kuppingercole ANALYSTS

Commissioned by:

BEYOND IDENTITY

Secure Software Supply Chains

Paul Fisher

January 4, 2023

Major cyber-attacks such as the SolarWinds and Kaseya incidents demonstrate the need to focus significantly more on software supply chain security as well as traditional cyber defense areas. Avoiding the code tampering that occurred in both of those attacks by criminals and internal parties is essential. This whitepaper looks at how to increase security throughout the Software Development Lifecycle and implement a multi-layered, defense-in-depth code tampering prevention and detection strategy.

Beyond Identity was founded in 2019 and offers identity and authentication solutions in three critical operational areas: workforce ID management, customer ID management and DevOps ID management which assists in software security and code provenance in the Software Development Lifecycle.

Contents

Figures

Introduction

> *The term "Software Supply Chain Security (SSCS)" refers to the ability to secure the software development lifecycle (SDLC) process throughout the development, testing, deployment, and maintenance phases – at every point along the way, including along the whole CI/CD pipeline.*

Industry awareness of software security has increased significantly since the end of 2020 due to two major attacks on software supply chains. 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 much more complex over the past few years. It does not just affect code running as applications, but also building blocks such as Infrastructure as Code (IaC) and the newer trend of Everything as Code (EaC). 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 – including code repositories. 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.

Highlights

- Existing security policies, tools and practices are not adequate for preventing supply chain attacks
- Vulnerability Management software cannot detect supply chain attacks, which exploit trusted software artifacts rather than traditional vulnerabilities
- Established CI/CD and DevOps pipelines rely on implicit permissions to enable rapid deployment, implementing security controls at the end of this process
- We need new protective methods built to address the unique characteristics of supply chain attacks and to shift security left
- Software supply chain attacks are a growing problem
- Attackers have struck a rich and relatively unprotected seam and will continue to develop this form of attack

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 the distribution of malicious software to their customers. By tampering with commercial off the shelf (COTS) software, attackers managed to multiply their attacks and gain access to thousands of other organizations.

Such is the demand for rapid delivery that code can be deployed with errors added after the original clean code was committed by the original, authorized developer. Those shipping code to production have no way of knowing if that code is original or has been modified with possible errors or vulnerabilities added by malicious actors. To counter these risks, organizations are looking to add a security layer within DevOps and other coding structures that limits access to code repositories and software lifecycles only to authorized and authenticated identities within the organization.

DevOps and coders are the new kings

Business management likes DevOps – they get things done. DevOps produce code, applications, and cloud-based services in response to demands from other lines of business. While DevOps took its name from the coming together of traditional Developers and Operations team practices to work towards a common goal, these days DevOps has become a catch-all term for various team structures that are responsible for the writing, testing and deployment cycle of code within an organization – delivering a continuous internal software supply chain that the organization feeds on.

While the structure and hierarchy of DevOps teams will differ from one organization to another, the common theme of all DevOps teams is speed, automation, and reliance on cloud infrastructure. Increasingly those writing code are also responsible for code testing and committing code to popular repository tools such as GitLab, GitHub, and Bitbucket for other developers or deployment teams to pick up.

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

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 often 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.

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 worse, there is a shortage of solutions that secure the entire SDLC. 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 (EaC) to push the build and deploy it into production.

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 gap of solutions that affects the full SDLC and the entire DevOps pipeline.

Organizations developing software must understand and address this challenge and also 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 Code Tampering Prevention

> 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.

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.

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.

Software at risk includes:

- Open-source code
- Third-party libraries
- Commercial software deployed
- In-house software deployed
- Software sold to customers
- Unaudited software and cloud purchased by LOBs

Four steps to getting code security right

> 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.

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 three important measures:

- 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.
- 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 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 adequate organization,

robust 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:

- 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.
- 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.
- 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.
- 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.
- Tech 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. 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.

Code security with Beyond Identity

> Beyond Identity is a young company (2019) whose founders have a background in Secure Sockets Layer (SSL) technologies, used by Certificate Authorities (CA) to establish secure connections, mostly across the web. It has developed this technology to improve Software Supply Chain Security.

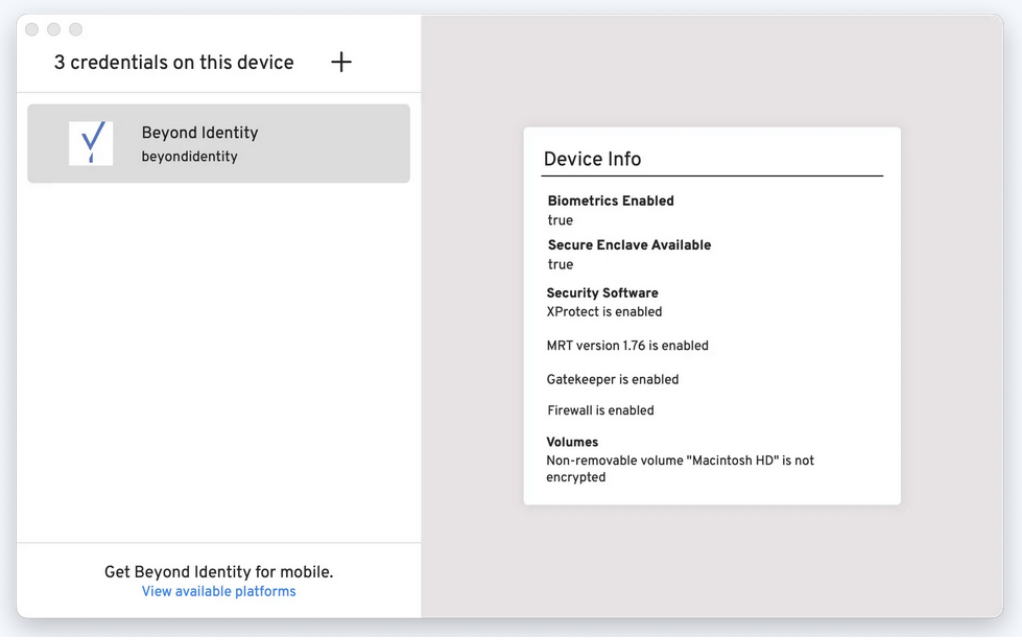

Figure 1 The Beyond Identity Authenticator admin console can monitor device state (*Beyond Identity*).

The founders took the concepts behind SSL/TLS and developed a passwordless authentication stack to create a trusted connection between identities, devices, and resources within IT infrastructures. This is known as the Beyond Identity Authenticator and the company has developed this core technology to underpin three distinctive identity

management products for separate applications. The products are Beyond Identity Secure Workforce, Beyond Identity Secure Customers and the focus of this Whitepaper: Beyond Identity Secure DevOps (SDO).

Beyond Identity Secure Workforce prevents password-based breaches by ensuring continuous user and device trust and eliminating passwords. Instead of relying on passwords and phishable factors, Beyond Identity creates a public-private key pair where the private key is generated, stored, and never leaves the secure enclave of a user's trusted device. Every authentication request is processed using certificate-based authentication, with no central certificate authority (CA) needed, and fine-grained user and device security signals for risk-based access policy evaluation. For the end-user, there is no password, one-time code, push notifications, or second devices required.

This architecture allows organizations to achieve frictionless zero trust authentication with phishing-resistant MFA, strong assurance of user identity and device security at login and continuously, and dynamic access policy enforcement based on real-time risk.

The private key securely stored in the TPM cannot be viewed or removed by anyone. The process allows logins entirely through an encrypted process with user and device identities stored in hardware and should in theory allow for faster logins as there are no additional default steps such as One Time Password (OTP) or separate texts needed.

Another advantage of this approach is that it is device agnostic, making it suitable for many Work from Home (WFH) and remote working scenarios. The platform can verify the user and device at the same time, so it does not matter if the device is not officially issued as the user is authenticated through their original registration with the Beyond Identity Authenticator, allowing the users to also use and add unmanaged devices. Configurable policies control whether users are allowed to add devices, and if so, which types of devices can be added by users.

Beyond Identity, by default, captures 25+ user and device "risk" signals from any device making a login request, these signals can be used to create security policies for end users and their devices. This allows for flagging risky or misconfigured devices as well as unusual behavior. Escalation requests can be applied, which, for instance, can be a further security step the end user has to take such as biometric identity check from the device. These checks also serve to see if the device is running out- o- date software or missing patches.

Beyond Identity Secure DevOps

Malicious code or even just buggy code is an increasing risk to enterprise software supply chains as expected code commit and deployment targets have been shortened

considerably. Beyond Identity Secure DevOps uses the Beyond Identity Authenticator technology to ensure that only valid code is pushed into CI/CD pipelines and that signatures are regularly checked for authenticity.

The platform focuses on code integrity by managing who changes code and ensures that code signature is linked to the human identity and, critically, the device identity. Many companies don't yet sign the code with both identities, while others do not have oversight of the provenance of code at all. The real security test is when code is shared to popular code repository tools such as GitLab, GitHub, and Bitbucket and picked up by deployment teams.

Developers must install the Beyond Identity Secure DevOps software to their device and enroll with the Beyond Identity Cloud. The platform then uses APIs to verify that signatures used to commit code are officially issued and registered by Beyond Identity and can be authenticated.

Added to this is a useful administrator console that enables managers to control which (user) devices can create keys and allows admins to block keys if a pattern of suspicious behavior is detected, as described earlier.

Developers do not sit in cubicles or crowded in rows anymore. Modern life and technology allow these important employees to develop from anywhere, and on multiple devices. This is good for them and good for the business, so we like that. Developers also have access to multiple open-source tools used with the blessing of their own management, but all this good stuff can involve risk. Beyond Identity has identified this risk and applied its Authenticator platform to go some way toward plugging the security gaps in modern, distributed DevOps and CI/CD environments.

The simplicity of the Beyond Identity Authenticator makes it perfectly suited to the speed and pressure of coding environments. A one-time install and registration process is all that is needed for developers to authenticate their code on a ny device they are using. This is enhanced by a similarly simple process when the code is sent to GitHub or similar coding repository. The process reduces the risk of bogus or bad code being entered into the software supply chain.

It is not yet perfect; it is possible that an authorized device can be hijacked by an unknown user if they have access to the username as the system does not require a password or OTP by default, unless set up by the administrator. There is also yet no support for private Git repositories, but Beyond Identity says this is in their roadmap. Overall, this splendidly simple solution goes some way to balancing the Convenience - Security - Authentication (CSA) parameters for this specific but important enterprise environment.

Recommendations

Organizations must act urgently on software supply chain security. This involves software they procure as well as software they develop themselves. 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. This involves:

- 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.

Code tampering prevention should be at the forefront of creating a secure SDLC. 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. Remember the following tips:

- IT Security teams need to work with Dev and coding teams, understand their techniques and how they use source code
- Do not blame the developers for supply chain attacks
- Investigate SBOM (Software Bill of Materials) as an approach, it may soon be written into new compliance regulations
- Investigate secure endpoints and remote workflows by developers and others
- Ensure that securing the software supply chain and processes does not limit Devs productivity and creativity

Related Research

Leadership Compass: Privileged Access Management for DevOps
Advisory Note: Integrating Security into an Agile DevOps Paradigm
Leadership Compass: CIEM Dynamic Resource Entitlement
Leadership Compass: Access Management 2022

Leadership Compass: Container Security

Copyright

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.