GitLab

The Beginner's Guide to Building Secure Software

Table of Contents

Part 1: The Importance of Secure Software

Software can be vulnerable to attacks that can steal private customer data, damage infrastructure, or even take services down completely. Cyberattacks and breaches are occurring more frequently than ever before, leaking vast amounts of private data. These breaches can have great repercussions for an organization, such as regulatory fines, service downtown, resulting in a loss of reputation, trust, and ultimately customers. The Open Web Application Security Project (OWASP) has identified the following as some of the main causes of software breaches:

- **Injection flaws:** These are flaws in the way that software handles user input, which can allow attackers to inject malicious code into the system.

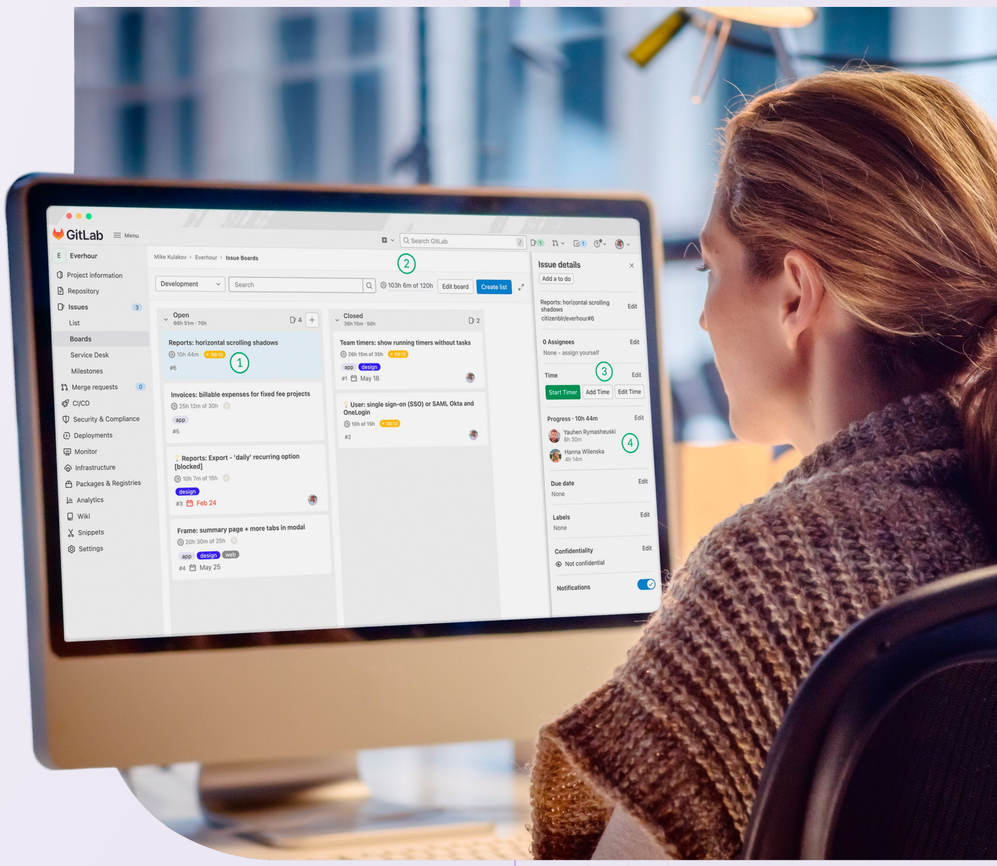
- **Broken authentication and session management:** These are flaws in the way that software manages user authentication and sessions, which can allow attackers to steal user credentials or take over user accounts.

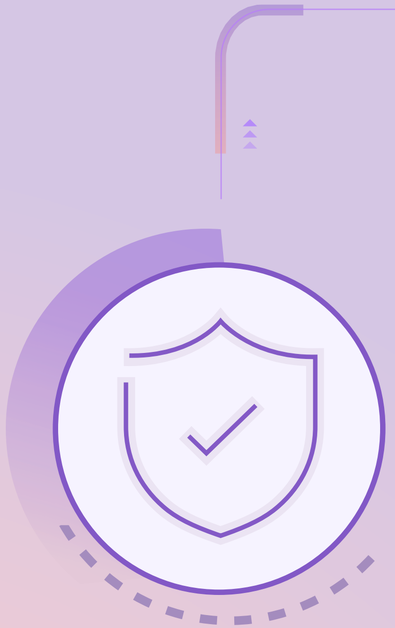
- **Cross-site scripting (XSS):** This is a type of attack that allows attackers to inject malicious code into websites, which can then be executed in the victim's browser.

- **Security misconfiguration:** This is a general term for flaws in the way that software is configured, which can leave it vulnerable to attack.

- **Insecure design**: This is a general term for flaws in the way that software is designed, which can leave it vulnerable to attack.

All these risks can be mitigated by implementing security best practices for the complete software development lifecycle (SDLC) and by following suitable security processes to ensure that an application is secure. This process involves building a security team, setting up security controls, enabling collaboration, and continuously adhering to compliance.





Part 2: Application Security Team Roles and Responsibilities

While security is everyone's responsibility, your organization's security team plays a crucial part in protecting the organization from security threats. Here are some of the personas you may find in your organization's security team:

- **Security architect:** Designs and implements the organization's security architecture, which includes security policies, procedures, and controls.
- **Security engineer:** Implements and maintains the organization's security controls, such as vulnerability scanning, penetration testing, and security incident response.
- **Security analyst**: Monitors the organization's security posture by analyzing system logs and network traffic.
- **Security researcher:** Researches and develops new security solutions which include vulnerability discovery, exploit development, and security testing.
- **Security trainer:** Develops security awareness training programs, conducts security training sessions, and provides security consulting services.
- **Compliance officer:** Ensures that an organization is compliant with internal and external regulations.
- **CISO (Chief Information Security Officer):** The CISO is responsible for the overall security of the organization, which includes developing and implementing security policies and procedures, and managing the security team.

These are some of the common roles you may need for your security team, but note that every team is built differently. These individuals must collaborate with developers to create and maintain a strong security posture that can help to protect the organization from a variety of different security risks.

Security teams typically have far fewer members than development teams, making collaboration a challenge. The path to nurturing a strong partnership between security and development teams is through the tools they use. For strong collaboration, these tools must be easy to use, reduce context switching, and provide a single-source-of-truth for both developers and the security team.

Part 3: Implementing Security within the Complete SDLC

When people think of security, they usually think of implementing controls at the application level to prevent a breach, such as a firewall. However, security involves more than just the application - it involves securing the complete SDLC.

The SDLC is a process which organizations use to develop high-quality and efficient software. It includes the following stages:

Planning: Create a plan for developing the software, which includes developing timeline, cost-analysis, and resource allocation.

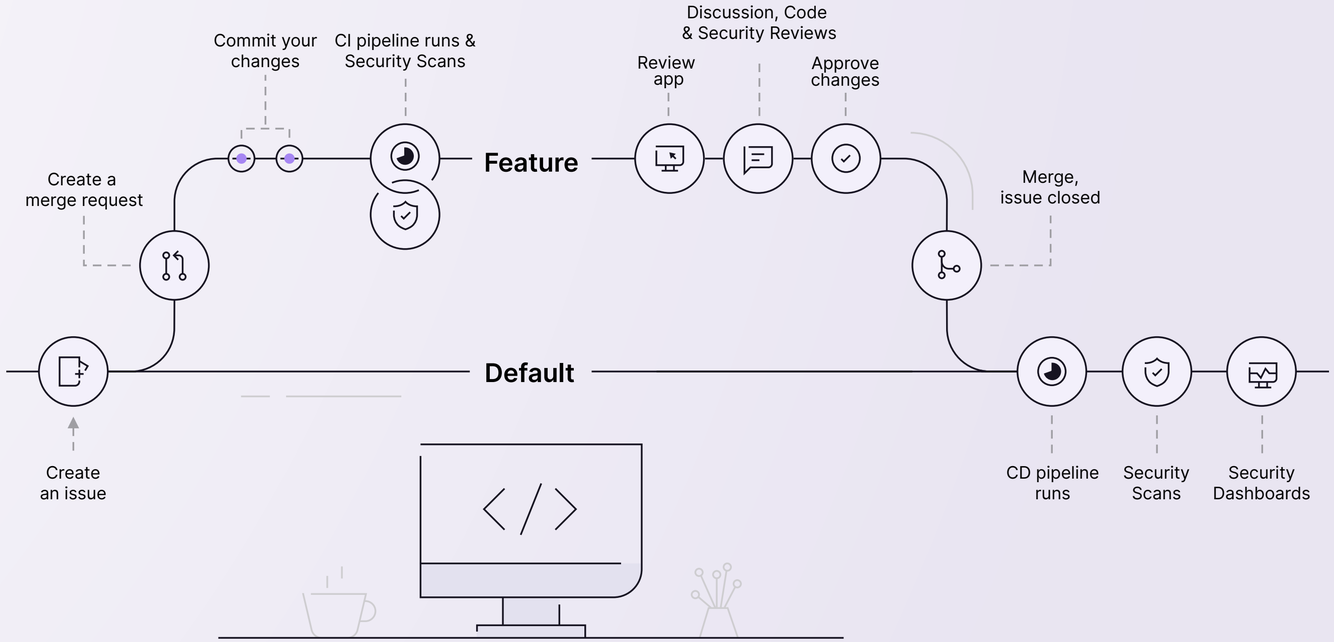
Designing: Determine the best solutions for developing the application.

Implementing: Begin writing the software according to the design specifications.

Testing/Securing: Examine code for bugs and security issues.

Deploying: Make the software accessible to others.

These stages should be automated as much as possible, and performed continuously as needs may change.



SDLC in a nutshell

Rather than just scanning for security flaws after deploying, security should be shifted-left, meaning it should be considered in earlier phases of the SDLC. Relating to shifting-left, the diagram above, shows the following:

- Security scans are run along with the CI pipeline which should include unit-tests.
- Production deployments are blocked until a code/security review is performed and changes are approved.
- Upon approval, deployment is automatically performed.
- Security dashboards are populated for security posture oversight and monitoring.

This process prevents insecure code from being deployed into production. In order to begin implementing security best practices to your organization's SDLC, you can start by implementing security scanners and adding security controls throughout your project.

Implementing Security Scanners

One of the first steps in Application Security is to regularly scan your application for vulnerabilities. Security scanners can help identify vulnerabilities before they are deployed to a production environment, allowing you to proactively address the risk of your application. Scanners can either scan static files or the deployed application itself for vulnerabilities.

Static security scanners search your source code in the Git repository for vulnerabilities. The recommended static scanners are as follows:

- **Static Application Security Scanning (SAST):** Checks your application source code for known weaknesses.
- **Secret Detection:** Scans all text files, regardless of the language or framework used to help prevent your secrets from being exposed.
- **Container Scanning:** Scans your application's container images for known vulnerabilities.

- **Dependency Scanning:** Analyzes your application's dependencies for known vulnerabilities, including transitive dependencies, also known as nested dependencies.

- **Infrastructure as Code (IaC) Scanning:** Scans your infrastructure definition files for known vulnerabilities.

- **Coverage-guided Fuzz-testing:** Sends random inputs to an instrumented version of your application in an effort to cause unexpected behavior. Such behavior indicates a bug or security risk that you should address.

- **License Scanning:** Scans licenses found in application dependencies. Some licenses may be restrictive and can have legal implications.

- **Code Quality Scanning:** Analyzes your source code's quality and complexity. This helps keep your project's code simple, readable, and easier to maintain further contributing to making it easy to maintain application security.

There are also dynamic security scanners that search for vulnerabilities in an application that has already been deployed and is running. Dynamic scanners should run in a pre-production environment to ensure that security issues are resolved before the code is pushed to production.

- **Dynamic Application Security Scanning (DAST):** Examines your running application in a deployed environment for vulnerabilities.

- **Web API Fuzz-testing:** Sets operation parameters to unexpected values in an effort to cause unexpected behavior and errors in the API backend. This helps you discover bugs and potential security issues that other QA processes may miss.

While the best approach is to run scanners within a pipeline for every code change, you can also schedule scanners to run as needed. These scanners should also be integrated into your code review system, allowing both developers and security team members to communicate and assess risk.

Adding Security Controls

A crucial control to safeguard your application is to implement separation of duties. Separation of duties means that more than one person should be required to complete a task. Some examples of separation of duties include:

- The same developer who added a new commit should not be allowed to merge it into a production-level branch without approval. Approval should be performed by another team member who has reviewed the code change.

- If a vulnerability is detected in a new commit, only a member of the security team should be able to approve the code change to production once they have validated that the vulnerability will not impact the production environment.

Separation of duties can be achieved by implementing the following types of controls:

- **Security Scanner Policy:** Allows approval to be required based on the vulnerability findings of one or more security scan jobs.
- **License Policy:** Allows approval to be required based on the licenses detected and the policy set for acceptable licenses.
- **Merge Request Approval Policy:** Allows a minimum baseline of approvals to be required either for all merge requests or for merge requests containing unsigned commits.
- **Scan Execution Policies:** Used to require that security scans run on a specified schedule or with the project pipeline on each Git push or Merge Request update.
- **Compliance Frameworks:** A label which identifies that your project has certain compliance requirements or needs additional oversight. The label can optionally enforce different compliance requirements for the projects on which it is applied.
- **Branch Protections:** Imposes further restrictions on certain branches, such as who can merge, push, etc. into the branch.
- **Code Owners:** Defines who has expertise for specific parts of your project's codebase in order to require owners to approve changes.

Part 4: Adhering to Compliance

As companies scale and practices change, controls may become outdated or non-functional, rendering them open to attacks by malicious actors. A compliance policy must be established and adhered to in order to make sure that your organization is secure and continues to be secure as you grow.

Compliance is the state of establishing and maintaining guidelines set by either regulatory organizations, governments, or internally. An important part of compliance is having an oversight of your system and its status.

Vulnerability Reporting

A key function of the security team is to strengthen the security posture of the organization as a whole. One key way of doing this is the ability to manage all the vulnerabilities contained within the production-level branch of an application. The Vulnerability Report enables security teams to maintain a good security posture by providing cumulative results of all security jobs, regardless of whether the pipeline was successful. The results provided are as follows:

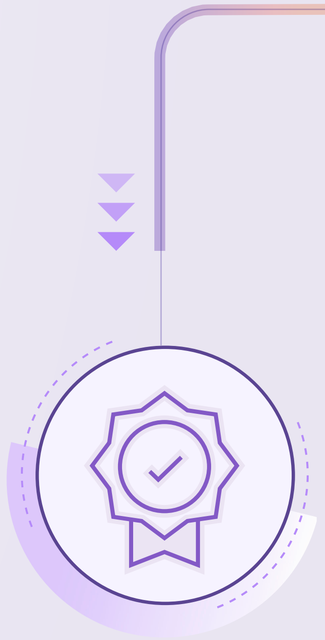
- Totals for vulnerabilities per severity level
- Filters for common vulnerability attributes
- Details of each vulnerability, presented in a tabular layout

Each detected vulnerability can be assigned a status such as Dismissed, Resolved, Confirmed, or Detected in order to provide documentation for others on the team. To assist with the triage process, additional vulnerability details can be seen when drilling down which include:

- Description
- When it was detected
- Current status
- Available actions
- Linked issues
- Actions log

This information is also provided in merge requests which developers can use this data to address issues while coding and communicate crucial information on the vulnerability to the security team. The security team can use this data in order to triage the vulnerabilities and build a plan on which to tackle first. Furthermore, security teams can assess risk directly in the vulnerability page by leveraging Artificial Intelligence (AI) and Large Language Models (LLM) to do the following:

- Summarize the vulnerability.
- Help developers and security analysts understand the vulnerability, how it could be exploited, and how to fix it.
- Provide a suggested mitigation and automatically create a Merge Request.



Security and Compliance Dashboards

Seeing Trends in Security are useful for the security team to understand the cause of increased security risks as well as what reduces them. This understanding can allow an organization to create initiatives based on the data to reduce the number of vulnerabilities introduced and increase the number of vulnerabilities mitigated. A few examples where trends can be valuable:

- Seeing a large number of introduced vulnerabilities on October 10, determine that a large refactor was performed, resulting in performing smaller refactor commits.
- Seeing a large number of vulnerabilities reduced on October 15, determine that a hackathon was held that day, resulting in increased focus on the hackathon.

Trends in compliance are equally as important. Compliance dashboards should be used to see where your organization is falling out of compliance.

Auditing

System audits are very important for compliance. You must be able to track important events, including who performed the related action and when, so that if something happens you are prepared to take action. You can use audit events to track, for example:

- Who changed the permission level of a particular user and when.
- Who added a new user or removed a user and when.

Part 5: Conclusion

By implementing the security measures mentioned in this book, you can help prevent unauthorized access, data breaches, and other security incidents. This can assist your organization in protecting your data, your customers' data, and its reputation.

To learn more, scan the QR code below or visit this link.

About GitLab

GitLab is the most comprehensive, AI-powered DevSecOps Platform for software innovation. GitLab provides one interface, one data store, one permissions model, one value stream, one set of reports, one spot to secure your code, one location to deploy to any cloud, and one place for everyone to contribute. The platform is the only true cloud-agnostic end-to-end DevSecOps platform that brings together all DevSecOps capabilities in one place.

With GitLab, organizations can create, deliver, and manage code quickly and continuously to translate business vision into reality. GitLab empowers customers and users to innovate faster, scale more easily, and serve and retain customers more effectively. Built on open source, GitLab works alongside its growing community, which is composed of thousands of developers and millions of users, to continuously deliver new innovations.