The Basics of Secure Software Development

Content

Chapter 1 - Introduction

What did we find out?

Condition Zebra has been offering cybersecurity solutions since 2007. It's been 15 years of providing services for SMEs, companies, the government and financial sectors.

Our security engineers usually conduct Vulnerability Assessments and Penetration Testing (**VAPT**$_{30}$) for web or mobile applications. What we found is that the majority of software applications in production have one or more security vulnerabilities.

This experience has propelled us to offer a **Secure Software Development Lifecycle (SSDLC**$_{28}$**) service** for companies looking to develop software that incorporates security aspects in the development process.

The cost of fixing the vulnerabilities in older software is sometimes up to 20–100 times greater. So it makes sense to use SSDLC when developing software to save costs in the long term, avoid security vulnerabilities, and build more secure software.

SolarWinds Hack

In early 2020, the SolarWinds attack happened[1], where hackers installed malicious code into the company's Orion software. This one vulnerability in the software has a destructive effect; every company or organisation using Orion was left vulnerable, including Microsoft, Cisco, FireEye, Intel, NATO, the UK Government, European Parliament, and multiple US federal agencies.

So what went wrong in the case of the SolarWinds hack?

An answer might be traced back to a software vulnerability where hackers successfully compromised the Orion IT monitoring platform, where the attacker install **backdoor**$_{4}$ into the networks of some 40 companies.

Therefore, the practice of Secure Software Development (SSD) is more needed than ever because of the rising cyber incidents affecting software applications on a global scale.

Why design security from the start?

SSD (Secure Software Development) simply means producing software with security in mind from the ground up. By embedding security practices (such as **Static application security testing (SAST)**[24], **Dynamic application security testing (DAST)**[9], **Secure code reviews**[25], etc) at each stage of the Software Development Lifecycle (SDLC), the following benefits can be achieved:

1) Software is more secure

By focusing on security[2] throughout the entire software development lifecycle results in software that is more secure than ever.

2) Demonstrating a continuous commitment to security

The objective is to give all the stakeholders who have an interest[3] in application security the education, information, and tools they need to succeed.

This will enable all interested parties, such as project managers, development managers, application developers, server configuration, release management, quality assurance teams, and others, to become committed and aware of security considerations.

"Organizations should be aware of the destructive "domino effect" that one vulnerability can have on software across the globe in the case of SolarWinds attack".

- The 12th Annual State of Software Security (SOSS) report by Veracode

3) Identify potential flaws early in the development process

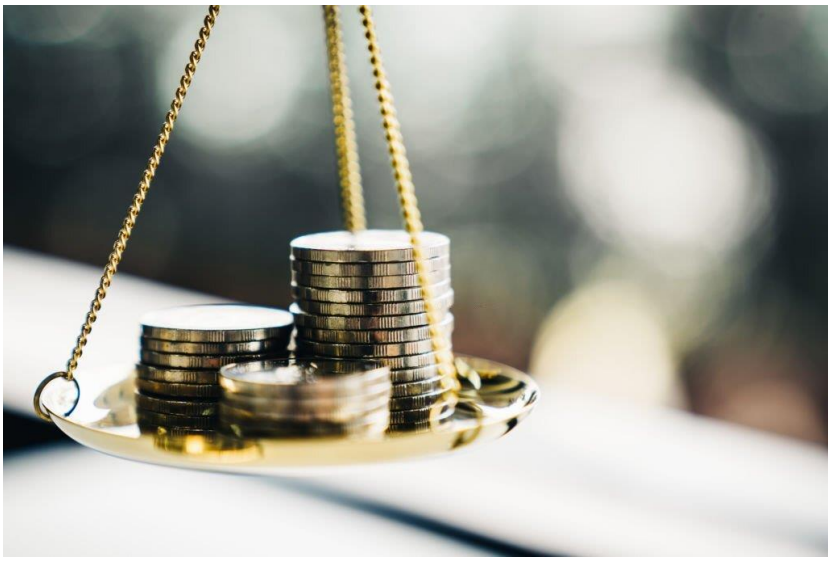
In a software development process, consider designing security from the start. This will help detect security flaws early, which will reduce business risks for the organization[2].

A few of the security activities that can help with this include:

a) Threat modelling is a step that can help to identify possible security issues early in the development stage.
b) **Secure Code Reviews**[25] are another step that can help find security flaws in the source code.
c) Security testing, including **SAST**[24] & **DAST**[9] and **Penetration Testing**[22], is critical. Both of these security activities are used to validate that the deployment is secure.

4) Reduce development costs

Fix security vulnerabilities in the initial stage than in later stages of the SDLC to reduce overall costs of software development.

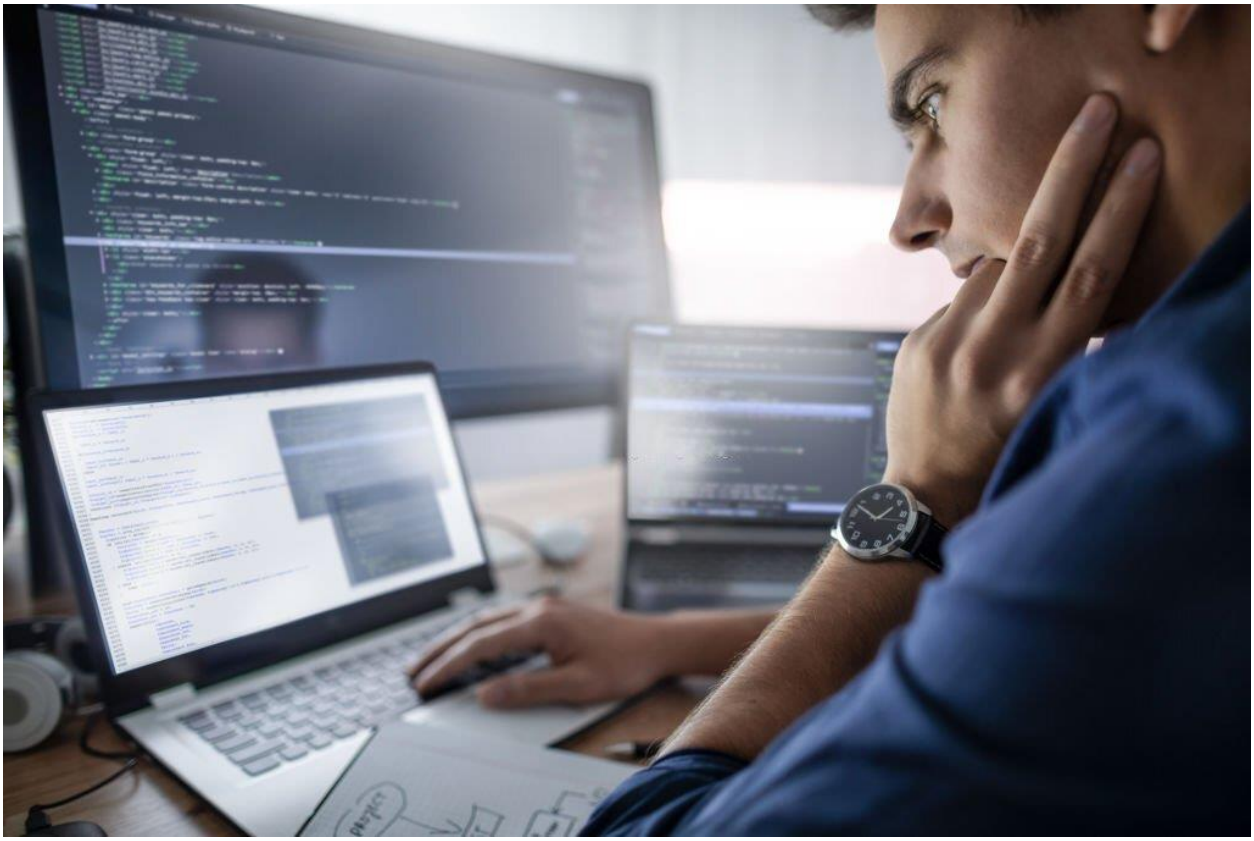


Chapter 2: Understanding SDLC and SSDLC

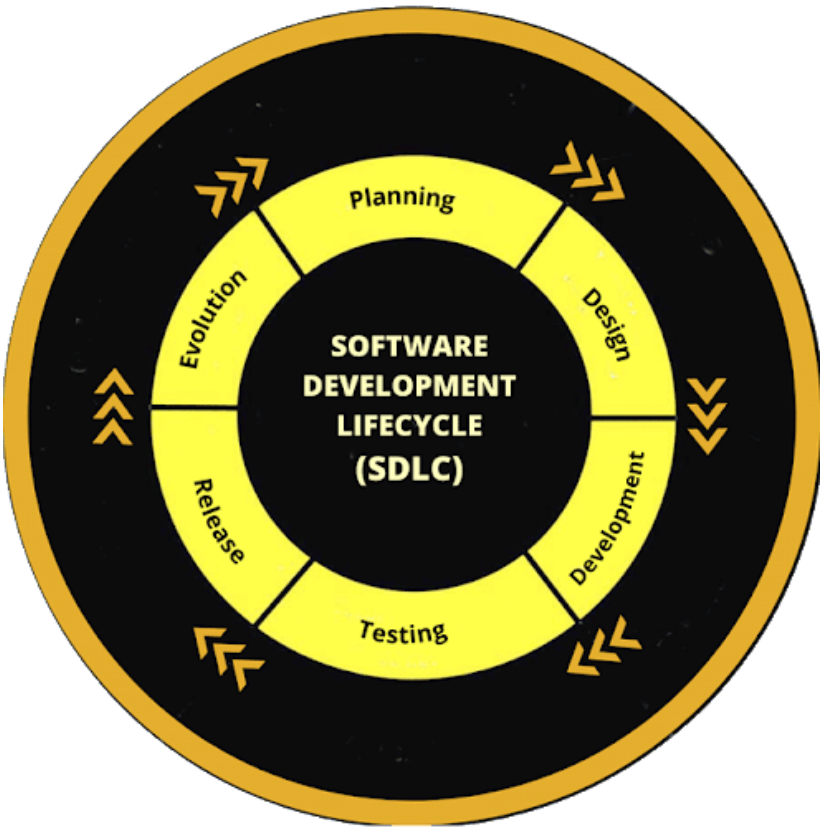
What is SDLC?

The Software Development Lifecycle (SDLC) is the usual process used by organisations to build software from start to finish. SDLC models include **waterfall**[38], **iterative**[14], and **agile**[1] processes.

"Lifecycle" means the process is continuous. Each development lifecycle phase has its own specific tasks that will be carried over into the next stage.



Below is the general model for the **SDLC**$_{30}$ process:



In a standard SDLC, below is the phases:

1) Planning: During this phase of the SDLC, various stakeholders from management, development teams, design teams, and any other relevant teams will plan, discuss, and define the goals and requirements for the software projects.

2) Design: In this phase, developers and other relevant teams will study the requirements and develop the design blueprints of the software.

3) Development: During this phase, developers begin to build the software.

4) Testing: The testing phase is done for quality assurance purposes and to ensure there are no significant code errors in the software.

5) Release: In the release phase, the software is launched to the public and can be used by users.

6) Evolution: In this phase, the developers maintain the usability of the software and provide support to fix minor issues.

Secure SDLC explained

In the planning phase of normal SDLC, there are no steps to discover and mitigate security threats. However, the only security-related task is in the testing phase, which results in finding security flaws too late or not at all[3].

Then, with time, all the parties involved in the software development process started to integrate security activities to easily detect security vulnerabilities for each phase, ensuring the end products would be less prone to errors and security risks. This ensures the quality, correctness, and security of the software product being built.

A Secure Software Development Lifecycle (SSDLC) is a methodology for building software with an emphasis on security elements in each stage of the Software Development Lifecycle (SDLC).

The SSDLC aims to produce high-quality software that is secure and meets customer expectations by using a systematic process that aims to build security at the start[11] by adopting, **"Shift-Left[26]"** mentality.

> ***"Shift-Left" mentality: In the software development lifecycle, this refers to moving towards the planning and design phase at the earliest stages.***

The Secure SDLC has frameworks and standards[2] based on **OWASP**[18] that can aid the secure software development process, such as **IS0 27034**[13], **BSIMM**[3] (Building Security in Maturity Model), and **OWASP SAMM**[19] (Software Assurance Maturity Model by OWASP).

The Open Web Application Security Project (OWASP) is an international non-profit organization dedicated to web application security.

OWASP SAMM

Software developers may use OWASP SAMM (Software Assurance Maturity Model)[4] to analyze and improve the current secure development lifecycle process.

There are 5 phases inside SAMM, with 3 security practices for each phase, which makes up 15 security practices that are helpful for the SSDLC process.

1) Governance: focuses on the processes and activities that an organisation uses to manage software development. This includes cross-functional groups involved in the development and business processes established at the organisation level.

- Strategy & Metrics
- Policy & Compliance
- Education & Guidance

2) Design: focuses on the processes and activities related to the design aspects of creating software in a development project. This usually includes requirements gathering, high-level architecture specification, detailed design, identifying and mapping threat vectors and protections in place.

- Threat Assessment
- Security Requirements
- Security Architecture

3) Implementation: focuses on processes and activities related to the implementation aspects of building and deploying software components together with their related defects. The goal is to release software that works as expected with minimum defects.

- Secure Build
- Secure Deployment
- Defect Management

4) Verification: focuses on processes and activities related to the verification aspects of checking and testing artefacts produced throughout software development. This typically includes quality assurance activities such as testing, and other review or evaluation activities.

- Architecture Assessment
- Requirements-driven Testing
- Security Testing

5) Operations: encompasses activities necessary to ensure confidentiality, integrity, and availability are maintained through the lifetime of an app and its data.

- Incident Management
- Environment Management
- Operational Management

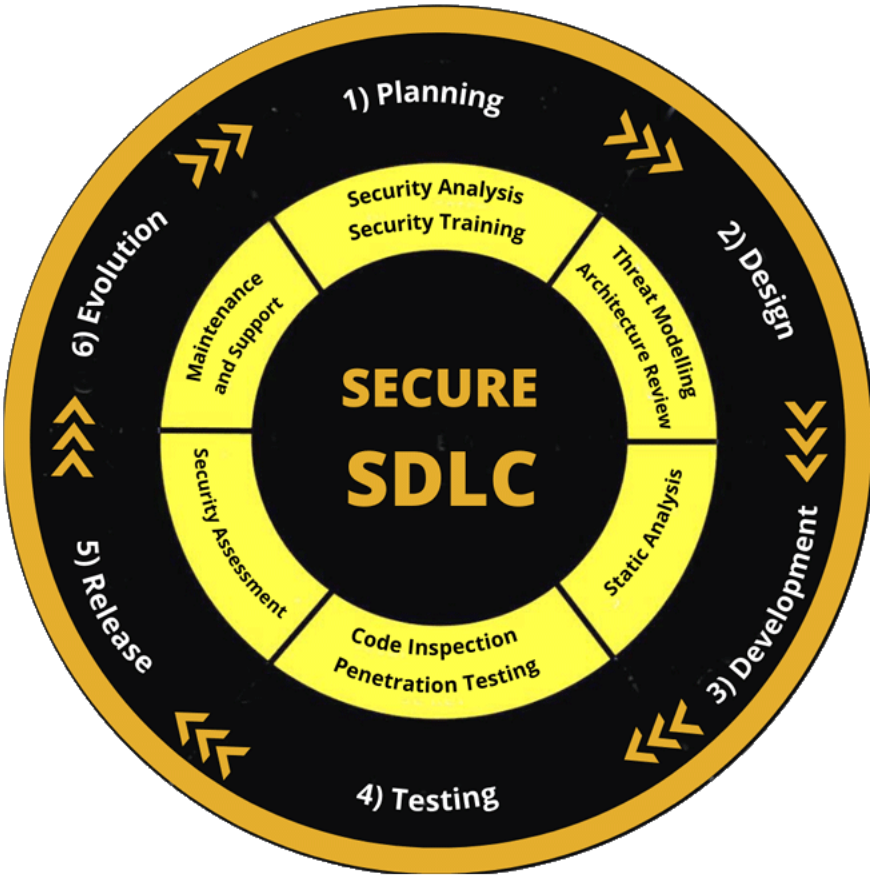
SSDLC vs. SDLC

SSDLC requires more time, work, and resources compared to the usual SDLC.

The Secure SDLC has its own specific tasks together with security tasks that will be carried over into the next stage. The Secure SDLC process is continuous and repeatable over many different activities in all the phases.

It's a multifaceted approach to security as a partnership. It requires teamwork, and it's a constantly evolving process.

There are typically six phases to an SSDLC, as shown below:



1) Planning

In the planning phase, the representatives from higher management and/or business owners, specialists, and developers will plan, discuss, and analyse the security problems and the scope and goals of the software project.

Organizational feasibility analysis is also performed to determine whether the organization has the available resources like budget or other constraints to conduct a successful security analysis and design of the software project.

In the normal SDLC process, there are no steps to discover and mitigate security threats during the planning stage.

- Security Analysis: performed to identify the possible risks connected with developing the software. This step is subject to a continuing process to enable future modifications and updates to the software to be done from time to time when new changes or new threats are introduced.

- Security Training: conducted to help a non-technical individual have a better understanding of information security elements.

2) Design

In the design stage, project managers, developers, and specialists will develop the blueprints for information security and implement key security policies such as encryption and security standards.

- Architecture Review: The developers will choose and review the right architecture for the software development during the Architecture Review process. It's crucial to do this properly because most vulnerabilities are introduced during this stage, and architectural flaws are the hardest to change. Each architecture needs a deep analysis of the technology profile and the attack surface to ensure it's not vulnerable by design.

- Threat Modelling: In the threat modelling step, the development team can work together with security professionals to better understand and identify specific threats.
  This involves understanding and identifying the specific threats by mapping the assets and protection in place. Then they create specific controls to handle those threats. The tools that can be helpful here are the Microsoft Threat Modelling Tool and the OWASP Threat Dragon.

3) Development

The latest platform, backend technology, programming language, and techniques are already chosen from the previous phase to ensure support for the development phase to produce high-quality and secure code. This is where the magic happens when the development phase starts, and developers begin to start building the software.

- **Static Analysis / (SAST)**[27]: In this activity, development teams evaluate the security threats connected with using third-party code, such as frameworks and libraries, and make preparations to minimise these risks. The tools that can be helpful here are static analysis tools or other security tools recommended for applications in the software development process.

Static application security testing (SAST) is performed without executing the application program, but rather inspecting the source code, byte code or application binaries for any security flaws.

4) Testing

The testing phase is done for quality assurance and to ensure there are no significant code errors in the software.

- Code Inspection: Software source codes will be reviewed using automated and manual ways to identify easy-to-spot code errors and critical vulnerabilities.

- Penetration Testing: Software will undergo hacking to identify and fix critical vulnerabilities before releasing the software to the public.

Dynamic application security testing (DAST) involves conducting simulated attacks on a running application program to analyse its reaction and discover security vulnerabilities.

5) Release

In the release phase, the software is launched to the public and can be used by users.

This phase will be finished with the security configuration steps to make sure the software operates securely when and after it is released. The objective of the Release phase and Security Assessment activity is to achieve maximum security for software to operate in a secure infrastructure, application, service, or system.

- Security Assessment: During this activity, developers conduct additional evaluation and validation testing on the whole software project to ensure the software is ready for release. Attack Surface Reduction steps will be conducted here to detect vulnerabilities to external attacks and diminish the software's attack surface.

6) Evolution

This is the most important phase because today’s software applications need constant monitoring, testing, modification, updating, maintenance, and support.

In this phase, the developers maintain the usability of the software and provide support to fix minor issues. The evolution phase and maintenance and support activities occur together.

- Maintenance and support: The battle for a stable, secure and reliable system is always needed whereas discovering the vulnerabilities and mitigating security threats is a constant effort to maintain security.

Chapter 3: How to be Successful in Secure SDLC

Nine security practises in SSDLC

There are nine components needed to implement information security in the software development lifecycle. It's a multifaceted approach to security as a partnership. It requires teamwork and it's a constantly evolving process.

1) Security Analysis

It is performed to identify the possible security risks connected with building the software and to ensure it is built according to the current security best practices.

Risk management also begins in this phase, which identifies, assesses, and evaluates the levels of risk an organisation is facing in terms of organisational security and organisational information.

The analysis of relevant legal issues that could affect the design of the security solution is also performed here.

2) Security Training

Each individual involved in the software development lifecycle will benefit from regular IT security training. This will ensure they understand The Basics: Security 101 and the common threats, as they constantly evolve and new things tend to come up every year.

3) Threat Modelling

In this step, it's all about understanding the attacker's perspective and visualising all the possible ways threats can harm the software, including internal and external threats.

This involves understanding and identifying the specific threats by mapping the assets and protection in place. Then creating specific controls to handle those threats. The tools that can be helpful here are the Microsoft Threat Modelling Tool and the OWASP Threat Dragon.

4) Architecture Review

In the design stage, it's important to ensure the right architecture is chosen for the software development. It's crucial to do this properly because most vulnerabilities are introduced during this stage, and architectural flaws are the hardest to change.

Each architecture needs a deep analysis of the technology profile and the attack surface to ensure it's not vulnerable by design.

Once the architecture for the software development project is confirmed, the way to move forward is to adhere to the best security practices and requirements. This includes choosing the proper programming language, database environments, systems, and features according to the current security best practices and standards.

5) Static Analysis

In the Static Analysis step, a combination of manual, static code analysis, and/or open source analysis is used.

Manual code analysis is the manual method of examining the code of the software to identify any logic flaws. The open-source analysis aims to avoid introducing critical vulnerabilities in the software. It's performed with the WhiteSource, SourceClear, or Snyk tool.

In Static Code Analysis, **white box security testing**[39] method is used to examine the source code of the software. Here automated code scans is conducted to identify obvious issues like programming errors, coding standards violations, undefined values, syntax violations, and security vulnerabilities. Identify issues and remediate them as soon as possible. The tools that can be helpful here are Veracode, Checkmarx, and Fortify.

White box security testing is the developer approach where access to the implementation of the software and its fundamental design and framework are available.

6) Code Inspection

It is done using a combination of manual/white box and automated/black box.

In the **white box testing**[39], source code is examined manually because the automated tool will not be able to identify logical flaws in the code.

In **black box testing**[5], tools are used to find easy-to-spot issues such as runtime bugs.

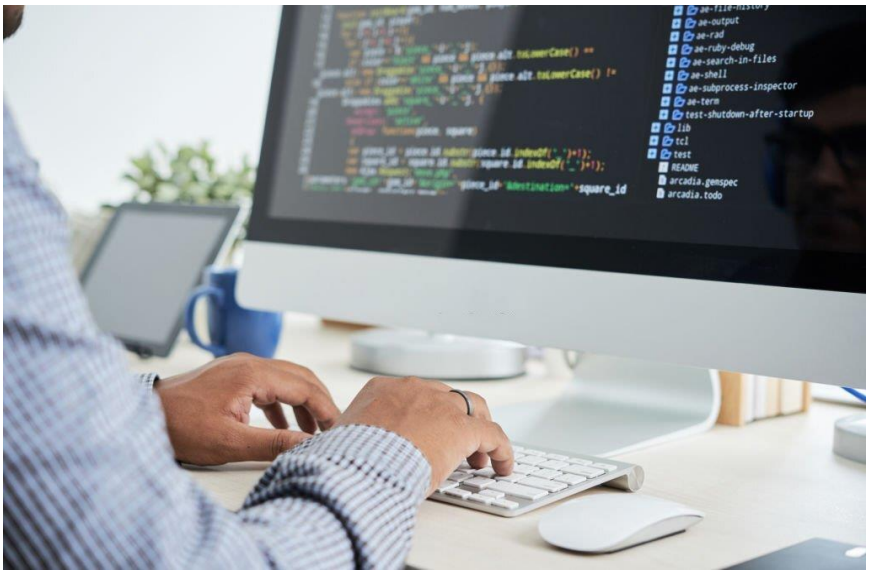
The benefit of this type of testing is to find issues that script kiddies (amateur hackers) can exploit. The tools that can be helpful here are ZAP, Burp, and AppScan. It's best to use a mix of both methods for the best results possible in the code review process.

Black box security testing*[5] *is the attacker approach where application is tested from the outside-in, with little or no prior knowledge of the application's internal workings.

7) Penetration Testing

Penetration Testing is considered hacking like real hackers would do to find and fix vulnerabilities before anyone else does. It’s done before releasing the software to the public and it’s best to do in a staging environment rather than in production, where it’s more dangerous.

In these steps, there would be not many vulnerabilities, about 20-30 findings, proving that all the initial phases in the secure SDLC is followed. The tools that can be helpful here are Burp, ZAP, and AppScan.



8) Security Assessment

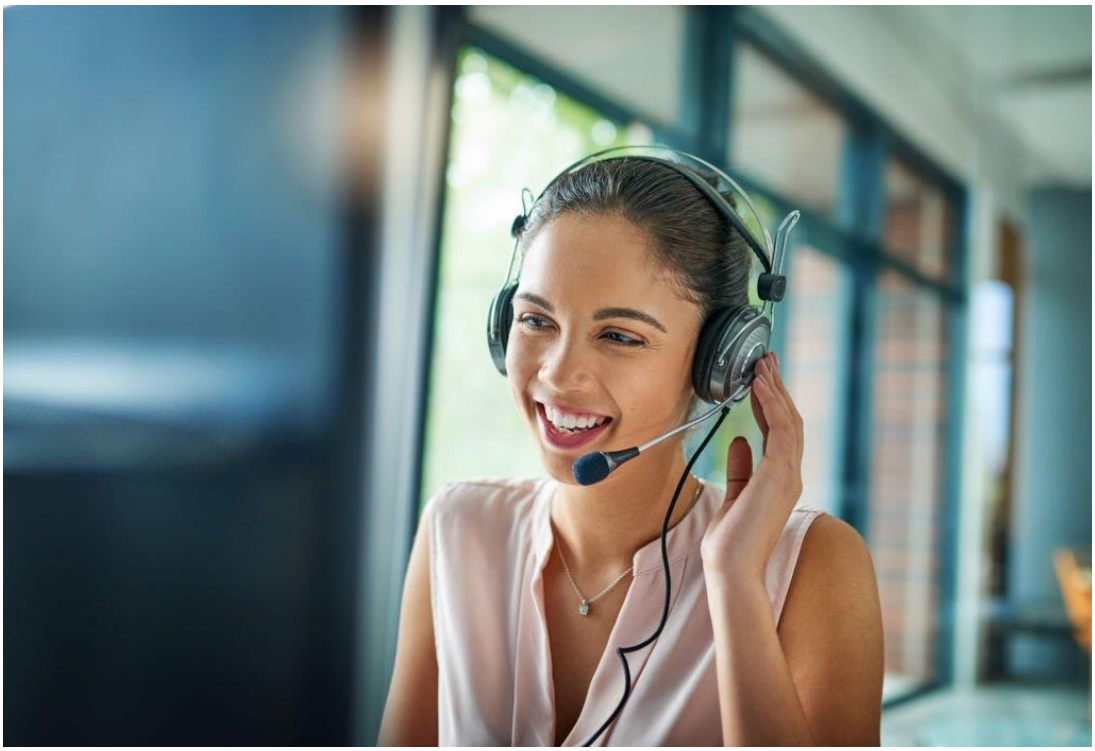
A security assessment is performed as a final evaluation and validation test on the software to ensure it is ready for release.

This phase evaluates the software from all aspects to support the final release and generates alternative solutions and fixes the final software before the release.

9) Maintenance and Support

For a period of up to 6 months, support for any changes will be provided, which also includes the maintenance and technical support for the software.

The SSDLC steps are continuous and always evolve as new vulnerabilities are discovered in the software.



Integrate the necessary security elements

In regards to using SSDLC, software developers can utilise the **OWASP ASVS**[20] and **OWASP Proactive Controls**[21] to provide guidance on how to integrate the necessary security elements into their development activities.

What is OWASP ASVS?

OWASP (The Open Web Application Security Project) and ASVS (Application Security Verification Standard) lay the groundwork for creating secure software.

Its main purpose is to serve as a guide for setting our security requirements baseline for our application.

How to use ASVS?

The Application Security Verification Standard is used as a blueprint to create a Secure Coding Checklist specific to the application, platform, or organization.

Tailoring the ASVS to the specific use cases will increase the focus on the security requirements that are most important to the projects and environments.

Application Security Verification Levels

Each software application project follows one of the levels.

ASVS Level 1 is for low assurance levels.

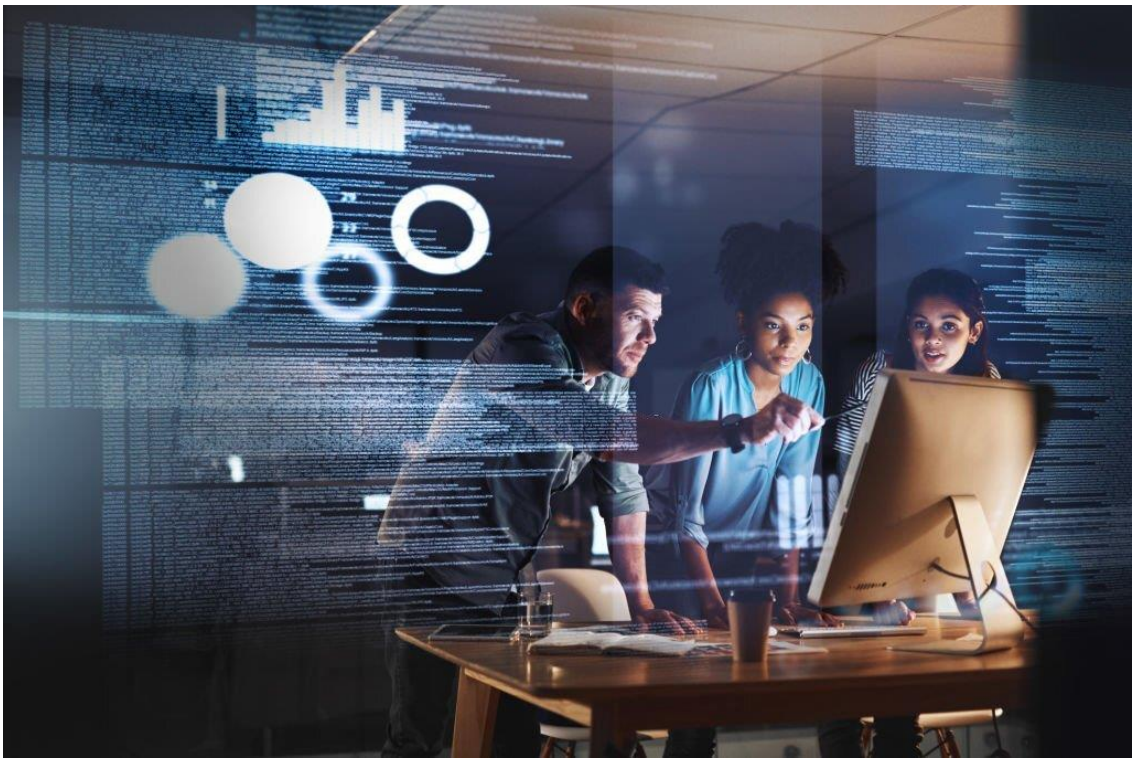
ASVS Level 2 is for applications that contain sensitive data that requires protection and is the recommended level for most apps.

ASVS Level 3 is for the most critical applications—applications that perform high-value transactions, contain sensitive medical data, or any application that requires the highest level of trust.

A few categories inside ASVS. Each category has security requirements that represent the best practices.

- **Authentication**[2]
- Access Control
- Session Management
- Input and Output
- Cryptographic
- Communications
- Data Protection
- Business Logic
- Configuration

and many others.



OWASP Proactive Controls

OWASP (The Open Web Application Security Project), Top 10 Proactive Controls 2018, is a useful guide for effective security techniques which will support software developers in building secure software.

List of proactive controls:

- Define Security Requirements
- Leverage Security Frameworks and Libraries
- Secure Database Access
- Encode and Escape Data
- Validate All Inputs
- Implement Digital Identity
- Enforce Access Controls
- Protect Data Everywhere
- Implement Security Logging and Monitoring
- Handle All Errors and Exceptions

Chapter 4: Common software vulnerabilities discussed

Software Developer's daily challenge is to make sure the software is secure. Before taking the steps to secure the software, as well as user and company data against an array of vulnerabilities, an understanding of the top 10 web application security risks[5] is crucial.

Let's explore the top 10 most frequent threats for 2021.

Broken Access Control

Broken access control allows attackers to access or view sensitive data without the authorization or permission level to do so. For example, changing the URL to access the admin page from /user1234/info to /admin/info.

The secure software development life cycle utilizes a code review process to enforce security in the software codes by defining user permissions and actions for each type of object that accesses a data source using input from the user.

Cryptographic Failures

Cryptographic failures in a software application can be caused by wrong data encryption, mishandling of cryptography keys and using old or weak cryptographic algorithms.

Cryptographic issues in an application can be avoided by using the approach below:

- Ensure that the correct data is encrypted and not leaving critical data exposed.
- Ensure proper storage and management of cryptographic keys
- Use existing encryption or hashing algorithms.

Injection

An injection attack allows attackers to manipulate systems by sending malicious code and requests through user input fields. This most commonly happens using:

- **SQL injection**[32]
- Lightweight directory access protocol (**LDAP**[15]) queries
- XML path language (**XPATH**[41]) queries
- Operating system (OS) commands
- Object-relational mapping (**ORM**[17])

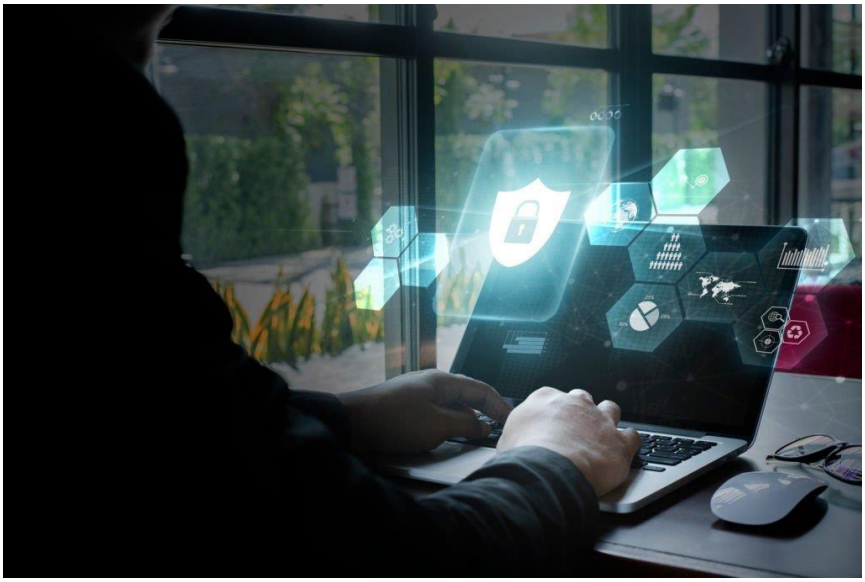
Secure software development life cycle uses security controls to eliminate these types of attacks through input validation (each input that is coming from a login screen or a search window should be validated, filtered, and sanitized). In the validation process, define data types and strict patterns for all string parameters. In the filtering process, special characters like the apostrophe, asterisk or equal sign should be restricted because they can be used by hackers to form arguments in code.

Allowing acceptable strings, and blocking everything else, will deny attackers to send malicious code through user input fields which keeps injection attacks away.

One of the injection attacks called **CRLF**[7] injection makes use of the **HTTP**[11] protocol with CRLF character sequences to signify where one header ends and another begins, and where headers end and the website content begins. Applications susceptible to these attacks can have malicious content injected or users redirected to a malicious location.

While CRLF injections can be used as attack vectors for cross-site scripting (**XSS**[40]) attacks, they are not one and the same. CRLF injections can occur when an attacker forces the application to return the CRLF sequence, plus the attacker's supplied data as a part of the response. That data could be an XSS attack, but not necessarily.

Cross-site scripting (XSS) is a software vulnerability usually found in Web applications. This XSS allows online criminals to inject client-side script into pages that other users view. The cross-site scripting vulnerability can be employed at the same time by attackers to over-write access controls. This issue can become a significant security risk unless the network administrator or the website owner doesn't take the necessary security means.



Insecure Design

Insecure design relates to the architecture flaws at the start of the software development journey, leading to insecure software as a by-product.

This can be eliminated by adopting the "Shift-Left" mentality. In SDLC, this refers to more focus on the planning and design phase by following secure design patterns and principles at the earliest stages. Threat modelling can be utilized to visualize and identify potential threats in the design phase to improve the design with security in mind.

Security Misconfiguration

Security misconfiguration in software could happen as it depends on the number of possible features or functions a software has. The more things that need to be configured and tweaked the easier it is to mess things up. Usually, it consists of disabled security features, debug features enabled and improper permissions vulnerabilities[6].

In most cases, security misconfiguration leads to information leakage in a web application by which any sensitive information leak could compromise security.

Examples of information leakage include:

- A stack trace indicating application error messages in the browser when users attempt to access a specific page.
- A detailed comment was left by a developer.
- Network configuration files that provide information about the network infrastructure powering the application.

Debugging[10] is a useful tool to help developers get detailed errors message to investigate and fix issues. Typically, it is turned off by default, but in some cases, developers forget to turn it off and leave it enabled during a long development process in a production environment. This will pose a risk where attackers will get information about the environment that can aid with how to compromise an application, or even an entire server or network.

Secure software development life cycle assists in developing a standard and repeatable process to review security settings across the entire environment to eliminate disabled security features and improper permissions vulnerabilities.

The review of security settings should be made continuous across the entire environment so the organization is always striving to improve its security settings and configuration for its applications.

Vulnerable and Outdated Components

Today's modern software will not exist without using the necessary components to make it function well. These components include popular libraries and frameworks for each programming language such as **Python[23]**, **C++[6]**, etc.

Avoiding vulnerable and outdated components is not an easy task for most organizations. But companies can take measures to get rid of such issues by maintaining an inventory of components used and ensuring it is updated.

Identification and Authentication Failures

Identification and **authentication[2]** failures happen when an authentication process in software is broken or vulnerable[8].

Secure software development life cycle has a process to create the authentication section of software appropriately. This is achieved by including authentication checks everywhere that allow users to access program functionality. **Multifactor authentication (MFA[16])** is also a good method to include in a software application for secure authentication.

Authentication: The process of authentication (or identification) of an individual is usually based on a username and a password. This process is used to allow access to an online location or resource to the right individual by validating the identification.

Software and Data Integrity Failures

Software and data integrity failures happen with the use of critical data or applications from untrusted sources without verifying their identity[7].

One of the most significant breaches of this nature is the SolarWinds Orion attack where malicious updates were distributed to more than 18,000 organizations worldwide.

Security Logging and Monitoring Failures

Security logging and monitoring for all failed authentication, denied access and input validation errors are important as a part of a good cybersecurity response plan. To avoid this vulnerability, utilize automation and monitoring processes to have a constant process in place to collect logs and also give incident alerts in the case of any breach.

Server-side Request Forgery

Server-Side Request Forgery (SSRF) allows an attacker to abuse server functionality to read or update internal resources. This is achieved by fetching a URL to the vulnerable web application which will often have privileges to read, write or import data. The application will be forced to send requests to access unintended resources such as sensitive, personal or corporate information.

Your Secure Software Development Partner!

Our development teams and technical teams consist of highly qualified software development experts and cybersecurity experts who are the perfect combination of unique talents, skills and experience to lead secure software development projects from start to finish.

To stay ahead, we apply the updated and cutting-edge technology in software development and information security best practices that will ensure your project will be successful and meet your expectations and goals.

About Condition Zebra

Condition Zebra is a **CREST**[8] certified and **ISO 27001:2013**[12] company that offers Professional Cybersecurity Solutions and Cybersecurity Training for SMEs in various industries, including Financial Services (Banks & Insurance), Government Ministries & Agencies, and Government-linked companies.

If you're looking to leverage our expertise, that is to get the best solutions that demonstrate the highest levels of knowledge, skills, and competence, then reach out to us today!

Condition Zebra (M) Sdn Bhd
Level 3-10, Block F, Phileo Damansara 1,
Jalan 16/11 Off Jalan Damansara,
46350 Petaling Jaya, Selangor,
MALAYSIA.

Website: www.condition-zebra.com
Email: info@condition-zebra.com
Phone: +603-7665 2021


ConditionZebra


ConditionZebra


Condition Zebra (M) Sdn Bhd


+60122293908


ConditionZebra


Condition Zebra

Glossary

1. **Agile**: Agile methodology is a modern approach in project management that is widely used in software development. Agile Software Development is an alternative to the traditional sequential process of development, known as “waterfall”. Agile helps with the timely response to any changes in the project, and corrects the workflow accordingly.

2. **Authentication**: Authentication is the process of confirming the correctness of the claimed identity.

3. **BSIMM**: BSIMM (pronounced “bee simm”) is short for Building Security In Maturity Model. BSIMM is a study of real-world software security initiatives organized so that you can determine where you stand with your software security initiative and how to evolve your efforts over time.

4. **Backdoor**: A backdoor is a tool installed after a compromise to give an attacker easier access to the compromised system around any security mechanisms that are in place.

5. **Black box security testing**: Black box security testing is the attacker approach where application is tested from the outside-in, with little or no prior knowledge of the application's internal workings.

6. **C++**: C++ is a cross-platform language that can be used to create high-performance applications. C++ was developed by Bjarne Stroustrup, as an extension to the C language. C++ gives programmers a high level of control over system resources and memory.

7. **CRLF**: CRLF stands for Carriage Return Linefeed, which is a special sequence of characters (0x0D 0x0A in hex) used by the HTTP protocol as a line separator. A CRLF Injection attack occurs when an attacker manages to force the application to return the CRLF sequence plus the attacker’s supplied data as part of the response headers.

8. **CREST**: The Council of Registered Ethical Security Testers - is an international accreditation and certification body, representing and supporting the technical information security industry.

9. **DAST**: Dynamic application security testing (DAST) involves conducting simulated attacks on a running application program to analyze its reaction and discover security vulnerabilities.

10. **Debugging**: Debugging is a useful tool to help developers get detailed errors message to investigate and fix issues.

11. **HTTP**: The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load web pages using hypertext links.

12. **ISO 27001:2013**: Specifies the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of the organization.

13. **ISO 27034**: Offers guidance on information security to those specifying, designing and programming or procuring, implementing and using application systems, in other words, business and IT managers, developers and auditors, and ultimately the end-users of ICT. The aim is to ensure that computer applications deliver the desired or necessary level of security in support of the organisation's Information Security Management System, adequately addressing many ICT security risks.

14. **Iterative**: Iterative development is a software development approach that breaks the process of developing a large application into smaller parts. Each part, called "iteration", represents the whole development process and contains planning, design, development, and testing steps. Unlike the Waterfall model, the iterative process adds features one-by-one, providing a working product at the end of each iteration, and increases functionality from cycle to cycle.

15. **LDAP**: The Lightweight Directory Access Protocol (LDAP) is a vendor-neutral application protocol used to maintain distributed directory info in an organized, easy-to-query manner. That means it allows you to keep a directory of items and information about them.

16. **MFA**: Multifactor Authentication, this type of authentication uses two or more factors to achieve authentication. These factors can include: something the users knows (a password or a PIN), something the user has (an authentication token, an SMS with a code or a code generator on the phone/tablet) and/or something the user is (biometric authentication methods, such as fingerprints or retina scans).

17. **ORM**: Object-relational mapping (ORM) is a programming technique in which a metadata descriptor is used to connect object code to a relational database. Object code is written in object-oriented programming (OOP) languages such as Java or C#. ORM converts data between type systems that are unable to coexist within relational databases and OOP languages.

18. **OWASP**: The Open Web Application Security Project (OWASP) is an international non-profit organization dedicated to web application security.

19. **OWASP SAMM**: SAMM (Software Assurance Maturity Model) is used to analyze and improve the current secure development lifecycle process.

20. **OWASP ASVS**: The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development.

21. **OWASP Proactive Controls**: The OWASP Top Ten Proactive Controls 2018 is a list of security techniques that should be included in every software development project.

22. **Penetration Testing**: This is a type of attack launched on a network or computer system in order to identify security vulnerabilities that can be used to gain unauthorized access to the network's/system's features and data. Penetration testing is used to help companies better protect themselves against cyber attacks.

23. **Python**: Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

24. **SAST**: Static application security testing (SAST) is performed without executing the application program, but rather inspecting the source code, byte code or application binaries for any security flaws.

25. **Secure Code reviews**: Secure code review is a manual or automated process that examines an application's source code. The goal of this examination is to identify any existing security flaws or vulnerabilities.

26. **Shift-Left**: In the software development lifecycle, this refers to moving towards the planning and design phase at the earliest stages.

27. **SDLC**: The Software Development Lifecycle (SDLC) is the usual process used by organisations to build software from start to finish.

28. **SSDLC**: Secure Software Development Lifecycle simply means producing software with security in mind from the ground up. By embedding security practices such as Static application security testing (SAST), Dynamic application security testing (DAST), threat modelling, secure code reviews, etc) at each stage of the SSDLC process to produce a more security focused application software.

29. **SQL Injection**: This is a tactic that used code injection to attack applications which are data-driven. The maliciously injected SQL code can perform several actions, including dumping all the data in a database in a location controlled by the attacker. Through this attack, malicious hackers can spoof identities, modify data or tamper with it, disclose confidential data, delete and destroy the data or make it unavailable. They can also take control of the database completely.

30. **VAPT**: Vulnerability Assessment and Penetration Testing (VAPT) consists of various types of security assessment services to discover vulnerabilities in your IT assets and infrastructures such as networks, firewalls, servers and applications (web/mobile).

31. **Waterfall**: The waterfall model in software engineering is a traditional and somewhat “old-fashioned” project management approach. It is called waterfall because it is linear and sequential. Just like waterfall it always moves forward, not a single step back. Each waterfall phase has strictly defined goals and deadlines and takes place one after another. The primary goal of the waterfall method is to gather and make clear all the requirements upfront, to prevent the development from going ‘downhill’ without the possibility of making changes.

32. **White box security testing**: White box security testing is the developer approach where one is able access the implementation of the software and its fundamental design and framework.

33. **XSS attacks**: Cross-site scripting (XSS) is a software vulnerability usually found in Web applications. This XSS allows online criminals to inject client-side script into pages that other users view. The cross-site scripting vulnerability can be employed at the same time by attackers to over-write access controls. This issue can become a significant security risk unless the network administrator or the website owner doesn't take the necessary security means.

34. **XPATH**: XPath is a major element in the XSLT standard. XPath can be used to navigate through elements and attributes in an XML document.

References

1. "State of Software Security v12: Don’t become complacent, but we’ve come a long way." Developer-Tech.com. 8 Feb, 2022. https://www.developer-tech.com/news/2022/feb/08/state-of-software-security-v12-dont-complacent-but-come-long-way/

2. "How to ensure app security with secure SDLC implementation." Sigma Software. 8 Jul, 2021. https://sigma.software/about/media/how-ensure-app-security-secure-sdlc-implementation

3. "How You Should Approach the Secure Development Lifecycle." Dataversity.net. 12 July, 2019. https://www.dataversity.net/how-you-should-approach-the-secure-development-lifecycle/

4. "What is OWASP SAMM?" OWASPSAMM.org. https://owaspsamm.org/about/

5. "OWASP Top Ten". OWASP.org. https://owasp.org/www-project-top-ten/

6. "Coders Conquer Security OWASP Top 10 API Series - Disabled Security Features/Debug Features Enabled/Improper Permissions". Secure Code Warrior. 11 Nov, 2020. https://www.securecodewarrior.com/blog/coders-conquer-security-owasp-top-10-api-series-disabled-security-features-debug-features-enabled-improper-permissions

7. "A08:2021-Software and Data Integrity Failures". Medium. 22 Sept, 2021. https://medium.com/@shivam_bathla/a08-2021-software-and-data-integrity-failures-967a564140a

8. "Coders Conquer Security OWASP Top 10 API Series - Broken Authentication". Secure Code Warrior. 16 Sept, 2020. https://www.securecodewarrior.com/blog/coders-conquer-security-owasp-top-10-api-series-broken-authentication

9. "OWASP Application Security Verification Standard". OWASP.org. https://owasp.org/www-project-application-security-verification-standard/

10. "OWASP Proactive Controls". OWASP.org. https://owasp.org/www-project-proactive-controls/

11. "Secure SDLC || Secure Software Development Life Cycle|| SSDLC in Information Security". UGC NET Competitive Exams. https://www.youtube.com/watch?v=ojPjMphY1ts

12. "What is SSDLC (Secure Software Development Lifecycle)?" - Syed Shah, Snr Security Tester.
https://www.youtube.com/watch?v=oOcO2kSttRM

13. "Introduction to Application Security (AppSec)" - Christophe Limpalair.
https://www.udemy.com/course/introduction-to-application-security-appsec/

14. "Agile Software Development". Easternpeak.com.
https://easternpeak.com/definition/agile-software-development/

15. "Iterative Development". Easternpeak.com.
https://easternpeak.com/definition/iterative-development/

16. "Glossary of Security Terms. SANS.org.
https://www.sans.org/security-resources/glossary-of-terms/

17. "BSIMM". bsimm.com.
https://www.bsimm.com/about/faq.html

18. "What is Black-box Security Testing?". Acunetix.
https://www.acunetix.com/blog/articles/black-box-security-testing/

19. "C++ Introduction". w3schools.com.
https://www.w3schools.com/cpp/cpp_intro.asp

20. "CRLF Injection". secapps.com.
https://secapps.com/vulndb/crlf-injection

21. "Trust and Assurance".
https://www.crest-approved.org/

22. "What is HTTP? Cloudflare.com.
https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/

23. "ISO/IEC 27001:2013
Information technology — Security techniques — Information security management systems — Requirements".
https://www.iso.org/standard/54534.html

24. "ISO/IEC 27034:2011+ — Information technology — Security techniques — Application security".
https://www.iso27001security.com/html/27034.html

25. "Cyber Security Glossary". HEIMDAL Security.
https://heimdalsecurity.com/glossary#M

26. "Lightweight Directory Access Protocol (LDAP)".
https://www.extrahop.com/resources/protocols/ldap/

27. "Object-Relational Mapping (ORM)".
https://www.techopedia.com/definition/24200/object-relational-mapping--orm

28. "What is Python Used For? A Beginner's Guide.
https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

29. "Secure Code Review".
https://www.synopsys.com/glossary/what-is-code-review.html

30. "XML and XPath". w3schools.com.
https://www.w3schools.com/xml/xml_xpath.asp

www.condition-zebra.com