

KuppingerCole Report

Whitepaper

Software Supply Chain Security: Don't Get Your Code Tampered

Recent events such as the SolarWinds and Kaseya incidents have demonstrated the need to focus significantly more on software supply chain security. Thus, avoiding code tampering by external attackers and internal parties is essential. This whitepaper looks at how to increase security throughout the Software Lifecycle and implement a multi-layered, defense-in-depth code tampering prevention and detection strategy.



by **Martin Kuppinger**
mk@kuppingercole.com

Contents

Executive Summary	3
Highlights	3
The Need for Software Supply Chain Security	4
Secure SDLCs and where Code Tampering Prevention comes in	5
How to do Code Tampering Prevention right – multi-layered & defense-in-depth	7
The Cyscale Approach to Code Tampering Prevention	9
Recommendations: Securing your SDLC, securing your customers, securing yourself	10
Related Research	11
Copyright	12

Executive Summary

Software security has gained massive awareness since the end of 2020 due to two major attacks on software supply chains. Both the SolarWinds and the Kaseya attacks affected the systems of many clients and put an increased focus on the need for improving software security.

The SDLC (Software Development Lifecycle) and the entire DevOps cycle, from creating software to running it in the cloud or any other environment, have become massively more complex over the past few years. This complexity is mainly due to the number of tools involved in managing code, such as Source Control Management (SCM) systems, as well as in building applications and deploying and operating code. Unfortunately, this complexity leads to a broadened attack surface.

From the SCM, where both application code and infrastructure-as-code are managed, to cloud-based build and runtime environments, the attack surface includes a multitude of tools that make up the CI/CD pipeline. Moreover, the high degree of integration and automation across the entire pipeline allows for lateral movement of attackers.

Therefore, securing the entire SDLC is both a challenge and an imperative. Code Tampering Prevention is a key element within software security and helps prevent internal or external attacks that tamper with code to create malicious software. Attackers might alter code or inject malicious code at any point, so code tampering prevention must span the entire pipeline.

Successful implementation of a secure SDLC with strong code tampering prevention, therefore, requires solutions that cover all stages of the software delivery pipeline from the SDLC to the runtime environment in an integrated manner.

Cycode delivers a software supply chain security solution that builds on a common set of policies, covering a range of capabilities including critical code monitoring and file integrity verification, and implementing security policies consistently across the entire pipeline. This allows organizations to implement a comprehensive solution for code tampering prevention, beyond isolated solutions such as SAST/DAST (Static/Dynamic Application Security Testing).

Highlights

- The changing threat landscape requires organizations to invest in software security, with software being central to the digital business of organizations and attacks, thus putting the business at risk.
- Securing the SDLC (Software Development Lifecycle) requires a range of activities and solutions, including both technical and organizational measures.
- Given that all software starts with coding, and that malicious code is a key element in software supply chain attacks and other types of attacks, preventing code tampering is at the forefront of these security measures.
- Code tampering prevention requires taking actions at various levels, from critical code monitoring and file integrity verification, to enforcing consistent authentication and access controls, and identifying anomalies.
- Cycode delivers a comprehensive platform that allows for a unified, policy-based approach on code tampering prevention across the entire CI/CD pipeline, from code to cloud.

The Need for Software Supply Chain Security

Organizations must implement a comprehensive approach to software supply chain security for protecting their digital services and delivering secure, reliable solutions to their customers. With the increase in software supply chain attacks, software security has shifted to the center of attention.

Since late 2020, software supply chain attacks have risen to the top of the agenda in cybersecurity. Two major incidents, affecting software vendors SolarWinds and Kaseya, resulted in distributing malicious software to their customers. By tampering with COTS (commercial off the shelf) software, attackers managed to multiply their attacks and gain access to thousands of other organizations.

With the SolarWinds and Kaseya attacks, software security has moved to the center of attention.

While the concept of software supply chain attacks is not new, the scale of these attacks has gone far beyond that of other attack types, shifting the focus towards software security and software supply chain security. From an attacker perspective, such attacks are immensely attractive, due to the “multiplier effect” they have. Thus, our prediction is that the number of such attacks will increase in 2022 and beyond, with some of the attacks having a major impact.

While traditional attacks, either via phishing and identity fraud, or utilizing weaknesses in software, also known as zero-day attacks, will continue to play their role in cybercrimes and cyberattacks, software supply chain attacks and, more generally, attacks on the SDLC (Software Development Lifecycle) are adding to the attack surface of organizations.

Development tools and the DevOps infrastructure have become a target of choice, not only because of the “multiplier effect”, but also because security within the software supply chain is, generally speaking, weak. With the advent of SCMs (Source Control Management Systems) such as Git and the related platforms like GitHub, there are entry points for attackers where they can cause major damage.

Source Control Management Systems are an ideal place for attacks, holding both application code and infrastructure-as-code, and therefore they must be as secure as possible.

Securing the SDLC starts with addressing the cultural and organizational divide between software engineering/DevOps and the security teams, and between IT infrastructure and networking teams. While DevOps tools are commonly owned by the respective team (or, worse, many teams with their own tooling), security teams commonly have little insight into these infrastructures, into how these are secured, and what is happening in them. Moreover, DevOps teams do not always include security experts; this causes a common gap in security management and governance.

To make matters even worse, there is a shortage of solutions that secure the entire SDLC. Across the pipeline, we find a broad attack surface that frequently isn’t well protected. Attacks can be executed against SCMs, build systems, package repositories, or the operations infrastructure such as the tenants in IaaS (Infrastructure as a Service) clouds, that frequently aren’t configured in an appropriately secure manner, and tend to utilize massively over-privileged service accounts.

The automation and interconnectedness of the DevOps approach boosts productivity but also increases the danger of lateral movement within the pipeline. For example, a compromised

developer account with access to the SCM could inject and sign code, and then use GitOps or everything-as-code to push the build and deploy it into production. Moreover, SCMs are just one element of the SDLC, and attacks can happen anywhere along the pipeline.

While there are tools for securing code and applications, such as Static and Dynamic Application Security Testing (SAST/DAST) and WAF (Web Application Firewalls), and API Security and Management for protecting applications, there is the aforementioned gap of solutions that affects the full SDLC and the entire DevOps pipeline.

Organizations developing software must understand and address this challenge. Organizations must also understand that software they procure must not be trusted blindly. The Zero Trust principles must be applied to software as well, i.e., verification instead of trust. Within DevOps organizations, this requires adequate organization, security controls and governance, and security tooling. For software that is purchased, similar security controls and governance must be put in place to mitigate risks from software supply chain attacks.

Within the broad scope of securing SDLCs and mitigating risks of software supply chain attacks, addressing the challenge of code tampering, i.e., code tampering prevention and detection, is an essential element. All attacks are, finally, about tampering with code by injecting malicious code.

Secure SDLCs and where Code Tampering Prevention comes in

Securing the SDLC (Software Development Lifecycle) involves code tampering prevention as one of the key measures. If attackers manage to maliciously alter code, then the impact can become massive, for both the organization itself and its customers.

Simply speaking: Code tampering is at the core of software supply chain attacks. Attackers look to inject malicious code to propagate their attacks downstream to the subsequent systems where the software is deployed. Thus, code tampering prevention is an essential element of every approach for increasing Software Security as well as Software Supply Chain Security.

Success in the Digital Age: Security as Differentiator

Success builds on competitive differentiation by delivering unique & secure digital services

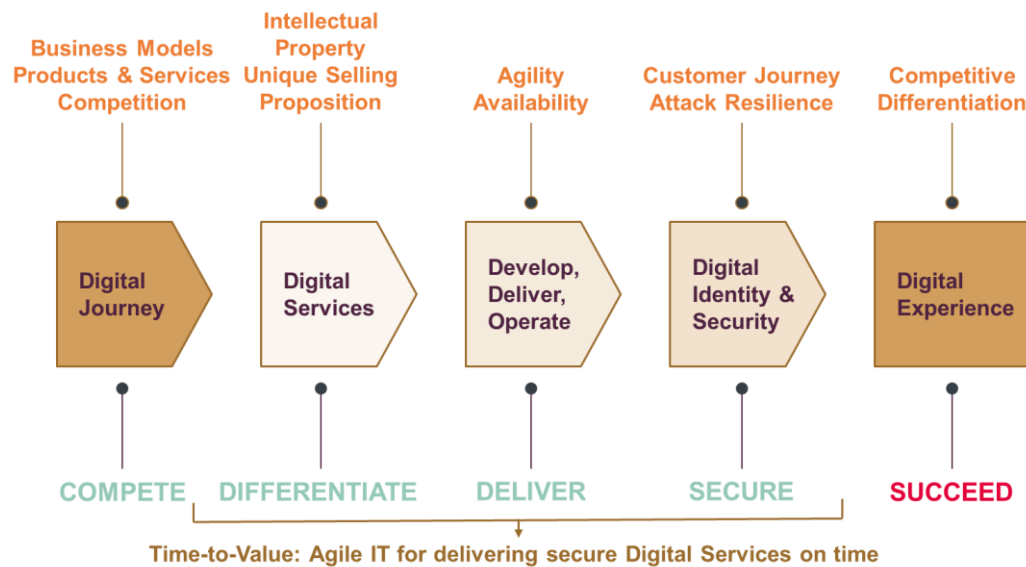


Figure 1: Digital Business builds on software. Security of software is a key success factor in the Digital Age.

In the digital age, software has become a differentiator in competition. Developing unique digital services is at the core of today's business strategies. This requires on one hand the ability to develop, deliver, and operate such digital services fast and in an agile manner. On the other hand, digital services have shifted to the core of the value chain. If they fail, the business is at risk. Thus, attack resilience is key to success in the digital age, and attack resilience starts with secure SDLCs.

In the digital age, digital services build the interface between businesses and their customers. Attacks on the software of these digital services can massively harm businesses.

Since the software is key to the success of digital business, organizations must implement proper code tampering prevention to protect themselves, their business model, and their customers against attacks. These attacks can come from both internal and external attackers.

Code tampering prevention can happen in many places in the SDLC, including at the coding, build, and production phases. The variance in ways code can be altered and locations where code can be tampered with, makes it a more difficult problem to tackle. In fact, there is not a single, simple countermeasure that can prevent or detect code tampering, instead It requires both technical and organizational measures. The latter starts with rethinking the common organizational structure in which DevOps teams are largely isolated from the security and infrastructure teams. There is a need for more collaboration, as there is a need for implementing controls and proper governance for the entire SDLC.

Other organizational measures involve 3rd party Software Supply Chain Risk Management, which focuses on mitigating risks that come through the re-use of software. Today, most software heavily utilizes libraries and other components, either open source or from other software companies. Understanding what is consumed and which risks might be injected by the use of 3rd party

components is an essential element, as well as proper patching and updates when, e.g., new versions of libraries are released. However, all these measures can only reduce risk, but risks inherited by relying on 3rd party components will always bear a significant level of risk, because neither the SDLC of 3rd parties nor the code are under (full) control.

How to do Code Tampering Prevention right – multi-layered & defense-in-depth

Code tampering prevention is not a single measure, but a set of actions. It involves critical code monitoring and file integrity verification along the entire CI/CD pipeline as well as employing a defense-in-depth approach by securing all tools involved, and continuous anomaly detection.

To implement code tampering prevention, multiple elements are required, including the following:

- Critical Code Monitoring
- File Integrity Verification
- Defense in Depth
- Anomaly Detection

Critical Code Monitoring

Critical code monitoring starts with understanding which code is critical, and then applying continuous monitoring to raise an alert if there is any change to this code. This also requires a defined process for handling such alerts. Moreover, this approach needs to be targeted to code that is immutable or experiences infrequent change in order to avoid false positives.

Code with high criticality includes CI/CD settings, branch protection rules, infrastructure-as-code, dependency updates, and build rules that trigger security scans. These are some of the areas that are of utmost criticality, given that attackers can utilize them to gain more control over the SDLC or, in the case of infrastructure-as-code, over the runtime environment.

Code tampering prevention is not a single measure, but a set of related activities that span the entire process from code creation to the runtime environment.

By having a monitoring process in place, changes can be immediately identified and reviewed, mitigating the risk of critical code becoming corrupted. Using critical monitoring judiciously, ensures false positives don't become a problem.

File Integrity Verification

File integrity verification, as a second measure, targets the entire SDLC and focuses on mitigating the risk of malicious code being injected or code being altered at any stage of this process. This involves a number of measures:

- Commit Signing - Commit signing is a capability where the one committing new code signs this code with a private key. It is performed ahead of the build process, and it acts as a form of authentication for the commit of new code.
- Build Reproducibility - The build process should be reproducible, so that verification of a correctly working, non-tampered with build system can be done, i.e., ensuring that no malicious code snippets have been entered at that stage of the SDLC.

- Build output - Build outputs should be verified against published and operational artifacts.

Taking these measures will help in reducing the risks of code tampering. But again, such an approach is not just based on a technical solution, but requires an adequate organization, processes, and governance to work well. If security steps and measures are ignored, the risk remains. Solutions that automate these processes and ensure that security controls are correctly implemented and enforced are essential to an effective code tampering prevention solution.

Defense in Depth

The term defense-in-depth focuses on all the measures for hardening the CI/CD pipeline, such as:

Least privilege enforcement - The consistent enforcement of least privilege access across all tools should be a standard procedure. However, this activity is rarely consistently implemented and enforced, and often on an irregular cadence. This reduces access to those who actively work on those code bases, and thus limits the pool of candidates that could tamper with that code.

Tool hardening - the configuration of the various tools must be hardened, by, e.g., implementing strong version control policies and branch protection rules, moving to restrictive security settings, and using trusted component registries. Unfortunately, most tools in the SDLC aren't deployed by default with security in mind, instead they are optimized for efficiency of use. This means security teams must actively harden the configuration settings of these tools to prevent them from becoming attack vectors.

Security Policies - this includes a central management of policies, including defined processes for creating, approving, implementing, and regularly reviewing policies, the consistent enforcement of such policies across all elements of the SDLC and for all DevOps teams in the organization, and, last but not least, auditing.

Authentication - security teams must implement consistent access controls such as strong, multi-factor authentication. It is essential to ensure that passthrough authentication is configured properly where used, to ensure that there are no weak links through insecure authentication at some of the stages within the SDLC. This helps reducing the chances of developer credential compromise, and thus reduces the risk of those credentials being used to tamper with code.

Technology alone is not sufficient for protecting the SDLC and preventing code tampering. Successful approaches combine organizational and technical measures.

The technical measures must be complemented by organizational measures such as peer code review for commits, with specific focus on security risks, and a good human resource management that also looks at personal integrity and a developer's satisfaction in the job.

Anomaly Detection

A final element is required for effective code tampering prevention measures: anomaly detection. There are many areas where this can take place, for example:

- Requests and grants for access that deviate from regular access
- Uncommon configuration changes in tools and code
- Deviation from defined workflows

- Developers touching code they have never touched before
- Developers writing to new parts of the codebase
- Behavioral changes in members of the development team, particularly those who are about to leave the team

Processes for anomaly detection require careful setup and execution. Developers might just be shifting to a new task and thus working on code they haven't touched before, or be contributing code to new parts of the codebase. Such changes are not necessarily malicious, in contrast: most changes are just new and different, but still normal behavior. Effective anomaly detection highlights potential areas of behavioral deviation which security teams may need to pay attention to, without a high degree of noise.

The Cyscale Approach to Code Tampering Prevention

Cyscale is a software vendor that delivers a comprehensive software supply chain security solution. This solution includes various components that help in code tampering prevention and thus can form a central element within a modern application security strategy.

Cyscale specializes in complete software supply chain security, from code to cloud. Cyscale's approach to code tampering combines verification of integrity throughout the entire SDLC with the implementation of consistent security controls in the tools that make up the SDLC. This multi-pronged approach is designed to detect code tampering attempts; and to reduce the risk of developer accounts and DevOps tools and infrastructure being improperly used to facilitate code tampering.

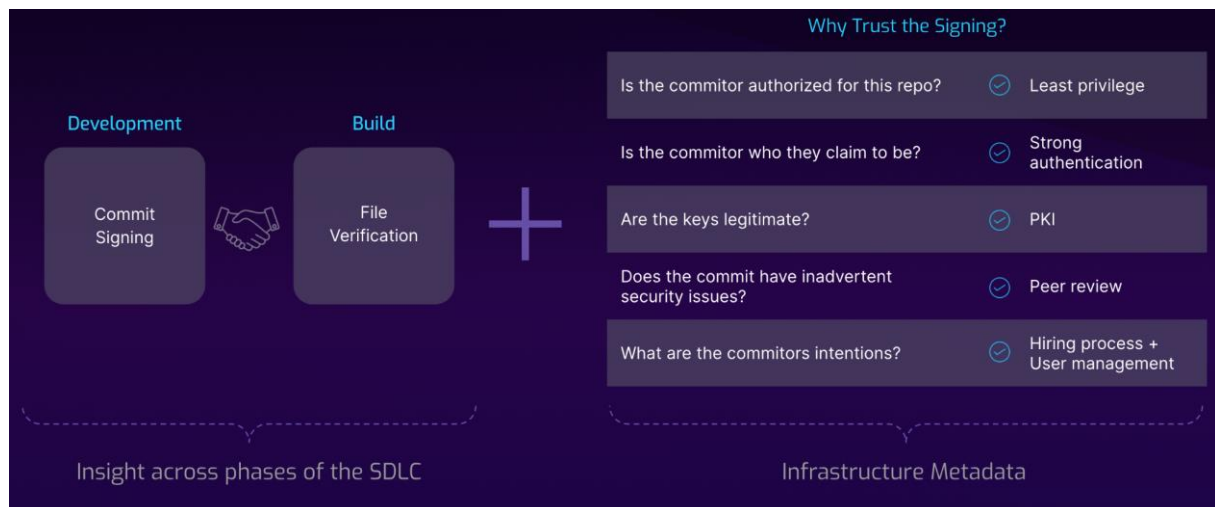


Figure 2: The Cyscale approach on Code Tampering Prevention [Source: Cyscale]

Cyscale provides central policy management to ensure consistent security and governance is in place across all tooling that makes up the SDLC. These policies cover various areas such as:

- Access controls such as MFA settings and IP restrictions.
- Least privilege, focusing on stale user & repositories as well as excessive privileges.
- Branch protection rules like requiring peer code review.
- Build rules, avoiding hardcoded secrets and enforcing file integrity checks.

These policies are managed centrally and can be enforced at all stages of the SDLC, from SCM environments to build tools, infrastructure-as-code, and the cloud tenants where applications finally run.

Moreover, Cymcode is built upon a knowledge graph that correlates and connects data from different tools and phases of the SDLC. This enables the platform to easily compare signed code from the build phase, with the output of the build and production phases, thus verifying code and release integrity.

A specific strength is in the consistent approach of managing security with policies across all components, and across all functional areas. With code tampering prevention affecting all stages of the SDLC, and requiring a range of intertwined activities, such a central approach that unifies activities helps in enforcing code tampering in a holistic manner.

The broad support for target environments supported by Cymcode allows for managing everything from the SCM to the runtime environments consistently. One of the examples where such an approach is mandatory is file integrity checking, which must not be limited to the source code, but cover, as aforementioned, build and deploy stages as well.

Within the portfolio of capabilities that the Cymcode platform delivers, many are directly supporting code tampering prevention. This includes, amongst others, features such as:

- File integrity checks
- Branch protection
- Access controls and policies

These are all elements of a comprehensive approach to code tampering prevention.

Recommendations: Securing your SDLC, securing your customers, securing yourself

Organizations must act on software supply chain security. This involves software they procure as well as software they develop themselves. With regard to software development, securing the whole SDLC (Software Development Lifecycle) is essential for delivering secure software to customers, for protecting customers, and for protecting an organization's own digital business.

Securing the SDLC will require tools, but also organizational measures. These involve:

- An integrated perspective and defined accountabilities and responsibilities for the various DevOps teams within the organizations, avoiding "wild" software development that is not integrated into the security and governance measures.
- Communication and collaboration between SecOps, the security teams, and the infrastructure teams, with defined processes, interfaces, accountabilities and responsibilities.
- A governance approach for defining, managing, approving, and enforcing policies, for setting and enforcing controls, and for regular audits.

For dealing with external parties that procure software to the organization, a well-defined approach on C-SCRM (Cybersecurity Supply Chain Risk Management) must be implemented, that enforces a secure SDLC at the suppliers.

Technical measures for a secure SDLC involve a range of measures, with code tampering prevention being at the forefront of these measures. To successfully implement code tampering prevention, it first must be understood which tools and DevOps teams are in place. Changes must be well-managed, to, e.g., apply hardening to new tools in the DevOps tools chain.

Ultimately, implementing code tampering prevention involves a lot of change management and explanation about why certain measures are required, but also solutions that help in implementing and enforcing these measures without inhibiting the efficiency of developers.

Related Research

[Maturity Level Matrix for Cyber Security](#)

[Integrating security into an agile DevOps paradigm](#)

Copyright

©2022 KuppingerCole Analysts AG all rights reserved. Reproduction and distribution of this publication in any form is forbidden unless prior written permission. All conclusions, recommendations and predictions in this document represent KuppingerCole's initial view. Through gathering more information and performing deep analysis, positions presented in this document will be subject to refinements or even major changes. KuppingerCole disclaim all warranties as to the completeness, accuracy and/or adequacy of this information. Even if KuppingerCole research documents may discuss legal issues related to information security and technology, KuppingerCole do not provide any legal services or advice and its publications shall not be used as such. KuppingerCole shall have no liability for errors or inadequacies in the information contained in this document. Any opinion expressed may be subject to change without notice. All product and company names are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

KuppingerCole Analysts support IT professionals with outstanding expertise in defining IT strategies and in relevant decision-making processes. As a leading analyst company, KuppingerCole provides first-hand vendor-neutral information. Our services allow you to feel comfortable and secure in taking decisions essential to your business.

KuppingerCole, founded in 2004, is a global, independent analyst organization headquartered in Europe. We specialize in providing vendor-neutral advice, expertise, thought leadership, and practical relevance in Cybersecurity, Digital Identity & IAM (Identity and Access Management), Cloud Risk and Security, and Artificial Intelligence, as well as for all technologies fostering Digital Transformation. We support companies, corporate users, integrators and software manufacturers in meeting both tactical and strategic challenges and make better decisions for the success of their business. Maintaining a balance between immediate implementation and long-term viability is at the heart of our philosophy.

For further information, please contact clients@kuppingercole.com.