Whitepaper

Secure Coding Practices

Upskilling Developers with Cybersecurity Skills to Build Robust Applications

OffSec
The Path to a Secure Future™

For today's organizations, strengthening digital security is a never-ending process, battle, and cause for concern. Significant resources are dedicated to implementing solutions and policies aimed at protecting the organization, its employees, and its customers from cyber attacks. Despite these efforts, many companies do experience breaches, often because of software vulnerabilities. With intense pressure to deliver new software to market quickly, developers often prioritize functionality over security – leaving the software exposed to attacks. But with cyber attacks on the rise and the costs of data breaches skyrocketing into the millions, organizations need to embrace secure software development, a new approach that focuses on implementing security at every stage of the software development lifecycle (SDLC).

This whitepaper will dive into secure software development and explore:

- Why secure coding is important
- Key principles and best practices for secure coding
- How developers can integrate security into each phase of the SDLC
- The benefits of this approach to the organization, developer, and security practitioner
- And lastly, why secure software development training is imperative to giving developers and security practitioners the knowledge and hands-on experience needed to secure software solutions and improve the overall security posture for the organization.

1 Why building secure software is essential

Companies are adopting new technologies, platforms, and software at an accelerated pace as they strive to continuously modernize and embrace digital transformation. Unfortunately, as companies become more tech savvy and sophisticated, so too do the cybercriminals looking to exploit any potential vulnerabilities within those organizations. Cyber attacks increased by 38% globally in 2022 vs. 2021[1], 450,000 new malware are created daily[2], and 80% of organizations have suffered a third-party software-related breach in the past year.[3] Very often these attacks are due to software vulnerabilities and can result in significant costs to organizations including fines, reputation damage, revenue loss, customer attrition, legal fees, and poor business outcomes.

Why is software posing a growing liability to an organization's security posture? Simply put, the traditional mindset around software development doesn't focus enough on security. Tasked with creating innovative software within aggressive timeframes, developers often see security as an impediment to their creativity and a quick time to market. Research shows that 67% of developers knowingly ship code with vulnerabilities[4], and only 20% of newly hired developers have received secure coding training.[5]

Security is often addressed later in the software development lifecycle when security teams test the software prior to implementation or after it's in production in the form of bug fixes. It just isn't top of mind, with 86% of developers saying they don't see security as a top priority when it comes to software development.[6] But this approach is flawed for two key reasons. First, it doesn't scale with an organization's need for innovation and rapid time to market with new solutions. Software is being created and deployed so quickly that it's going into market with vulnerabilities that put an organization at risk for a breach. Second, this traditional process has a significant negative impact on the business's bottom line given it's 6x more costly to fix a bug during implementation, 15x more expensive to fix that same bug during testing, and 100x more expensive to fix the same bug after deployment.[7]

Today's organizations need to adopt a modern mindset around software development that focuses on security at every stage of the SDLC. To do this, organizations should be proactive by training developers on secure software development practices so they can identify and mitigate vulnerabilities early in the software development process when it costs less and is more effective.

> It is **6x more costly** to fix a bug during implementation, 15x more expensive to fix that same bug during testing, and 100x more expensive to fix the same bug after deployment.[19]
>
> Today's organizations need to adopt a modern mindset around software development that focuses on security at every stage of the SDLC.
>
> Organizations should be proactive by training developers on secure software development practices so they can identify and mitigate vulnerabilities early in the software development process when it costs less and is more effective.

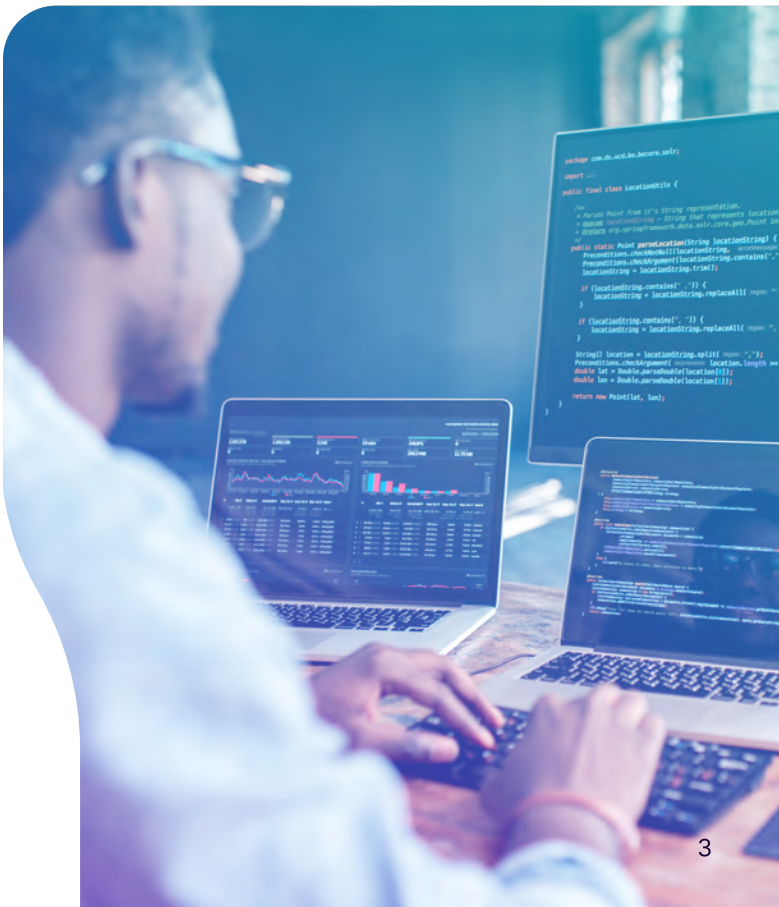
2 What is secure software development?

Developing software is in many ways like manufacturing a car. They both require a step-by-step process for the build and a quality assurance (QA) check at the end. They also both require several security measures to be put in place along the way to conform with compliance regulations and to keep customers safe. Imagine building a car, only to have QA discover it lacks key safety features like airbags, antilock brakes, seat belts, and child safety locks. The manufacturer has two choices now: one is to go back and put all these features into the completed car, which is going to be costly and delay time to market, or two, they can release the car to market without these key safety features, which puts the company, the driver, and other drivers and pedestrians at significant risk. Neither option is ideal.

Organizations and developers can avoid falling into a similar trap by using secure software development, a comprehensive approach that involves incorporating security measures into every phase of the SDLC, including from the very beginning of the development process, ensuring that security considerations are woven into the fabric of the application. Examples would be using threat modeling, secure design, security testing, vulnerability assessments, and more as early as possible in the development cycle. This methodology enables companies to “shift left” by adopting a mindset where security is no longer seen as a separate process, but rather a key component of the development process and a key responsibility for developers.

Secure software development is guided by three key principles development teams can use to address common security threats and vulnerabilities.

1. **Secure by design**: By integrating security into the software design process from the ground up, developers can make security an essential aspect of the design process vs. an afterthought or add-on.
2. **Least privilege:** Developers should grant users the absolute minimum level of access to system resources required to execute their tasks based on their functional roles and responsibilities. By limiting access, organizations can minimize the potential impact of any security breaches or unauthorized access.
3. **Defense in depth:** Using a layered approach to system security, developers can ensure that if one security layer fails, there are others in place to help mitigate and control the damage. These protection layers can include firewalls, access controls, intrusion detection systems, and security monitoring systems, among others.



3 Secure coding best practices

There are several secure coding practices developers can use to significantly reduce the risk of introducing security vulnerabilities into their software applications.

Input validation

Input validation is one of the most fundamental secure coding practices that involves validating and sanitizing user inputs to prevent malicious inputs from introducing code in a system. Checking input data for its size, type, format, and other characteristics is critical in blocking various threats like injection attacks.

Secure authentication

Authentication is a critical component of securing software applications, and strong authentication mechanisms ensure that the right people have access to the right data. Developers can follow secure coding practices by ensuring that passwords are stored securely, hashed with appropriate algorithms such as bcrypt, and using session tokens or cookies for maintaining authenticated sessions.

Secure session management

Secure session management involves proper handling of user sessions so they remain valid while in use but expire when no longer in use. Maintaining user sessions securely is crucial in safeguarding user data as well as protecting against session hijacking. Developers can follow best practices by ensuring that session cookies are randomly generated, are sufficiently long, and employ secure encryption.

Secure communication

Secure communication is essential to ensure that data transmitted over networks remains confidential and cannot be modified by unauthorized persons. Secure communication can be achieved by using appropriate encryption techniques, such as SSL/TLS or AES encryption. Developers can use standardized encryption libraries to ensure secure communication in their applications.

Error handling

Proper error handling is an important aspect of secure software development. Because error messages that provide too much detail can be exploited by attackers to uncover vulnerabilities in the system, error messages should not reveal sensitive information and should only be used in meaningful situations where they can be helpful to users and reduce confusion.

4 The Secure Software Development Lifecycle

The secure SDLC is a standardized process that guides software development teams in building secure software applications. However, 70% of organizations are missing critical security steps in their SDLC: almost 50% do not dedicate a step for security validation, 20% don't plan their application security, and 4% don't have a dedicated security implementation step.[8] But the benefits of shifting left are significant, with 9 in 10 of those that have adopted a shift left approach reporting reduced vulnerabilities.[9]

Integrating security into each phase of the SDLC helps ensure that the final software product is secure, reliable, and trustworthy. There are five main phases of the SDLC with a focus on security:

Step 1
Planning

The development team outlines the software's scope and objectives. Security is integrated into this phase by considering security risks and establishing security requirements for the application. Security requirements can include access control mechanisms, threat modeling, authentication protocols, and encryption requirements.

Step 2
Design

Developers create a detailed architectural plan for the application. Secure coding principles, such as secure by design, are integrated into this phase to ensure that security is part of the software's design. The design phase also includes setting up security protocols like Transport Layer Security (TLS) and network encryption.

Step 3
Implementation

During this phase, developers write the code and incorporate secure coding practices like input validation and proper error handling. Developers can use code analysis tools for secure coding practices and identify vulnerabilities before the code is deployed. Code reviews and testing tools can ensure that the code adheres to security standards.

Step 4
Testing

The code is tested by the developers for functionality, performance, and security. Security testing techniques focus on discovering vulnerabilities in the software application, such as penetration testing and vulnerability scanning, to identify security flaws in the application.

Step 5
Deployment

Developers deploy the software application and ensure that the environment in which the application operates meets the necessary security requirements. Security is integrated into this phase by ensuring that access controls and network security mechanisms, such as firewalls, are in place.

5 Benefits of embracing secure software development

In the digital age, where cyber attacks are becoming more prolific, sophisticated, and dangerous, taking a secure software development approach is fundamental to protecting an organization, its people, customers, and assets. Key benefits of secure software development include:

Reduced breach risks

Developers can identify and address potential security vulnerabilities in the software early in the development lifecycle, which reduces the likelihood of data breaches occurring during and after deployment.

Improved customer trust

83% of consumers will stop spending with a business for several months in the immediate aftermath of a security breach, and over a fifth (21%) of consumers claim they will never return to a business post-breach.[10] By using a secure software development approach, developers provide reliable and robust software that is less prone to vulnerabilities and data breaches – critical to customer retention.

Cost savings

In 2023, the average cost of a data breach reached $4.45 million, a 15% increase over 3 years.[11] And globally, damage from cyber attacks is expected to increase by 300% over 10 years.[12] Using a shift left approach to software development can greatly reduce the risk of a breach and subsequent fines. Additionally, fixing vulnerabilities in the early stages of development can be up to 100x less expensive than fixing vulnerabilities in the later stages of development or attempting to address a breach or vulnerability that occurs after deployment.[13]

Regulatory compliance

Regulatory requirements, such as the General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS), require organizations to implement secure software development practices. Failure to do so can result in massive fines, for example anywhere from 2%-4% of their annual revenue from the preceding year if they violate GDPR standards.[14] By using secure coding practices to adhere to these regulations, organizations can achieve compliance and avoid any violations or fines.

Competitive advantage

32% of customers will abandon their favorite brand after just one bad experience, according to a new study.[15] Offering customers reliable and secure software can differentiate an organization from its competitors and build trust among customers, which in turn can lead to increased market share.

6 Why secure software development training is imperative

Secure software development training is hands-on learning that teaches developers and security professionals how to implement security concepts throughout software development lifecycles. Through a combination of content on language-agnostic secure coding principles, videos, and practical exercises, development teams can build skills applicable to a broad range of web technologies.

This training is imperative to giving developers the knowledge and hands-on experience they need to shift left – ensuring security is embedded into every stage of the SDLC while also enabling them to innovate and meet aggressive deadlines. However, many developers say they are either not adequately trained on secure software development, not confident in their skills, or that the methodology is at odds with other corporate priorities. Less than 50% of software developers say they can spot security vulnerabilities in software[16].

Who needs this training?

Anyone committed to the defense or security of enterprise applications should experience comprehensive and hands-on learning for secure software development, including security software engineers, applications engineers, release managers, software developers, and security practitioners.

For developers, this training is essential to ensure they have the knowledge and skills to incorporate security practices and principles into their code from the very beginning. By understanding secure coding techniques, developers can proactively identify vulnerabilities and implement robust security measures to minimize the risk of potential breaches or exploits.

For security practitioners, the training enables them to effectively collaborate with developers. By acquiring knowledge about the software development process, security practitioners can better understand the inherent challenges and complexities involved in building secure code. This experience allows them to provide more accurate guidance and recommendations to developers, bridging the gap between security and development teams.

Benefits of secure software development training

There are significant benefits for organizations that use secure software development training. In fact, a recent survey found that when continuous training is delivered by third parties and implemented in tandem with code reviews and code scanning tools, 100% of organizations saw improvement in their code security.[17] Additionally, research shows that secure coding training has a high return on investment: almost 30% of organizations utilizing continuous training have prevented over 90% of vulnerabilities from reaching production.[18] Additional benefits of secure software development training include:

- Learning how to write secure code and avoid common bugs and vulnerabilities
- Enriching DevOps and DevSecOps skills with secure coding principles
- Creating a security-first mindset through practical exercises
- Gaining confidence in skills
- Fostering a culture of security awareness and accountability across the organization

By training both developers and security practitioners on secure software development practices, they can establish a shared language and set of best practices to promote effective collaboration and communication. The results are the creation of more secure software solutions and a stronger overall security posture for the organization.

7 Conclusion

With the rapid adoption of software and continuous onslaught of cyber attacks, secure software development is imperative for today's modern organization. Adopting a shift left approach by prioritizing security across the entire SDLC greatly improves an organization's security posture while reducing the risks and costs associated with fines and bug fixes. Secure software development training is critical to helping developers and security practitioners adopt a security-focused mindset and implement the proper security controls during each phase of the SDLC.

OffSec offers a secure software development Learning Path to empower developers to build and deploy secure software from the start to prevent vulnerabilities, and to give security professionals a deeper understanding of the software development process. Learn more.

[1] https://www.av-test.org/en/statistics/malware/
[2] https://www.av-test.org/en/statistics/malware/
[3] https://www.darkreading.com/operations/identity-related-breaches-last-12-months
[4] https://research.contrary.com/reports/snyk
[5] https://resources.infosecinstitute.com/topics/secure-coding/only-20-of-new-developers-receive-secure-coding-training-says-report/
[6] https://blog.gitguardian.com/top-10-practices-for-secure-software-development/
[7] https://www.researchgate.net/figure/IBM-System-Science-Institute-Relative-Cost-of-Fixing-Defects_fig1_255965523
[8] https://www.helpnetsecurity.com/2023/01/23/trained-developers-code-scanning-tools/
[9] https://www.helpnetsecurity.com/2023/01/23/trained-developers-code-scanning-tools/
[10] https://securityboulevard.com/2023/01/what-happens-to-a-customer-after-a-data-breach/#:~:text=In%20the%20US%2C%2083%25%20of,to%20a%20business%20post%2Dbreach
[11] https://securityintelligence.com/articles/cost-of-a-data-breach-2023-geographical-breakdowns/
[12] https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/new-survey-reveals-2-trillion-dollar-market-opportunity-for-cybersecurity-technology-and-service-providers
[13] https://www.researchgate.net/figure/IBM-System-Science-Institute-Relative-Cost-of-Fixing-Defects_fig1_255965523
[14] https://www.eqs.com/compliance-blog/biggest-gdpr-fines/
[15] https://technative.io/how-convenience-balanced-with-security-builds-brand-loyalty/
[16] https://arxiv.org/pdf/2101.02085.pdf
[17] https://www.helpnetsecurity.com/2023/01/23/trained-developers-code-scanning-tools/
[18] https://www.helpnetsecurity.com/2023/01/23/trained-developers-code-scanning-tools/
[19] https://www.researchgate.net/figure/IBM-System-Science-Institute-Relative-Cost-of-Fixing-Defects_fig1_255965523