adoption of this principle people started to push security testing and requirements earlier in the development lifecycle.

This was accomplished by incorporating Static Application Security Testing and Software Composition Analysis into the BUILD phase, and the project teams were able to find the majority of the vulnerabilities earlier, allowing them to remediate faster and with more time before deploying the web application. However, this strategy merely addresses the security testing issue rather than the root cause.

We got better at finding vulnerabilities but remediating all of them is still a challenge.

The objective should not be to acquire more tools and perform more testing, but rather to assist developers in designing secure applications from the start. As a solution to an old problem, this entails extending beyond the requirements phase of any project and investing in people, training, and education. Secure coding is not taught in schools, and there is a lack of understanding about the resources and technologies that may assist enterprises in building a community of developers who can exchange best practices and how to satisfy security standards.

Without a training platform, developers will continue to write bad code while focusing only on functionality and on-time delivery. As security experts, we can assist the community by sharing free materials and technologies that may be used, automated, and incorporated into daily routines.

Secure SDLC as a technique, along with training and education for our developers, will be a start in correcting some of the most frequent security concerns and shifting our perspectives on how we design tomorrow's apps.

ADDRESING THE GAP

Training and education for our personnel is an important aspect of any governance program. It ensures that corporate goals, strategies, and management support are all aligned in order to achieve the organization's purpose and vision.

Developer security training on secure code, best practices, and industry standards is where change begins for every software development business.

MICROSOFT SDL

Microsoft Security Development Lifecycle *consist of a set of practices that support security assurance and compliance requirements. The SDL helps developers build more secure software by reducing the number and severity of vulnerabilities in software, while reducing development cost.*

Practice #1 **Providing Training**

Security is everyone's job. Developers, service engineers, and program and product managers must understand security basics and know how to build security into software and services to make products more secure while still addressing business needs and delivering user value.

Effective training will complement and re-enforce security policies, SDL practices, standards, and requirements of software security, and be guided by insights derived through data or newly available technical capabilities.

Although security is everyone's job, it's important to remember that not everyone needs to be a security expert nor strive to become a proficient penetration tester. However, ensuring everyone understands the attacker's perspective, their goals, and the art of the possible will help capture the attention of everyone and raise the collective knowledge bar.[2]

[2] Microsoft SDL, https://www.microsoft.com/en-us/securityengineering/sdl/practices

1. Explore all Microsoft SDL Practices: https://www.microsoft.com/en-us/securityengineering/sdl/practices

2. Learn more about Cisco's Secure Development Lifecycle: https://www.cisco.com/c/dam/en_us/about/doing_business/trust-center/docs/cisco-secure-development-lifecycle.pdf

Now that we've covered the basics of SDL, it's time to go further and learn about the Three Pillars of Information Security: Confidentiality, Integrity, and Availability. These are the fundamental elements for establishing and maintaining a strong security program in any organization.

Confidentiality: safeguards meant to protect sensitive information from unwanted access.

Integrity: ensuring the consistency, correctness, and trustworthiness of data throughout its lifespan.

Availability: Information should be continuously and easily accessible to authorized parties.

Learn more about the CIA Triad:

- https://www.cybereason.com/blog/three-pillars-of-infosec-confidentiality-integrity-and-availability
- https://www.f5.com/labs/articles/education/what-is-the-cia-triad

O W A S P

The Open Web Application Security Project® (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.[3]

OWASP offers several initiatives aimed at assisting developers in writing better code, and it will be one of the most important tools for anybody who wants to learn secure coding, security best practices, and how to build Application Security Programs.

[3] OWASP, https://owasp.org/

OWASP Security Design Principles

1. Minimize attack surface area: use only the features and components that you need
2. Establish secure defaults: establish safeguards for higher access and privileges
3. The Principle of Least Privilege: grant only the minimum privileges needed to accomplish a task
4. The principle of Defense in Depth: implement multiple layers of security controls; validation, audit, logging
5. Fail securely: when application fail this should not give users additional privileges or access to sensitive data
6. Don't trust services: always validate data from third party services
7. Separation of duties: preventing individuals from the ability to perform fraudulent tasks
8. Avoid security by obscurity: implement enough security controls to keep the application safe without hiding core functionality
9. Keep security simple: keeping application architecture simple will reduce the risk of errors
10. Fix issues correctly: identify and fix the root cause[4]

You can learn more about Security by Design here:

1. https://patchstack.com/articles/security-design-principles-owasp/
2. https://www.ncsc.gov.uk/collection/cyber-security-design-principles

OWASP Proactive Controls

Software developers are the foundation of any application. To achieve secure software, developers must be supported and helped by the organization they author code for. As software developers author the code that makes up a web application, they need to embrace and practice a wide variety of secure coding techniques. All tiers of a web application, the user interface, the business logic, the controller, the database code and more – all need to be developed with security in mind. This can be a very difficult task and developers are often set up for failure. Most developers did not learn about secure coding or crypto in school. The languages and frameworks that developers use to build web applications are often lacking critical core controls or are insecure by default in some way. It is also very rare when organizations provide developers with prescriptive requirements that guide them down the path of secure software. And even when they do, there may be security flaws inherent in the requirements and designs. When it comes to software, developers are often set up to lose the security game.

The OWASP Top Ten Proactive Controls 2018 is a list of security techniques that should be included in every software development project. They are ordered by order of importance, with control number 1 being the

[4] PatchStack, https://patchstack.com/articles/security-design-principles-owasp/

most important. This document was written by developers for developers to assist those new to secure development.[5]

- C1: Define Security Requirements
- C2: Leverage Security Frameworks and Libraries
- C3: Secure Database Access
- C4: Encode and Escape Data
- C5: Validate All Inputs
- C6: Implement Digital Identity
- C7: Enforce Access Controls
- C8: Protect Data Everywhere
- C9: Implement Security Logging and Monitoring
- C10: Handle All Errors and Exceptions

OWASP Top 10:2021 Most Significant Risks

The OWASP Top 10 is a list of the top ten application vulnerabilities. It also depicts the risks, consequences, and countermeasures. Understanding the risks and the mitigation measures can help you prepare and lower the chance of being breached.

- A01 Broken Access Control
- A02 Cryptographic Failures
- A03 Injection
- A04 Insecure Design
- A05 Security Misconfiguration
- A06 Vulnerable and Outdated Components
- A07 Identification and Authentication Failures
- A08 Software and Data Integrity Failures
- A09 Security Logging and Monitoring Failures
- A10 Server Side Request Forgery (SSRF)

OWASP Code Review Guide

Code quality issues include known security defects or patterns, reusing variables for multiple purposes, exposure of sensitive information in debugging output, off-by-one errors, time of check/time of use (TOCTOU) race conditions, unsigned or signed conversion errors, use after free, and more. The hallmark of this section is that they can usually be identified with stringent compiler flags, static code analysis tools, and linter IDE plugins.[6]

[5] OWASP, Top 10 Proactive Controls 2018, https://owasp.org/www-project-proactive-controls/
[6] OWASP, Code Quality Issues, https://owasp.org/Top10/A11_2021-Next_Steps/

Learn more about Code Review here:

https://owasp.org/www-pdf-archive/OWASP_Code_Review_Guide_v2.pdf

So far, we've covered the information security pillars, concepts, secure coding approaches, the most prevalent threats, and the hacker mindset. We started with the OWASP Code Review Guide and will continue with more tools to teach developers how to construct safe apps. Next, we'll look at one of the greatest training tools available for developers, security practitioners, and businesses looking to adopt a Safe SDLC Program, learn how to write better code, or just comprehend secure coding techniques.

OWASP Security Knowledge Framework

Security Knowledge Framework is a full open-source Python-Flask / Angular web-application that uses many other great open source projects to train you and your team in building secure applications, **by design**.[7] What you will find:

Your Secure Coding Starts Journey Here:

- Code Examples: an extensive library of popular hacks, exploits, and recommended practices
- Training Platform: develop your secure coding and hacking abilities with interactive labs and learn the hacker mindset
- Knowledge Base: all requirements are correlated to knowledge base items to give you more in depth information about attack vectors, impact, mitigation, and best practices
- Checklists: Application Security Verification Standard and Mobile Application Security Verification Standard

Other Free Secure Coding Training Resources

- Kontra: https://application.security/
- Global Learning Systems: https://globallearningsystems.com/free-application-security-training/
- ShiftLeft: https://www.shiftleft.io/community-and-training/
- Snyk Learn: https://learn.snyk.io/
- SEI CERT Coding Standards: https://wiki.sei.cmu.edu/confluence/display/seccode/SEI+CERT+Coding+Standards
- OWASP Cheat Sheets: https://cheatsheetseries.owasp.org/index.html

[7] Security Knowledge Framework, https://www.securityknowledgeframework.org/

KEY TAKEAWAYS

- SDLs from Microsoft and Cisco will provide you a better grasp of best practices and a high-level perspective of the process.
- The concept of building secure apps by design will lead you in your future attempts to construct resilient applications that are simple to manage and upgrade.
- The Top 10 Risks can assist you in understanding the hacker mindset as well as the strategies used to mitigate vulnerabilities.
- Understanding current threats and countermeasures will enable any developer to become an asset in any organization by avoiding typical coding errors and learning how to code the next application more efficiently.

REQUIREMENTS PHASE

Requirements – Defining the functional and design requirements of the application and what will be necessary to do to meet those items. It may also be called the Analysis phase.

Including security criteria alongside functional and design requirements may help teams plan better and collaborate to achieve common goals. During this phase, the security teams will often disclose the rules, standards, and procedures that apply to the project. Considering security from the start of a project has huge benefits for every team and company. A checklist and other best practices might be useful for developers at this time.

The OWASP Application Security Verification Standard will be used by security testing teams to evaluate technical security measures and will give a set of standards for secure development to developers. This may be used as a guideline for any software project and can be automated and included into everyday operations to ensure that secure apps are built continually.

OWASP ASVS: https://owasp.org/www-project-application-security-verification-standard/

DESIGN PHASE

Design – Outline the design features to implement to meet the functional, security, and privacy requirements defined earlier.

A common security activity performed during the design phase is Threat Modeling. This is an effective way to secure your systems, applications, networks, and services. It's an engineering technique that identifies potential threats and recommendations to help reduce risk and meet security objectives earlier in the development lifecycle.[8]

When you perform threat modeling, you begin to recognize what can go wrong in a system. It also allows you to pinpoint design and implementation issues that require mitigation, whether it is early in or throughout the lifetime of the system. The output of the threat model, which are known as threats, informs decisions that you might make in later design, development, testing, and post-deployment phases.[9]

Free Threat Modeling Training

- Microsoft: https://docs.microsoft.com/en-us/learn/paths/tm-threat-modeling-fundamentals/
- Coursera: https://www.coursera.org/lecture/software-security/threat-modeling-or-architectural-risk-analysis-bQAoU
- GitHub: https://github.com/hysnsec/awesome-threat-modelling
- OWASP: https://owasp.org/www-project-threat-dragon/
- Threat Modeling Manifesto: https://www.threatmodelingmanifesto.org/

[8] Microsoft, Threat Modeling Security Fundamentals, https://docs.microsoft.com/en-us/learn/paths/tm-threat-modeling-fundamentals/
[9] Threat Modeling Manifesto, https://www.threatmodelingmanifesto.org/

CODE/BUILD PHASE

This is where developers work their magic; they create the future and help the business remain competitive in the market by producing high-quality and secure software.

Even with all of the precautions in place and resources available, it is human nature to make mistakes. The good news is that it is not too late to fix any code errors. For that, we offer some free materials that can be utilized by anybody and used to decrease risks and the number of potential vulnerabilities.

Free Static Code Analysis Tools

- Analysis Tools: https://analysis-tools.dev/
- SonarQube: https://www.sonarqube.org/
- Microsoft (see #Practice 9 Perform Static Analysis Security Testing): https://www.microsoft.com/en-us/securityengineering/sdl
- OWASP Source Code Analysis: https://owasp.org/www-community/Source_Code_Analysis_Tools

TEST PHASE

If you want to take care of security end-to-end, there is only one more step to verify and confirm the quality of your code. Assessing the running application for vulnerabilities can be accomplished by performing fuzzing techniques and Dynamic Application Security Testing (DAST).

This will give the look and feel of what are some of the tools and techniques that are being used both by the good and the bad guys. Understanding their mindset can help you build better defenses for your applications.

Free Testing Resources

- Fuzzing: https://github.com/secfigo/Awesome-Fuzzing
- OWASP ZAP: https://www.zaproxy.org/download/