March 31, 2023

Best Practices For Secure Software Development

SECURITY & COMPLIANCE

By Dzuy Tran

Secure software development best practices are necessary because security risks are everywhere. In an era of cyberattacks, they can affect everyone — including individuals, corporations, and governments. For that reason, ensuring security in software development is essential.

Here we explain what is secure software, how to ensure security, and provide best practices for secure software development (https://www.blazemeter.com/blog/best-practices-security-testing-software).

Read along or jump ahead to the section that interests you the most:

What Happens without Secure Software Development?

Cyberattacks make headlines. Duqu (https://resources.infosecinstitute.com/duqu-2-0-the-most-sophisticated-malware-ever-seen/#gref) and Stuxnet (https://www.csoonline.com/article/3218104/malware/what-is-stuxnet-who-created-it-and-how-does-it-work.html) had everyone talking in 2010 and 2011. And, cyberattacks have only gotten worse since then. WannaCry (https://www.csoonline.com/article/3227906/ransomware/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html) hit important systems in 2017, including Britain's National Health Service. GitHub (https://www.wired.com/story/github-ddos-memcached/) was hit by a denial of service attack in early 2018. And a 2021 Log4j (https://www.zdnet.com/article/log4j-flaw-thousands-of-applications-are-still-vulnerable-warn-security-researchers/) vulnerability is still being exploited today.

📕 Related Content: GitLab SAST (https://www.perforce.com/blog/kw/how-to-use-klocwork-gitlab): Learn how to use GitLab with Klocwork.

Embedded Systems Aren't Immune to Secure Software Engineering Risks

Embedded systems (https://www.perforce.com/solutions/embedded-systems) are increasingly open to risk. That's led to recalls in the medical device and automotive industries. And, the automotive industry, in particular, is vulnerable to cyberthreats (https://www.pwc.com/us/en/industries/industrial-products/library/automotive-cyber-readiness.html).

This is a huge problem.

Cyberattacks against embedded systems could lead to wide-scale damage to:

- Critical infrastructure, including power generation, oil, and gas refining.
- Telecommunications.
- Transportation.
- Water and waste control systems.

📕 Related White Paper: How to Improve Embedded Systems Security (https://www.perforce.com/resources/qac/how-improve-embedded-systems-security).

5 Key Secure Software Development Risk Factors (http://www.informit.com/store/software-security-engineering-a-guide-for-project-managers-9780321509178?w_ptgrevartcl=Why+Is+Security+a+Softwa

The five key secure software development risk factors are:

1. Interdependent systems make software the weakest link.

2. Software size and complexity complicates testing.

3. An **outsourced software supply chain** increases risk exposure.

4. Sophisticated attacks find more risk.

5. Legacy software is reused.

Common Secure Software Issues in Today's Application Security (AppSec) Landscape

Today, various types of software applications are developed for embedded systems, mobile devices, electric vehicles, banking, and transactional services. However, it is often overlooked that many apps and digital experiences are designed and operated without security measures, which can be risky if security is not a top priority.

📕 Related Content: Get an Overview of Application Security (https://www.perforce.com/blog/kw/what-is-appsec)

Even if security is prioritized and secure software development practices are implemented, companies can still be caught off guard. The common issues in today's application security landscape include:

- **Vulnerabilities in third-party libraries and frameworks**: Many applications rely on third-party libraries and frameworks, which can introduce vulnerabilities into the application if not updated regularly.
- **Injection attacks**: Injection attacks involve an attacker injecting malicious code or commands into an application's input fields, such as login forms or search boxes, to gain unauthorized access to the application or its underlying database.
- **Cross-site scripting (XSS)**: XSS attacks involve an attacker injecting malicious code into a website or web application, which can then execute in the user's browser, potentially stealing sensitive data or performing unauthorized actions on behalf of the user.
- **Insecure authentication and authorization**: Poorly designed or implemented authentication and authorization mechanisms can allow attackers to bypass security controls and gain access to sensitive data or functionality.
- **Insufficient logging and monitoring**: Without adequate logging and monitoring, it can be difficult to detect and respond to security incidents or identify the root cause of security issues.
- **Mobile application security**: With the proliferation of mobile devices, ensuring the security of mobile applications has become increasingly important. Mobile applications can be vulnerable to a range of attacks, including those targeting the device itself or the application's backend servers.

- **Cloud security**: With the growing use of cloud computing, ensuring the security of cloud-based applications has become critical. Cloud-based applications can be vulnerable to a range of attacks, including those targeting the cloud infrastructure, the application itself, or the data stored in the cloud.

One or more of the secure coding compliance measures, such as OWASP Top 10 (https://www.perforce.com/blog/kw/what-is-owasp-top-10), CWE Top 25 (https://www.perforce.com/blog/kw/what-is-cwe), and CERT (https://www.perforce.com/blog/kw/what-is-cert) rules set, could be utilized to detect the items on the above list for secure software development.

How Do SAST Tools Help Ensure Best Practices for Secure Software Engineering?

More organizations are investing in software security development and cybersecurity technologies, which include SAST tools — like Klocwork (https://www.perforce.com/products/klocwork). Despite that many advances have been made in cybersecurity coverage, much of the effort has been focused on adding security after the fact and improving threat detection.

Many are now realizing the importance of SAST (https://www.perforce.com/blog/kw/what-is-sast) and enforcing a secure development process.

It's not enough to apply new security technologies. The software itself needs to close risk gaps. Putting stronger locks on your front door is no use if the windows are left open.

Why Is Security in Software Development Difficult?

Secure Software Isn't a Big Enough Priority

Security in software development isn't a big enough priority for most developers.

There's an old saying that you need to:

- Get to market fast.

- Include all features planned.
- Maintain a high level of quality.

But, you can only have two out of the three. So, while quality is part of the conversation, security is often left behind.

Features and deadlines drive development checklists. And, secure software usually isn't a feature or a requirement. So, it's rarely addressed.

Quality Doesn't Necessarily Guarantee Security

Improving software quality and software integrity (https://www.perforce.com/blog/qac/what-is-software-integrity)*can* reduce security flaws that result from defects. But, QA usually doesn't take hacking into consideration.

Too Many Moving Parts in Embedded Development

Embedded systems (https://www.perforce.com/solutions/embedded-systems) are big and complex.

There's new and legacy code — and connectivity components. And, embedded systems run on a variety of operating systems.

Multiple development teams work on software. And, they're often spread around the world.

Not to mention it's difficult enough to ensure that the software functions properly. It can be even more difficult to ensure secure software.

Not Enough Training for Security

Unfortunately, many people involved in software development don't know how to recognize security problems. This includes the security implications of certain software requirements — or lack thereof.

And, they don't know how security impacts the way software is:

- Modeled
- Architected
- Designed
- Implemented
- Tested
- Prepared for distribution and deployment

So, developers may not design secure software. Security requirements may be lacking. And, developers might not understand how a mistake turns into a security vulnerability.

No One Owns Security

Most embedded development teams don't have someone tasked with software security. Instead, they rely on a variety of roles — from product management to development to QA — to make software secure. And, that doesn't always work.

📕 Related Content: Guide for Software Development and Software Security (https://www.perforce.com/blog/kw/software-development-and-software-security).

10 Best Practices for Secure Software Development

With the understanding that we could potentially have one or more of the common AppSec issues mentioned above, ask yourself, "What are the most effective ways to ensure security in code development, practices, processes, or methodologies?"

Modern thinking dictates that secure software development pertains to the approach of creating software applications that are intentionally designed and executed with security considerations.

Even if you have access to the best testing toolchains for scanning and analyzing your software, this process should entail implementing various practices and methodologies to identify and alleviate potential security threats and weaknesses at every stage of the software development lifecycle.

📕 Related Content: Guide to Secure Coding Practices (https://www.perforce.com/blog/qac/secure-coding-standards).

Here are 10 best practices for secure software development:

1. Threat Modeling

Threat modeling involves analyzing the software architecture and identifying potential security threats and vulnerabilities. This helps in designing the software with security in mind and implementing the necessary security controls.

2. Secure Software Coding

Developers must adhere to secure coding practices, such as input validation, secure data storage, and secure communication protocols. Secure coding practices help to prevent common security vulnerabilities such as SQL injection, cross-site scripting, and buffer overflow attacks.

3. Code Review

Code review involves reviewing the code written by developers to identify potential security issues. This helps in detecting and correcting security vulnerabilities early in the development process.

4. Testing

Regular security testing, including penetration testing and vulnerability scanning, can help identify potential security weaknesses in the software. This helps in fixing security issues before the software is deployed.

5. Secure Configuration Management

Configuration management ensures that software systems are deployed with secure configurations. This includes configuring access controls, network settings, and other security-related settings to reduce the risk of unauthorized access.

6. Access Control

Access control ensures that only authorized personnel can access the software system. This includes implementing user authentication and authorization mechanisms, as well as role-based access control.

7. Regular Updates and Patches

Regular software updates and patches help to address security vulnerabilities and reduce the risk of security breaches. It is important to stay up to date with security patches and updates for all software components used in the system.

8. Security Training

Developers and other personnel involved in the software development process should receive regular security training to ensure that they understand the importance of security and the best practices for secure software development.

9. Incident Response

Organizations should have a well-defined incident response plan in place to respond to security incidents. This includes identifying potential security incidents, containing the impact of security incidents, and recovering from security incidents.

10. Continuous Monitoring

Continuous monitoring helps in detecting and responding to security incidents in real time. This includes monitoring system logs, network traffic, and user behavior for any signs of security breaches.

By following these best practices, organizations can develop secure and reliable software applications that can withstand potential security threats and vulnerabilities. It is crucial to prioritize security in every stage of software development to prevent unauthorized access and protect sensitive data.

Use Static Code Analysis Tools to Help Ensure Secure Software Development

Static code analysis supports a secure development process because half of all security defects (https://cwe.mitre.org/documents/sources/SevenPerniciousKingdoms.pdf) are introduced at the source code level. So, finding and fixing bugs as soon as code is written is critical.

But, many developers lack security training. And, identifying security problems during a code review can be difficult, if not impossible. Security mistakes can be subtle and easy to overlook even for trained developers.

Static code analysis tools can bridge that knowledge gap, and they flag security vulnerabilities and accelerate code reviews.

Using static analysis, developers can identify errors, including:

- Memory leaks
- Access violations
- Arithmetic errors
- Array and string overruns

This maximizes code quality (https://www.perforce.com/blog/qac/what-code-quality-and-how-improve-it) and minimizes the impact of errors on the finished product — and project timeline.

Plus, static code analysis tools — such as Helix QAC (https://www.perforce.com/products/helix-qac) for C/C++, and Klocwork (https://www.perforce.com/products/klocwork)C, C++, C#, Java, JavaScript, Python, and Kotlin — can be used to comply with CERT C (or MISRA) coding rules. And, they can identify CWE coding errors faster.

Learn more about applying secure coding standards to better ensure a secure software development process.

➡ SIGN UP FOR A KLOCWORK FREE TRIAL

(https://www.perforce.com/products/kw/free-static-code-analyzer-trial)

Dzuy Tran (/author/dzuy-tran)

Klocwork and Helix QAC, Sr. Solutions Architect, Perforce

Dzuy Tran has over 30 years of experience in designing and development of Hardware and Software Embedded Systems, RTOS, Mobile Applications and Enterprise Systems. He helps customers when they have technical questions, assists with Proof of Concepts, and conducts demos of the Static Code Analysis tools and help guided customers on DevOps implementation processes and Continuous Integration deployment. Dzuy holds a master's degree in Computer Science and Computer Engineering from National Technological University.

Helix TeamHub (/Products/Helix-Teamhub)
Helix Swarm (/Products/Helix-Swarm)
Methodics IPLM (/Products/Methodics-Iplm)
VersIC (/Products/Versic)

Test & Validate

Helix QAC (/Products/Helix-Qac)
Klocwork (/Products/Klocwork)

Operate, Manage, & Scale

SourcePro (/Products/Sourcepro)
HostAccess (/Products/Hostaccess)
HydraExpress (/Products/Hydraexpress)
[illegible]
(/Products/Pv-Wave)

Version Control (/Solutions/Version-Control)
IP Lifecycle Management (/Solutions/Ip-Lifecycle-Management)
Static Analysis (Https://Www.Perforce.Com/Solutions/Static-Analysis)
Audit & Compliance (/Solutions/Audit-And-Compliance)
Configuration Management (Https://Www.Puppet.Com/Puppet/Use-Cases/Continuous-Configuration-Automation)
IT Infrastructure & Automation (Https://Www.Puppet.Com/Products/Puppet-Enterprise)
Backlog Management (/Solutions/Product-Backlog-Management-_Z)

Helix Core (/Support/Consulting/Helix-Core-Consulting)
Helix QAC (/Support/Consulting/Helix-Qac-Consulting)
Klocwork (/Support/Consulting/Klocwork-Consulting)
Methodics IPLM (/Support/Consulting/Methodics-Iplm-Consulting)
OpenLogic (Https://Www.Openlogic.Com/Services/Consulting)
Perfecto (Https://Www.Perfecto.Io/Professional-Services-Implementation)
Zend (Https://Www.Zend.Com/Services)

Training

Training Overview (/Support/Training)
Hansoft (/Support/Training/Hansoft-Training)
Helix ALM

SUPPORT >

CUSTOMERS >

Case Studies (/Customers)

Integrations (/Integrations)
Resellers (/Partners/Reseller-Partners)

PERFORCE

SEND FEEDBACK