Securing the Software Supply Chain

From CI/CD Security Risks to Protection Strategies

Introduction

Navigating the Landscape of Software Supply Chain Security

In the rapidly evolving world of software development, the security of the software supply chain has emerged as a critical concern. The attack surface for potential adversaries has expanded dramatically as we increasingly rely on open-source components, automated processes, and cloud-based services. The software supply chain, once a peripheral concern in cybersecurity, has now become a primary target for sophisticated attackers.

There has been an astonishing 742% average annual increase in Software Supply Chain attacks over the past 3 years. Furthermore, the estimated financial impact of software supply chain attacks are expected to surpass $80.6 billion by 2026, representing a substantial 76% rise compared to the estimated losses of $45.8 billion in 2023.

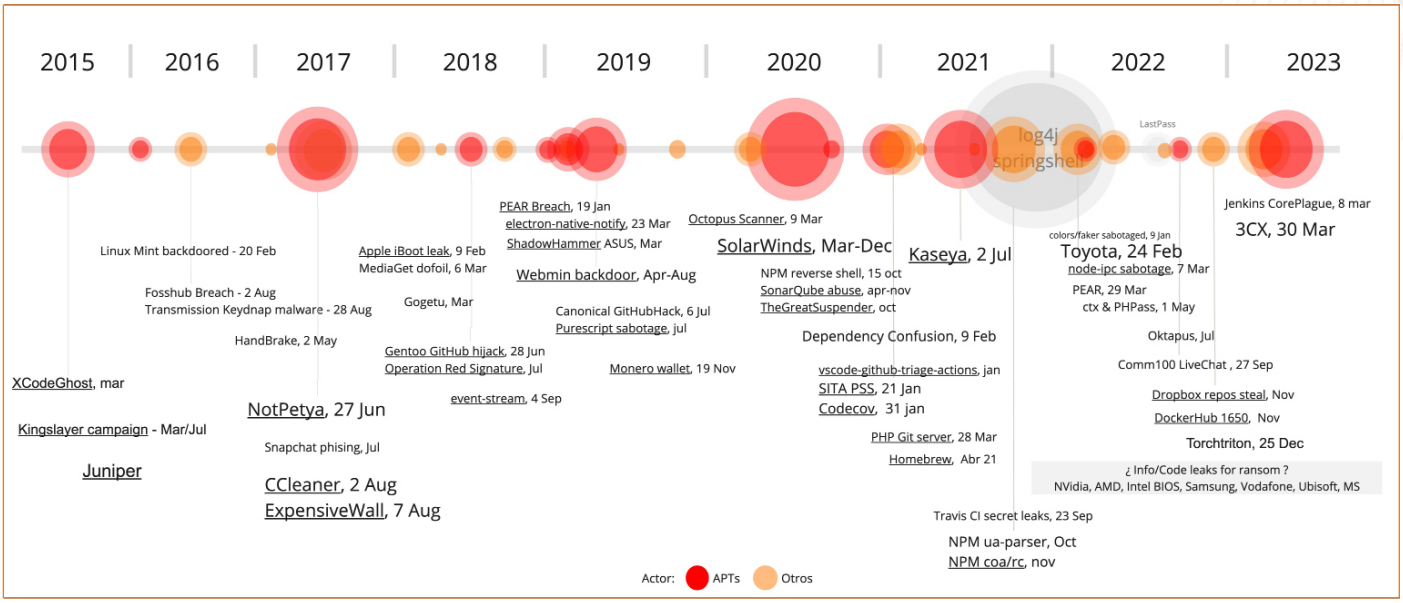


Image: Software supply chain attacks since 2015

Introduction

Companies and organizations should be aware of the software supply chain's threats and the strategies they can employ to secure it.

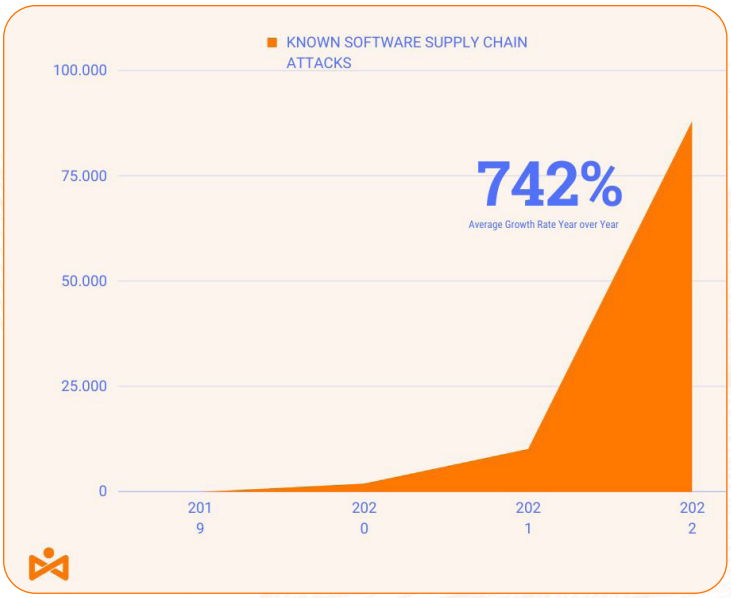
The software supply chain encircles the entire process of building and deploying software, from the initial development of code to its deployment in production environments. This process has undergone significant changes in recent years, with the advent of practices such as Continuous Integration/ Continuous Deployment (CI/CD) and the widespread adoption of open-source components. These developments have brought numerous benefits, including increased efficiency and flexibility, but they have also introduced new vulnerabilities that attackers can exploit.

$80.6 Billion/year

is

The Annual Projected Cost of Vulnerable Software Supply Chains by 2026

One of the key shifts in the threat landscape has been the move from attacking the software product itself to **targeting the processes and tools used to build and deploy that software**. Several factors, including the improved security of software products and the increased use of automation and cloud services in the software development process, have driven this shift. As a result, attackers are now focusing their efforts on infiltrating build systems, code repositories, and open-source packages, with CI/CD systems being a desirable target.



The security of the software supply chain is not just about protecting individual components. It's about securing the entire process. This requires a comprehensive approach that addresses every link in the chain. It's not enough to secure the code itself; we must also secure the tools and processes used to build, test, and deploy that code.

In this ebook, we explore the specific threats facing the software supply chain and explore strategies for mitigating these threats. We will **examine the tactics attackers use to infiltrate CI/CD systems and other parts of the software supply chain and enumerate the measures we can take to protect against these tactics**. We will also look at the broader context of software supply chain security, considering how changes in software development are influencing the threat landscape.

It's important to remember that the security of the software supply chain is not a static goal. The threats are constantly evolving, and the strategies for securing the software supply chain must evolve with it. This book is intended to provide a roadmap for that journey, offering insights and guidance to help you secure your software supply chain against the threats of today and tomorrow.

CHAPTER 1
DevOps and Risk in CI/CD

The Advent of DevOps and Associated Risks

The advent of DevOps has revolutionized the software development industry, enabling faster, more efficient development and deployment of software. Following an Atlassian report, companies that adopt DevOps principles report an average of 45% higher customer satisfaction, 43% higher employee productivity, 41% improvement in defect rates, and 38% less IT-related costs. However, with this evolution comes new risks, particularly in the realm of Continuous Integration and Continuous Delivery (CI/CD).

CI/CD systems form a crucial part of modern software infrastructure. They automate the process of integrating changes from multiple developers and deploying the software to production environments. However, the security of the software supply chain is often overlooked, leading to an unacceptable level of risk.

Attackers are well aware of these trends and have adapted their tactics to target software development processes in the organizations. The shift towards automation has seen developers taking over more activities beyond coding and unit testing while security considerations are moving to earlier stages in the development process. While this has led to improvements in the security of the software product, it has also opened up new avenues for attackers.

Cloud and Open Source: the Changing Landscape of Software Development

How software is built has changed drastically over the past 5-10 years. Most software is now built using open-source components and tools, with many processes triggered automatically when a change in source code is uploaded. The infrastructure where the software is deployed is primarily virtual, and cloud-based systems are the norm. These systems involve numerous infrastructure, connectivity, and security configurations. **Misconfigurations** can greatly impact the security posture of your CI/CD pipeline and make it susceptible to cyber-attacks.

Rapid development cycles lead to the inclusion of insecure code or the improper integration of third-party components. It introduces vulnerabilities into the CI/CD pipeline and significantly increase the attack surface. Without a proper third party auditing and monitoring, development teams could integrate **components or libraries with malicious code** that could be exploited in any customer after the release delivery.

The combination of open-source components from unknown origins and cloud-based systems has significantly increased the attack surface, not only on the software product itself but also on the build and deploy pipelines.

1. DevOps and Risk in CI/CD

Key Security Risks in CI/CD Systems

Supply chain attacks target weak parties within the supply chain to breach others connected to the chain. Pipelines often use third-party integrations. However, **misconfigurations in the integration** or improper usage of these services can introduce security weaknesses into the pipeline that could provide undesired access level to attackers.

If any actor injects **malicious code or commands in the build or delivery pipeline** configuration could execute any malicious actions during the build process and even take benefit of insufficient Pipeline Based Access Controls (PBAC) to move laterally in or outside the CI/CD system. Enforcing pipeline protection and permissions is fundamental for the CI/CD security and to ensure the integrity and security of our releases.

Related to permissions management, it is also essential to control the **level of permissions that user accounts have in CI/CD**. Stale accounts are common, and highly privileged user accounts are used for machine operations. When running in Source Code Management (SCM) systems, the access token often can do things unnecessary for the build, and attackers leverage these excessive privileges.

"Misconfigurations, malicious components, and insecure tools and integrations introduce vulnerabilities into the CI/CD pipeline. Malicious actors can exploit these vulnerabilities, injecting code or gaining unauthorized CI/CD process access. Automation can introduce threats at scale. Poorly managed secrets and errors in Infrastructure as Code (IaC) can lead to further threats. To mitigate these risks, regular scanning, properaccess controls, secure configurations, and careful management of third-party components are essential security practices."

Automation: Introduction of threats at scale

Automation itself is perhaps the biggest risk. The primary goal of automation in DevOps is to speed up the software development process. However, this emphasis on speed can lead to security being overlooked. For example, automated processes might **deploy elements that haven't been properly reviewed for security threats**.

Automated processes often lack the manual oversight that catches subtle or complex security issues. Automated tools are excellent at catching many types of vulnerabilities, but may not be as effective at identifying complex and new security issues. Thus, due to automation and the large number of pipelines run daily, the **attacker's activity could pass unnoticed for a long time.**

The automation **tools** themselves can be a source of security risk if they are **not properly configured and secured.** If a CI/CD system is compromised, it could be used to introduce malicious code into the software being developed. It is critical to enforce a proper set of permissions and configuration of all DevOps infrastructure.

Automation requires the **use of secrets** such as API keys or passwords. These elements cannot be hashed one-way like passwords in the end system; they need to be stored encrypted for recovery in cleartext at build execution. If these secrets are not properly managed —for example, if they are hard-coded into scripts or stored without encryption— they can be a significant security risk.

Finally, **Infrastructure as Code (IaC)** is a key component of many CI/CD pipelines, allowing for the programmable and automated setup of environments. However, errors in IaC scripts or configurations can lead to insecure environments, misconfigured services, and other vulnerabilities at scale.

CHAPTER 2
The Shift in Attack Tactics

Evolution of Cyber Threats and the Rise of Application Security

As the software development landscape has evolved, so have the tactics employed by malicious actors.

Over the past two decades, software security has improved significantly. Application Security (AppSec) has gained traction, leading to a focus on avoiding vulnerabilities in software introduced unintentionally during design or coding. This has resulted in the rise of Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), and Software Composition Analysis (SCA). However, as the doors to application or product vulnerabilities have closed, **attackers have found new paths to exploit.**

The Shift to Software Construction and Delivery Chains and Tactics Employed by Attackers

Instead of attacking the product, i.e., applications and software systems deployed in production, which are better protected, attackers have shifted their focus to the **software construction and delivery chains** themselves. This shift has seen the "arms race" move to the software pipelines arena. Attackers have changed their operations to infiltrate build systems, code repositories, and open-source packages. In particular, the CI/CD systems are often considered the "crown jewel" for many attackers.

New generations of Attackers

DevOps Infrastructure is an easy target

68% organizations with DevOps infrastructure have reported increased vulnerability due to inadequate security measures

Large attack surface.

74% of organizations have experienced an increased attack surface due to the growing complexity and the expansion of their SSC

Current security organization priorities

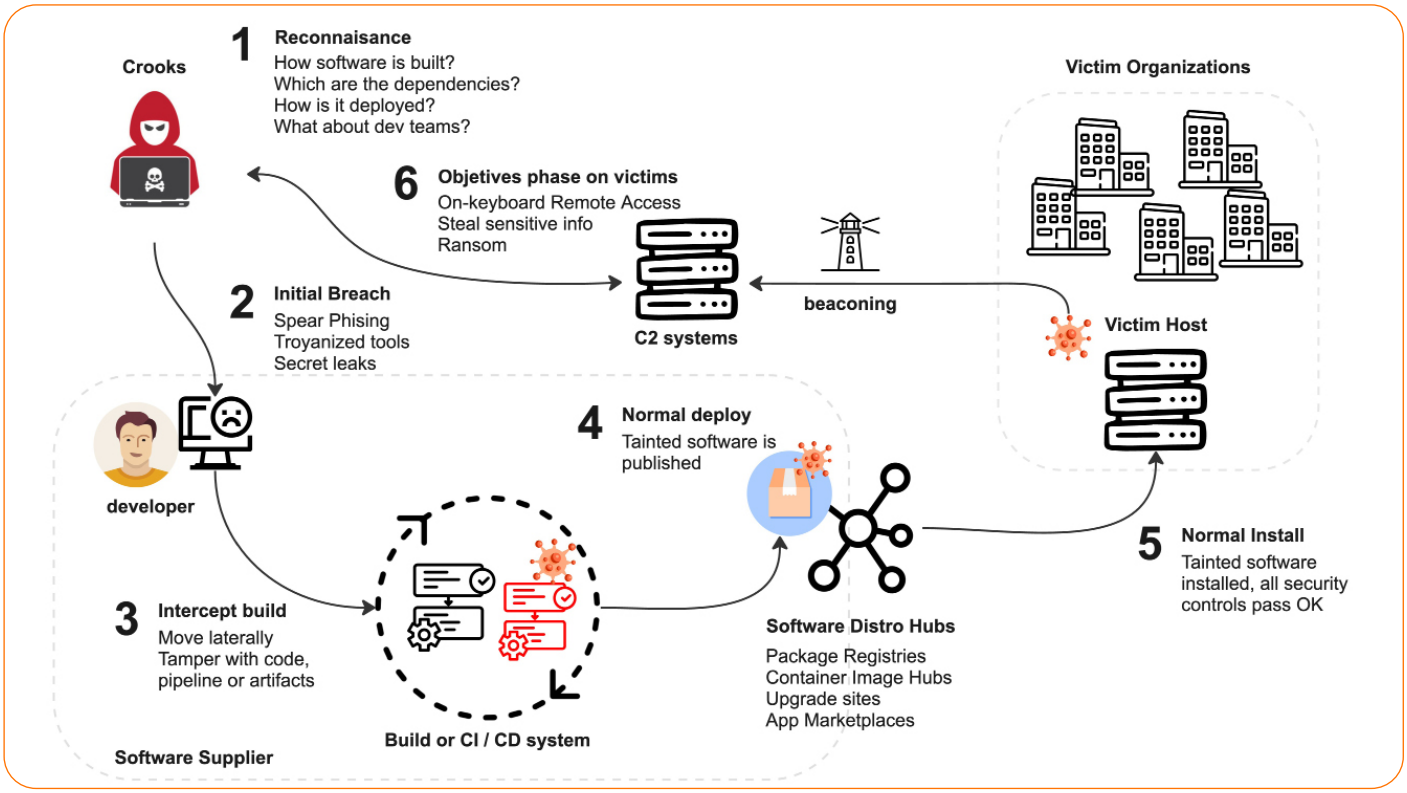
Only 45% of organizations have placed sufficient emphasis on securing their software supply chains, revealing a significant gap in security.

2. The Shift in Attack Tactics

The tactics, techniques, and procedures employed by attackers share many common steps in any IT infrastructure cyberattack. The targets could be a popular third-party component in a public registry, an internal component, or tampering with the software artefacts in a distribution system. However, the targets are often **the build systems and related infrastructure**.

After the common reconnaissance phase, where attackers gain information on how the targeted organization builds and deploys software, the initial breach often goes to a developer or DevOps engineer with access to the infrastructure. These resources are infiltrated using compromised accounts. The attacker may tamper with source code, often during the build process itself, to avoid commits that could be quickly detected.

The malicious software (or 'implant') is usually approved and verified once it has been compiled, meaning that standard authenticity and code integrity checks do not detect any problems. This compromised software is deployed and installed, often directly on customer machines. The users of the software become the victims of this breach. Meanwhile, **the software vendor, whose systems were infiltrated, unwittingly becomes an intermediary**, distributing the malicious behaviour.



The attackers' targets could be a popular third-party component in a public registry, an internal component, or tampering with the software artifacts in a distribution system, although the build systems and related infrastructure are the "crown jewel"

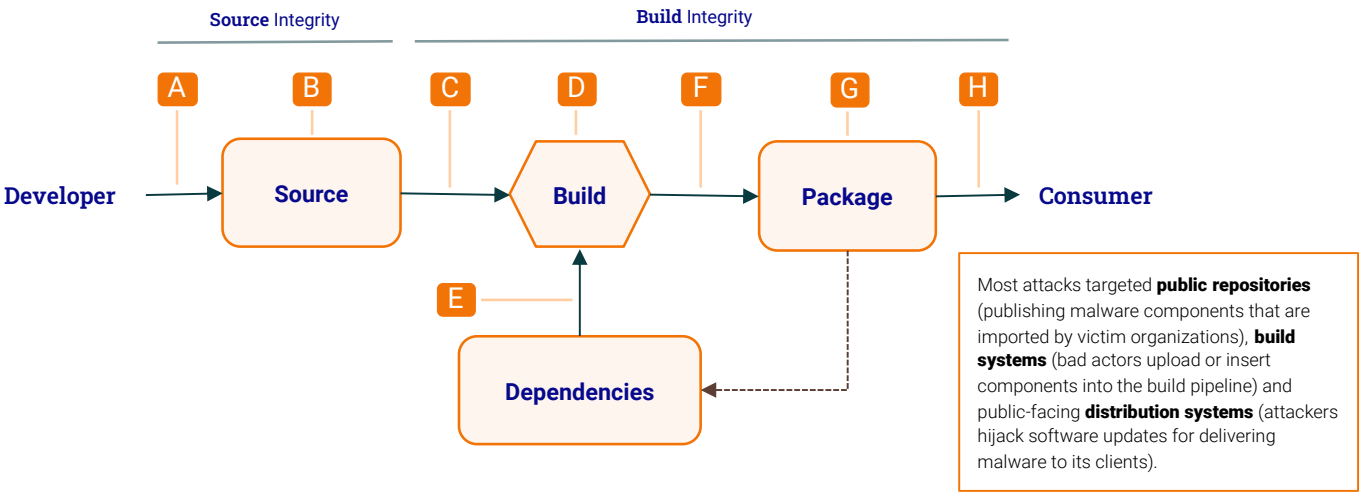
CHAPTER 3

Attack Points in the Software Supply Chain

Understanding the Software Supply Chain and Its Vulnerabilities

Attackers have **tried and succeeded at every single point in the software supply chain**. Source code repositories, developer tools, build and testing tools, package managers, internal and public package registries and artifact hubs, Infrastructure as Code configurations, software upgrade services, and cloud provisioning tools have all been targeted.

SSC Security refers to the practice of identifying and addressing risks in the technologies and processes that are part of SDLC.



A Submit unauthorized change
B Compromise source repo
C Build from modified source
D Compromise build process
E Use compromised dependency
F Upload modified package
G Compromise package repo
H Use compromised package

from "The software supply chain problem, SLSA"

3. Attack Points in the Software Supply Chain

The Future of the Industry: From Unpreparedness to Understanding and Mitigating Threats

The software supply chain is a complex network of processes, infrastructure and tools that are used to build, test, and deploy software. It includes everything from the initial code development to the software product's final deployment.

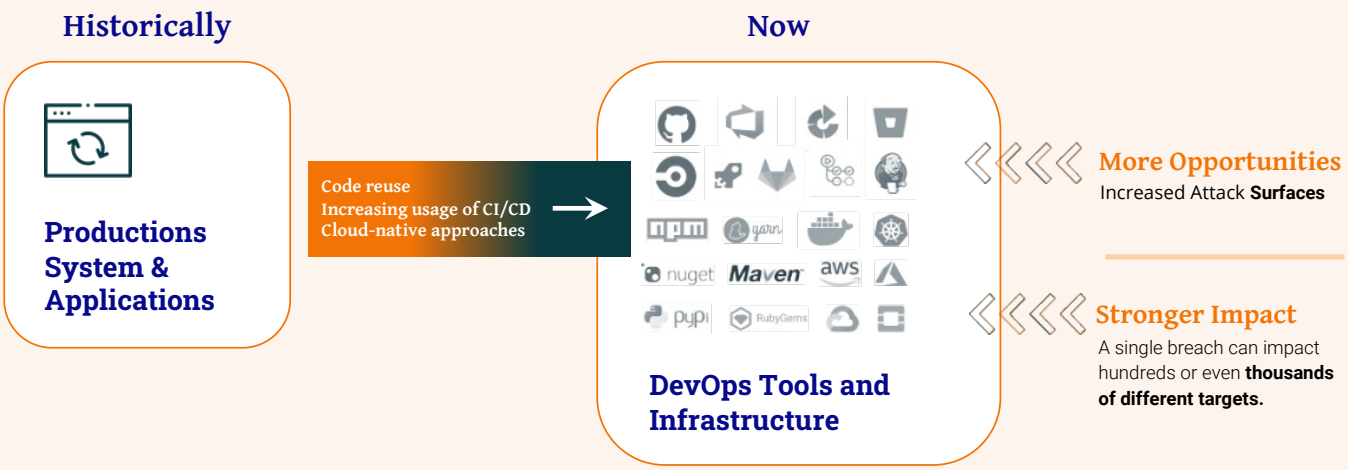
All the software supply chain is under constant attack. The industry is unprepared for these new attack tactics. There are exceptions, but most professionals simply don't know which points hackers leverage to infiltrate a software pipeline.

Developers could be trained on securing software and avoiding the most common vulnerability types, but they do not know what malicious code looks like when performing a code review.

When talking with security professionals and DevOps managers, they are often overconfident that their pipelines are locked against external threats.

Every point in the software supply chain, from source code repositories to cloud provisioning tools, is vulnerable to attacks. Relying exclusively on standard security for apps, and the cloud is no longer adequate to protect all the software supply chain. The industry is largely unprepared for these threats, with many professionals unaware of how hackers infiltrate software pipelines. Despite training on securing software and avoiding common vulnerabilities, developers often don't recognize malicious code. Security professionals and DevOps managers frequently overestimate the security of their pipelines.

Attackers' Priorities are Shifting



CHAPTER 4
Poisoned Pipeline Execution

How malicious actors operate when targeting CI/CD

The approach used depends on the specific tool being targeted, but a common traditional method is known as "log-as-user-then-go-admin." In this method, attackers first gain access to a user account and then attempts to escalate their privileges to gain administrative access. They often try to exploit vulnerabilities in the CI/CD system by using passwords derived from variations of the organization's name. Even with non-privileged access, attackers can still obtain sensitive information such as build pipelines, build logs, and configuration properties. Once they manage to escalate their privileges, it becomes a significant security threat, potentially resulting in a disastrous situation. For instance, in Jenkins, administrators have access to the credential store and the Jenkins script console, which can be exploited to execute remote code or open reverse shells with Groovy scripts.

Poisoned Pipeline Execution is a significant attack where adversaries modify pipeline commands. This type of attack is so critical that it has been given its own name. If adversaries manage to hijack a developer account and there is no review of the pipeline files, a direct change in an existing pipeline or the addition of a new pipeline that attackers may launch, can lead to severe security breaches. This highlights the importance of stringent security measures and regular reviews in CI/CD configura-

It is important to note that attackers have a wide range of tools at their disposal, and sophisticated individuals may even leverage zero-day attacks or develop custom tools to evade detection. However, in many cases, attackers rely on well-known red-team tools like Metasploit during their campaigns. Even with non-privileged access, attackers can still obtain sensitive information such as build pipelines, build logs, and configuration properties. Once they manage to escalate their privileges, it becomes a significant security threat, potentially resulting in a disastrous situation.

4. Poisoned Pipeline Execution

Injecting malicious actions through modification of the pipelines

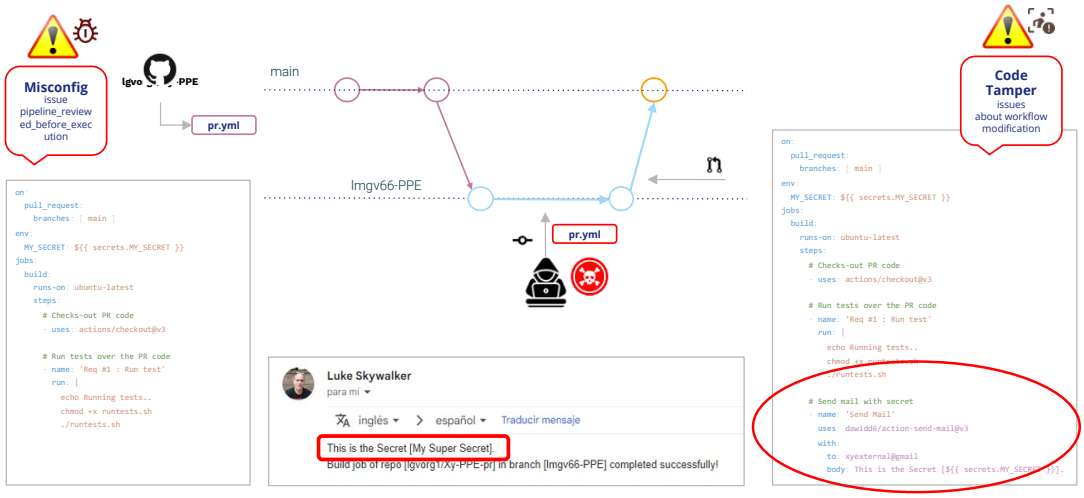
CI/CD security is poorly understood. The technology evolved to make things easy and quick, but continuous integration and deployment tools are complex, with wide attack surfaces, and are highly privileged elements in modern software. They need access to secrets, often with too wide privileges to critical systems for automated building, testing, code signing, registration of components, and cloud infrastructure templates. For cybercriminals, the CI/CD is an appealing game to be collected. Many issues are simply not known.

Cybercriminals frequently manipulate the actual commands executed by a pipeline, a type of attack so significant that it has been dubbed **"Poisoned Pipeline Execution."** Pipelines, which are essentially scripts typically written in YAML or domain-specific languages, are often kept under version control. This aligns with the Anything-as-code approach, where operational scripts are commonly stored alongside the source code. If a malefactor gains control of a developer account and the pipeline files are not reviewed, they can directly alter an existing pipeline or introduce a new one, potentially leading to serious security breaches.

These attackers can indirectly influence the commands executed by altering files utilized in the pipeline steps. They might introduce false configuration files used by testing or security tools to evade automated checks and cover their tracks. They could also exploit the tool extension capabilities to execute unintended code within the CI/CD runner systems. This code injection could occur without needing access to the files under version control. A series of critical errors or misconfigurations could enable an attacker to propose changes via a pull request from a remote branch or fork, bypassing necessary reviews or approvals. This is a situation that should be avoided.

In conclusion, the security of CI/CD systems and processes is a complex issue that requires a deep understanding of the tools and processes involved. It is not enough to secure the software product; the entire development and deployment pipeline must be secured.

Poisoned Pipeline Execution



CHAPTER 5
Prevention and Protection

As technology advances, so do the techniques employed by malicious actors to exploit vulnerabilities within the supply chain. To safeguard these systems, companies must shift their attention towards implementing effective strategies that can proactively prevent such threats and ensure the protection of critical software development and delivery infrastructure and automated processes.

However , it's crucial to understand that **security is not a one-size-fits-all solution**. Each organization has unique needs, risks, and infrastructure, and the security measures implemented should reflect these unique factors. Therefore, before adopting any security recommendations, it's essential to ask yourself some critical questions:

- Do I control the risk in the process of building the software from sources and deploying it in production?
- Do I even know what the pipelines are in action today, what vulnerabilities they present, and what pieces compose them?
- Do I have the means to determine the configurations related to the security against the mentioned attacks?

These questions will help you understand your current security posture and identify areas that need improvement. They will guide you in understanding the unique vulnerabilities of your CI/CD systems and the steps you need to take to secure them.

"CI/CD security requires a proactive, ongoing approach. It's about balancing convenience and security, understanding your pipelines, and continuously adapting to the evolving threat landscape. The ultimate goal is to protect your software supply chain and ensure software security & integrity."

5. Prevention and Protection

Specific Strategies for Securing CI/CD Systems

1.- Configuration is critical to CI/CD security. There needs to be a balance between convenience and security. Misconfigurations can expose the software to security breaches and various forms of attacks. There are many insecure configurations, such as unprotected code delivery branches, inadequate code reviews, weak access control practices (e.g., lacking multi-factor authentication), publicly accessible storage buckets in cloud infrastructure, unencrypted critical data at rest, and weak password policies combined with non-rotated encryption keys. This is a delicate balance to strike, as convenience often leads to increased productivity, but it should not come at the cost of security.

2.- Create a management plan that integrates into the DevOps process: It is important to create a plan to resolve all detected issues, such as hardcoded secrets, misconfigurations, malicious code, etc. Therefore, it is essential to integrate their management with corporate ticketing tools to facilitate their resolution straightforwardly.

3.- Foster knowledge sharing and consider the human factor. Developers and DevOps engineers play a crucial role in CI/CD security. They are the ones who build and maintain the pipelines, and their actions can either secure or compromise these systems. Therefore, they need to be trained on securing software and recognizing malicious code. Overconfidence in pipeline security can lead to overlooked vulnerabilities, so continuous training and awareness are key.

4.- Analyze Unusual Activities. Examining activities across multiple sources in the Software Development Life Cycle (SDLC), such as code repositories, build systems and deployment pipelines, allows for the detection of abnormal behaviour that could indicate a security breach or unauthorized access. Xygeni's role is vital in detecting discrepancies caused by diverse factors like human error, misconfigurations, or malicious actions. Furthermore, it provides auditing and monitoring features to identify any abnormal activity, such as unauthorized access or suspicious modifications to code.

5.- When choosing a CI/CD security framework, consider its maturity and the support it offers. The industry is still reacting to software supply chain security, and standards, frameworks, and guidelines are still under construction.. For example, the NIST Cybersecurity Framework, initially released by the National Institute of Standards and Technology (NIST) in 2014, comprises a collection of standards, guidelines, and best practices to effectively manage cybersecurity risks. In 2018, it underwent updates to stay relevant in the rapidly evolving landscape. However, there are newer frameworks, like Google Supply-chain Levels for Software Artifacts (SLSA), that are specifically designed to uphold the integrity of software artifacts across the entire software supply chain. These frameworks, among others, play a crucial role in establishing robust security measures and ensuring the protection of software systems. Choosing a mature and well-supported framework can provide a solid foundation for your CI/CD security. However, it is also important to create comprehensive policies and guidelines that outline the specific security requirements for our company.

6.- Harden your build environment and tools. This includes implementing key security controls and regularly updating and patching your tools to protect against known vulnerabilities. Regular updates and patches are crucial as new vulnerabilities are discovered frequently, and outdated tools can provide an easy entry point for attackers.

Remember that securing your CI/CD systems is not a one-time effort. It requires continuous monitoring, evaluation, and improvement. The threat landscape is constantly evolving, and your security measures need to evolve with it. By taking a proactive approach to CI/CD security, you can protect your software supply chain and ensure the integrity of your software products.

CHAPTER 6

OWASP Top 10 CI/CD Security Risks

The Open Web Application Security Project (OWASP) has compiled a list of the top 10 security risks for CI/CD systems, which provides a comprehensive overview of the potential vulnerabilities and how to mitigate them. **Here are a summary of them and their potential impact:**

1. **Insufficient Flow Control Mechanisms**: This risk refers to the lack of proper controls to manage the flow of data and tasks in the CI/CD pipeline. Without these controls, unauthorized changes could be introduced into the pipeline, leading to potential security vulnerabilities.
2. **Inadequate Identity and Access Management**: This risk involves not properly managing who has access to the CI/CD pipeline and what they can do. Without proper identity and access management, unauthorized individuals could gain access to the pipeline and introduce malicious changes.
3. **Dependency Chain Abuse**: This risk refers to the potential for attackers to exploit vulnerabilities in the dependencies used by your software. If these dependencies are not properly managed and secured, they could provide an avenue for attack.
4. **Poisoned Pipeline Execution**: This risk involves attackers potentially introducing malicious code into the CI/CD pipeline. This could lead to the execution of malicious code in the production environment.
5. **Insufficient PBAC (Pipeline-Based Access Controls)**: This risk refers to the lack of proper access controls based on the pipeline. Without these controls, unauthorized individuals could access sensitive parts of the pipeline.
6. **Insufficient Credential Hygiene**: This risk involves not properly managing and securing credentials used in the CI/CD pipeline. If these credentials are compromised, it could provide an attacker access to the pipeline.
7. **Insecure System Configuration**: This risk refers to potential security vulnerabilities due to improperly configured systems in the CI/CD pipeline. Attackers could exploit these vulnerabilities.
8. **Ungoverned Usage of 3rd Party Services**: This risk involves using third-party services without proper oversight and control. These services could introduce security vulnerabilities if they are improperly managed and secured.
9. **Improper Artifact Integrity Validation**: This risk refers to the lack of proper validation of artifacts produced by the CI/CD pipeline. Without proper validation, malicious or compromised artifacts could be introduced into the production environment.
10. **Insufficient Logging and Visibility**: This risk involves insufficient logging and visibility into the activities in the CI/CD pipeline. It could be difficult to detect and respond to security incidents without proper logging and visibility.

6. OWASP Top CI/CD Security Risks

Other compliance & cybersecurity frameworks and standards

Other guidelines and standards enumerate the main risks affecting CI/CD, such as those from NIST, CIS, Google SLSA, or recently published by the NSA.

All these documents agree when mentioning the primary issues existing in the software supply chain, such as the following:

1. Lack of visibility and transparency regarding the SBOM (software bill of materials),
2. Uncontrolled use of third-party components and software dependencies
3. Unauthorized access and lack of permission management
4. Code tampering and poisoned pipeline execution
5. Malicious code injection
6. Vulnerabilities in IaC (infrastructure as code) and misconfigurations
7. Exposure of secrets in the code (hardcoded passwords).

CLOSING

Securing the Future of Software Supply Chain

The security of the software supply chain is not just about protecting individual components. It's about securing the **entire processand infrastructure, from the initial code development to its deployment in production environments.** This requires a comprehensive approach that addresses every link in the chain. It's not enough to secure the code itself; we must also secure team activity, tools and processes used to build, test, and deploy that code.

But understanding these risks is just the first step. To truly secure our software supply chains, we need to take proactive measures to mitigate these risks.

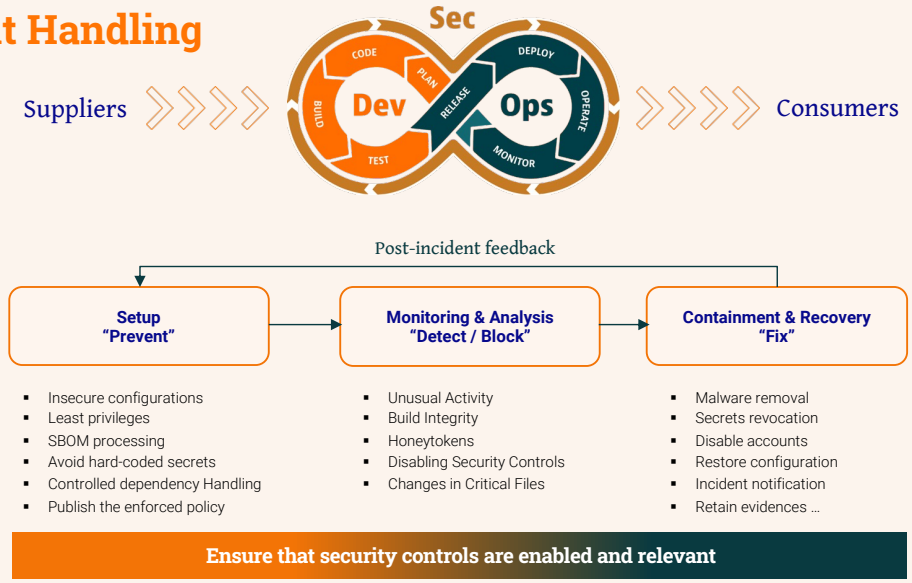
This means implementing robust security controls, regularly auditing our systems and processes and staying informed about the latest threats and vulnerabilities. It also means fostering a **culture of security within our organizations**, where everyone from developers to executives understands the importance of software supply chain security and their role in maintaining it.

As we move forward, it's crucial to remember that the threat landscape constantly evolves. The tactics and techniques used by attackers today may not be the same ones they use tomorrow. Therefore, our approach to software supply chain security must also evolve. We must be willing to adapt our strategies and tools in response to new threats and vulnerabilities. We must also be prepared to invest in the resources necessary to protect our software supply chains, recognizing that the cost of a successful attack can far outweigh the cost of prevention.

Moreover, relying on manual controls alone is not sufficient to ensure the robust security of the software supply chain. The dynamic and intricate nature of modern software development calls for specific and advanced tools, such as Xygeni, that can effectively mitigate risks and reduce dependence on manual interventions.

These specialized tools can automate security processes, perform continuous monitoring, detect threats, and facilitate rapid response to emerging ones. By leveraging such tools, organizations can enhance their ability to maintain a secure software supply chain while minimizing human errors and vulnerabilities introduced through manual procedures. Embracing these advanced technologies not only streamlines security practices but also ensures a higher level of consistency and reliability throughout the entire software development and deployment lifecycle.

SCS Incident Handling



In conclusion, securing the software supply chain is a complex but essential task. It requires a deep understanding of the threats we face, a comprehensive approach to security, and a commitment to continuous learning and adaptation. By applying the knowledge and strategies we've discussed in this document, we can make significant strides towards securing our software supply chains and protecting our organizations from potential attacks.

End-to-End Software Development & Delivery Security

Protects the integrity and security of your software assets, pipelines and infrastructure of the entire software supply chain.

Contact

Get in touch today!

www.xygeni.io
https://www.linkedin.com/company/xygeni
https://twitter.com/xygeni