
Technology
Innovation
Institute
SUPPLY
CHAIN
SECURITY

ABSTRACT

Despite billions of dollars of investment in software supply-chain security tools, enterprises and governments continue to be hacked. At the Security Systems Research Center of the Technology Innovation Institute (TII SSRC, Abu Dhabi, UAE, tii.ae/secure-systems), researchers are working on new solutions to seal up the security gaps that make these breaches possible. This white paper delves into SSRCs' work to bolster supply-chain security (SCS) security across efforts like Supply-Chain Levels for Software Artifacts (SLSA, slsa.dev), the open-source Nix package manager (nixos.org), and Identity and Access Management (IAM) infrastructures, all to automate a secure build pipeline. Key enhancements include new Software Bill of Materials (SBOM) generation tools, vulnerability-management automation, a flexible public key infrastructure, an automated Zero Trust development-and-build pipeline, and creation of scalable and ephemeral build environments.

INTRO

Recent years have seen a marked increase in attacks on the software supply-chain—with notable impacts on such security-conscious organizations as SolarWinds, Microsoft, OKTA, Kaseya, and British Airways. These attacks typically involve malicious or vulnerable code inserted into legitimate products at almost any stage of the product's lifecycle.

Perhaps most alarmingly, at the end of March 2024, a Microsoft developer reported that someone (just who is still unknown) had planted a back door in the XZ compression tool slated for inclusion in coming Linux updates. [1] Had they succeeded, this vulnerability could have given the perpetrators remote access to Linux systems underpinning much of the modern economy. This vulnerability was not discovered by a security researcher, but rather by a performance engineer who wondered why the compression utility seemed to be running so slowly.

These threats have prompted significant regulatory and industry-led responses, notably *U.S. Executive Order 14028: Improving the Nation's Cybersecurity*. [2] The order mandates Software Bills of Materials (SBOMs) for all software delivered to the American government. Concurrently, industry-led initiatives such as the Supply-Chain Levels for Software Artifacts (SLSA) framework seek to standardize verification methods and terminology in software delivery.

An added challenge is that sometimes new vulnerabilities (such as the Log4shell vulnerability in the Log4J logging tool [3]) are discovered in open-source libraries long after development teams have adopted them. Once these weaknesses are found, development teams face a race against time to patch them, which takes them away from regular development activities. Sonatype's 2023 State of Software Supply Chain report found that 245,000 malicious packages had been discovered during the previous year—twice the number discovered in all previous years combined—and that one-eighth of all downloaded open-source code contains known risks[4].

In the interconnected modern business ecosystems, amplified by the proliferating Internet of Things (IoT), business opportunities and heightened risks of cyberattacks seem to go hand-in-hand. The industry advisory firm Cybersecurity Ventures predicts that the cost of cybercrime will rise from $3 trillion in 2015 to $10.5 trillion by 2025. Enterprises spent about $150 billion on cybersecurity tools in 2021, up about 12.8% from the previous year. However, McKinsey researchers have argued that the scope and cost of security threats suggest a $2 trillion market opportunity.[6].

[1] Goddin, Dan. "What we know about the XZ Utils backdoor that almost infected the world," Ars Technica, Apr. 1, 2024. https://arstechnica.com/security/2024/04/what-we-know-about-the-xz-utils-backdoor-that-almost-infected-the-world/
[2] "Executive Order 14028: Improving the Nation's Cybersecurity," issued by U.S. President Joseph R. Biden, Jr. on May 21, 2021. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/
[3] "Log4j vulnerability - what everyone needs to know," National Cyber Security Center, Dec. 15, 2024. https://www.ncsc.gov.uk/information/log4j-vulnerability-what-everyone-needs-to-know
[4] "Understanding Open-Source Adoption: Insights from the 9th State of the Software Supply Chain Report." Accessed: Apr. 05, 2024. [Online]. Available: https://www.sonatype.com/state-of-the-software-supply-chain/introduction
[5] "2023 Official Cybercrime Report," eSentire. Accessed: Apr. 05, 2024. [Online]. Available: https://www.esentire.com/resources/library/2023-official-cybercrime-report
[6] "New survey reveals $2 trillion market opportunity for cybersecurity technology and service providers | McKinsey." Accessed: Apr. 05, 2024. [Online]. Available: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/new-survey-reveals-2-trillion-dollar-market-opportunity-for-cybersecurity-technology-and-service-providers

When malefactors breach front-line security tools and infrastructure, the situation becomes critical, and the computing establishment quickly focuses on the issues.. But securing the back-end software supply chain, the structure that delivers the applications, demands just as much attention. Keeping software clean requires considering in detail the complex ecosystem of development tools, code libraries, compilers, and hosting infrastructure. The security research community has thus been exploring many ways of hardening and enforcing a chain of trust around the tools for developing, building, and deploying code.

It's important to note that modern software supply chains span wide ranges of processes, tools, and artifacts—including code libraries, code management, development practices, and provisioning infrastructures. A weakness in any part of this chain can allow hackers to insert malicious code into applications and infrastructure. Protecting software components throughout their lifecycles, from design to delivery, is critical for maintaining business reputation, safeguarding intellectual property, and ensuring customer safety.

Clearly, development organizations must cultivate appropriate combinations of tools, infrastructure, and workflows to protect their software and its infrastructure. The TII's SSRC has explored best practices for building a Secure Software Supply Chain as part of its work on the Ghaf framework for building a highly secure virtual OS and ecosystem that covers phones, workstations, drones, and embedded systems using a Zero Trust architecture (ZTA). The Ghaf platform is an essential tool in research and development for implementing Zero Trust frameworks, necessitating a secure development process.

[1] Goddin, Dan. "What we know about the XZ Utils backdoor that almost infected the world," Ars Technica, Apr. 1, 2024. https://arstechnica.com/security/2024/04/what-we-know-about-the-xz-utils-backdoor-that-almost-infected-the-world/
[2] "Executive Order 14028: Improving the Nation's Cybersecurity," issued by U.S. President Joseph R. Biden, Jr. on May 21, 2021. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/
[3] "Log4j vulnerability - what everyone needs to know," National Cyber Security Center, Dec. 15, 2024. https://www.ncsc.gov.uk/information/log4j-vulnerability-what-everyone-needs-to-know
[4] "Understanding Open-Source Adoption: Insights from the 9th State of the Software Supply Chain Report." Accessed: Apr. 05, 2024. [Online]. Available: https://www.sonatype.com/state-of-the-software-supply-chain/introduction
[5] "2023 Official Cybercrime Report," eSentire. Accessed: Apr. 05, 2024. [Online]. Available: https://www.esentire.com/resources/library/2023-official-cybercrime-report
[6] "New survey reveals $2 trillion market opportunity for cybersecurity technology and service providers | McKinsey." Accessed: Apr. 05, 2024. [Online]. Available: https://www.mckinsey.com/capabilities/risk-and-resilience/our-insights/cybersecurity/new-survey-reveals-2-trillion-dollar-market-opportunity-for-cybersecurity-technology-and-service-providers

HOW SOFTWARE SUPPLY CHAINS GET COMPROMISED

(SIDEBAR)

Modern software supply chains are built across complex ecosystems of tools, services, and processes. Each element may have potential vulnerabilities, and these may sometimes be compromised. Here are some examples:

Infrastructure as Code:
The complexity of Infrastructure as Code (IaC) tools for automating provisioning, combined with human error, can sometimes overlook misconfigurations. These can replicate across running infrastructure to turn simple coding errors into significant vulnerabilities.

Leaked secrets:
Secrets such as passwords and tokens are sometimes embedded in code, either accidentally or for the coder's convenience. Hackers who discover these keys gain direct access to critical systems to steal data or plant malware.

CI/CD pipeline:
Continuous Integration/Continuous Deployment pipelines automatically provision and deploy of new code. When CI/CD systems are compromised, hackers can introduce malicious code into production systems.

Over-provisioning of access:
Poor security policies may result in overprovisioning access (giving a set of credentials wider rights than strictly necessary) to development environments. Hackers who access these credentials or malicious insiders introduce malicious code, steal data, or install backdoors for further exploitation.

Open source and third-party components:
Malicious actors sometimes plant malware or backdoors in software components incorporated into new software or the tools and systems used to build software. In other cases, new vulnerabilities are discovered in existing libraries; these must be patched and deployed into releases.

Compromised development tools and accounts:
Hackers who compromise a developer account can introduce malware and vulnerabilities into software build systems, compilers, and code management services like GitHub.

Update infrastructure:
Hackers sometimes find ways to access the infrastructure responsible for software updates to inject malware.

SUPPLY CHAIN SECURITY LIFECYCLE

Security vendors and the open-source community have developed various security tools to help protect multiple aspects of the software supply chain. For example, software composition analysis helps inventory code in production and under development, to check it against vulnerability reporting databases. Application security tools (AppSec) apply various security-testing approaches to identify security issues. Secure code management tools help protect code repositories from malicious changes.

[1]"Understanding Open Source Adoption: Insights from the 9th State of the Software Supply Chain Report." Accessed: Apr. 05, 2024. [Online]. Available: https://www.sonatype.com/state-of-the-software-supply-chain/introduction

BRIEF WALKTHROUGH OF SCS LIFECYCLE (SIDEBAR)

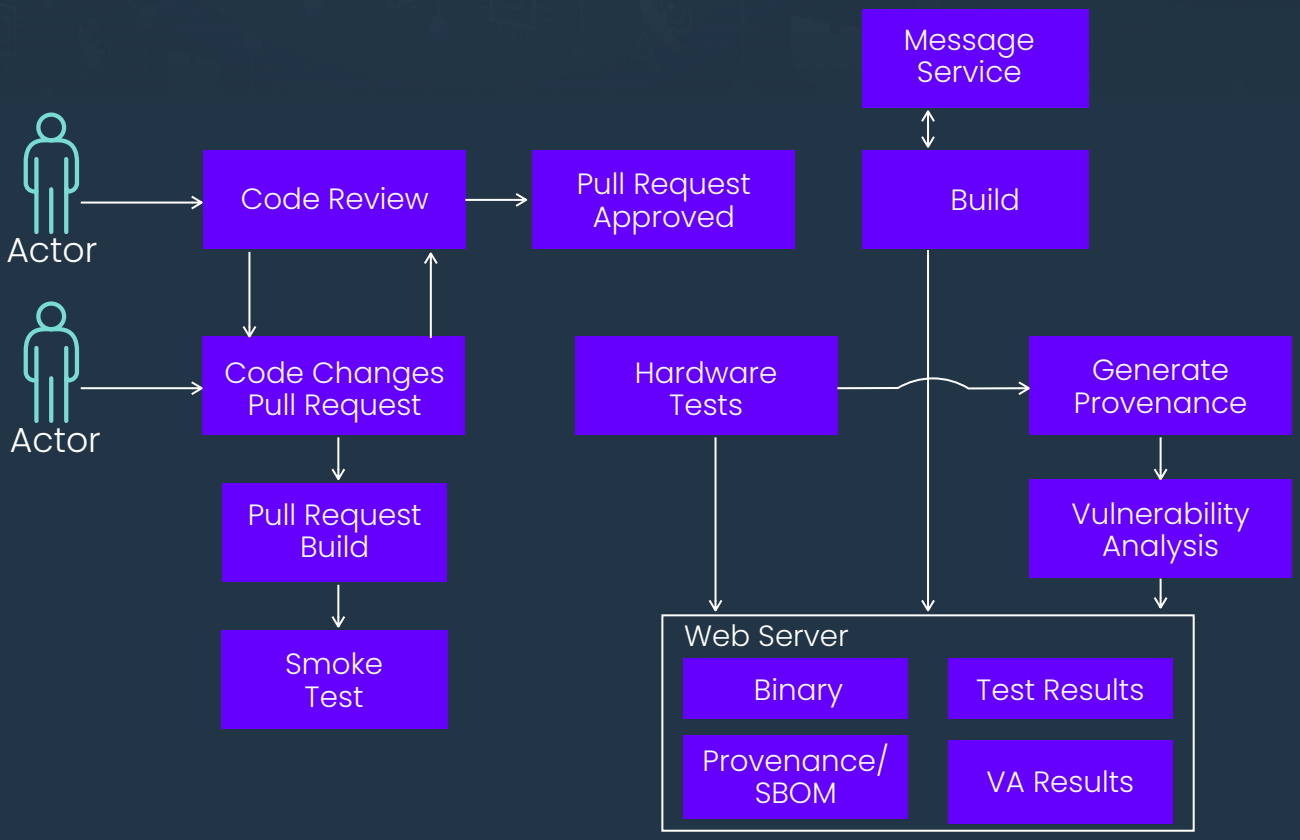


Figure 1: SCS lifecycle required to amalgamate binaries, SBOM, test results, and vulnerability data.

Essential efforts that stand out in hardening this chain of trust are Nix for packages, SLSA for practices, SBOMs for components, and public key infrastructure for signing software artifacts:

Software Composition Analysis (SCA): Plays a crucial role in identifying and monitoring software components and their dependencies. Its implementation provides real-time visibility into the software supply chain, enhancing security measures.

TSupply-Chain Levels for Software Artifacts (SLSA): The OpenSSF specifies a framework for safe development practices for hardening development processes.

Nix: Package manager helps build tamper-evident seals in containers that help transform raw ingredients into application binaries.

Software Bill of Materials (SBOM): Helps track the components that go into the final product, making it easier to identify and remediate vulnerable components across infrastructure and ensuring that they do not find a way into future software builds.

Public Key Infrastructure (PKI): Helps automate Identity and Access Management (IAM) to ensure that only authorized users and applications can add, change, delete or execute software artifacts.

These efforts are all making meaningful progress and are still early in their maturity and adoption. So, the Technology Innovation Institute's Secure Software Research Center has led research to help understand some of the current challenges, add new capabilities where required, and improve workflow flexibility, hardening overall software supply chain security. This work parallels SSRC's broader Ghaf program to develop Zero Trust secure virtual operating systems for workstations, phones, drones, and other embedded systems.

KEY TII ADVANCES (SIDEBAR)

SLSA:
The Supply-Chain Levels for Software Artifacts (SLSA) framework offers a set of standards for ensuring software artifact integrity. The TII created a framework for combining the SLSA provenance record and the SBOM, furnishing them alongside each binary. This helps software consumers authenticate and comprehend the composition of software artifacts. The provenance record is vital in tracing and addressing potential security issues, providing an extensive background of the artifact's journey. Meticulously documenting the genesis and evolution of software artifacts enhances security and transparency—fostering trust and confidence in the integrity of a software solution.

Nix tooling:
The TII created **sbomnix**, a utility for generating SBOM from Nix paths that provides a comprehensive list of all components used in the build process, thereby enhancing the traceability and transparency of the software supply chain. This includes tools for querying dependency graphs **(nixgraph)**, summarizing package attributes **(nixmeta)**, improving vulnerability analysis **(vulnxscan)**, tracking package versions and associating vulnerabilities **(repology_cli** and **repology_cve)**, and identifying outdated packages **(nix_outdated)**.

CVE Management:
Developed tooling to help developers and users efficiently track potential issues in upstream packages using CVEs (Common Vulnerabilities and Exposures) to proactively address security vulnerabilities and ensure the integrity of their software solutions.

PKI Workflow:
The TII adopted a three-tier public key infrastructure (PKI) workflow rooted in a hardware key that supports multiple clouds and scalability. The method automates the chain of trust for developers, applications, and the provisioning of cloud infrastructure to support specific requirements.

CI/CD:
SSRC developed an advanced continuous-integration-and-deployment software signing process. This mechanism guarantees software integrity during transitions between stages of the build process while ensuring that cryptographic keys never leave the secure environment.

Ephemeral and hermetic build environments:
Pioneering research spins up temporary and isolated build environments, which guard against advanced persistent threats. Isolation prevents cross-contamination between automated builds inside the build system. Hermetic builds reduce the risk of introducing vulnerabilities through dependencies.

SLSA

The Supply-Chain Levels for Software Artifacts (SLSA) method is a comprehensive framework to enhance software artifacts' integrity across supply chains. Developed collaboratively by Google and industry leaders, SLSA version 1.0 (released in April 2023), categorizes security measures in four distinct levels:

LEVEL 0

Incorporates no SLSA requirements.

LEVEL 1

Encompasses basic SLSA requirements.

LEVEL 2

Incorporates recommended SLSA guidelines.

LEVEL 3

Meets comprehensive SLSA requirements.

Each level builds upon its predecessor, culminating in Level 3, which adopts the most stringent security measures. These levels define specific protocols that uphold software artifact integrity, ranging from digital signatures and vulnerability checks to advanced security practices.

The SLSA framework v1.0, released under the Open SSF and part of the Linux Foundation, offers a standardized language for discussing and enhancing software supply chain security per the SLSA standard. [7] It is characterized by flexibility and adaptability, aligned with modern software development principles and security assurance.

Some specific Supply-Chain Security (SCS) suite provisions include alignment with SLSA standards, code reviews, automated vulnerability scanning, build environment monitoring, and automated SBOM generation. The shift towards reproducible builds in the forthcoming platform iteration is a significant stride towards this goal.

[7] "SLSA specification," SLSA. Accessed: Apr. 05, 2024. [Online]. Available: https://slsa.dev/spec/v1.0/

ALIGNING NIX AND SLSA

TII SSRC created the Ghaf Framework and the Ghaf Platform to support a secure, scalable, Zero Trust environment for creating, deploying, and maintaining robust and reliable edge applications.[8]In the context of software supply chain security, the Ghaf platform leverages a robust build system to embed SLSA provenance at the build stage, ensuring high reproducibility and reliability. This system captures essential details such as source code, build environment, and parameters, forming a comprehensive SLSA provenance record. This record provides an exhaustive history of the software artifact from inception to distribution.

The system concurrently generates a Software Bill of Materials (SBOM), listing all components, versions, and dependencies. As part of our SCS suite, sbomnix helps developers comprehensively understand the runtime and build-time dependencies of every artifact within an image. This SBOM, provided by sbomnix, can now produce both CycloneDX[9] and SPDX[10] formats without any need for conversion, enabling broader compatibility and enhanced transparency across the software supply chain.

The amalgamation of the SLSA provenance record and SBOM, furnished alongside each binary, lets consumers comprehend and authenticate the composition of software artifacts. The provenance record is vital in tracing and addressing potential security issues, providing an extensive background of the artifact's journey.

The SLSA framework and its integration into Ghaf's processes represent a significant stride in securing software supply chains. By meticulously documenting each bit of software's genesis and evolution, Ghaf enhances security and transparency, ultimately fostering consumer trust and confidence in the integrity of their software. Using CVEs (Common Vulnerabilities and Exposures) to proactively address security vulnerabilities, the new tooling helps consumers efficiently track potential issues arising from upstream packages. This comprehensive approach fosters consumer trust and strengthens confidence in the reliability of Ghaf's software offerings.

A pivotal premise of the Ghaf approach to software supply chain security is transparency and trust in the components used to construct the software. And central to this is the Software Bill of Materials (SBOMs)—like the familiar industrial bill of materials, a detailed list of every component, its origin, specifications, nature, and function.

[9] "OWASP CycloneDX Software Bill of Materials (SBOM) Standard." Accessed: Apr. 05, 2024. [Online]. Available: https://cyclonedx.org/
[10] "SPDX – Linux Foundation Projects Site." Accessed: Apr. 05, 2024. [Online]. Available: https://spdx.dev/

Using SBOMs bolsters software security in many ways. Primarily (and as already noted), SBOMs facilitate comprehensive assessments of software components, their interdependencies, and their known vulnerabilities and interactions. In the face of complex, fast-evolving cyber threats, good SBOMs allow indispensable rapid reactions.

Furthermore, SBOMs significantly help to manage licenses, increasing compliance while reducing bookkeeping overheads. By providing explicit details on component licenses, SBOMs preempt the legal and financial complications of violating licenses. Additionally, SBOMs are instrumental in managing software-component lifecycles, ensuring timely replacement of unsupported or obsolete elements to avert the hazards of outdated code.

Understanding the critical importance of SBOMs, the Ghaf team developed *sbomnix* as an innovative in-house SBOM generation tool, which they then shared with the community. They conceived *sbomnix* as a much-needed, community-maintained SBOM generator capable of analyzing Nix artifacts to produce SBOMs to meet rigorous quality standards.

Sbomnix is seamlessly integrated into the TII SSRC's software supply chain and is operational during the software-artifact-testing phase (coordinated by the Jenkins Project, jenkins.io). After the SBOMs are generated, they are converted into CycloneDX and SPDX formats to facilitate accessibility and usability by diverse stakeholders in the software supply chain.

While SBOMs, as defined by standards such as CycloneDX and SPDX, primarily document software components, the TII approach uniquely integrates these SBOMs with additional provenance data and direct links to binaries. This extended integration goes beyond current standards to enhance transparency and trust across the supply chain. This initiative reflects more than the TII SSRC commitment to software supply chain security: it is a testament to our dedication to providing stakeholders with high-quality, transparent, and accountable software components by reliably creating high-quality SBOMs.

[9] "OWASP CycloneDX Software Bill of Materials (SBOM) Standard." Accessed: Apr. 05, 2024. [Online]. Available: https://cyclonedx.org/
[10] "SPDX – Linux Foundation Projects Site." Accessed: Apr. 05, 2024. [Online]. Available: https://spdx.dev/

NIX

Nix (available at nixos.org and github.com/NixOS/nix) offers tools for provisioning a reproducible, declarative, and reliable build pipeline that hardens the process of turning software code into binary packages. Nix builds packages in isolation from each other to ensure the process is reproducible and has no undeclared dependencies. Nix also provides a framework for sharing software development and build environments in a way that reduces configuration and library installation requirements for developers.

The *Nix* infrastructure ensures that updating one package doesn't break others. It helps create tamper-proof packages for managing, and collaborating on, the various tools for developing, building, and running apps. *Nix* also makes it easier to ensure that a particular app incorporates an exact version of a specific library. It also creates an audit trail for identifying where that library is used if a problem is discovered later.

Analyzing a system requires a thorough listing of all the applications and all of their dependencies. This can be done during the build phase or the runtime phase. An end user does this analysis at runtime to see all the versions of applications, libraries, and configurations on the system. An IT department may also want to check what will be sent to the users in new updates or review what users are currently running. These checks can be done offline, on a user's system, or in some build environment.

TII SSRC is also working on tools that work with the *Nix* package manager and flakes, an experimental feature that enables precise version pinning of packages and source codes. This approach advances the pursuit of fully reproducible builds. The *Nix* ecosystem further supports automated vulnerability scanning (vulnxscan) and continuous integration, which further reinforce the security infrastructure.

Shortcomings of previous automated SBOM generators have long troubled the Nix community. The SSRC evaluated many packages that purported to support *Nix*. We still needed help finding a SBOM generators that could scan *Nix* artifacts and reliably produce an SBOM of acceptable quality. Although a few tools did exist, they were either not maintained or were part of another individual's project.

A workable SBOM-generator utility should meet the following performance standards:

- It must produce valid SBOM in standard format (CycloneDX or SPDX), which should be importable to other tools such as Dependency-Track.
- It must produce the most complete possible SBOMs (for instance, the tool should include the license information)
- It must clearly show which dependencies are included. Given a target artifact, does the tool include only runtime dependencies, or does it also include build-time dependencies?
- It should include the dependencies of a static build binary, or clearly indicate that this information is omitted.
- It should explain how patches are handled. A vulnerability might not impact a vendor-specific product version because it is forked, or the patch is backported.

TTo address these shortcomings, the SSRC developed a suite of tools, each adding a bit of functionality to the overall goal of automating the SBOM. When pieced together, they help build the foundation for a secure supply chain security process.

The **ghafscan** repository (github.com/tiiuae/ghafscan) automates vulnerability scans for the Ghaf Framework. The repository also includes Ghaf vulnerability reports that can be incorporated into an automated vulnerability scan workflow. It builds on the set of tools that SSRC has created as part of the **sbomnix** repository. These include the following (expanding on the earlier descriptions):

- **Sbomnix:** generates Software Bill of Materials (SBOMs) from Nix derivations or output paths.
- **Nixgraph** facilitates querying and visualizing dependency graphs for Nix derivations or output paths. This tool aids in understanding the intricate dependencies among software components, which is essential for identifying potential security vulnerabilities and managing risks preemptively.
- **Nixmeta:** summarizes nixpkgs meta-attributes from the given nixpkgs version, further streamlining the software development process and ensuring consistency across packages

- **Vulnxscan:** represents a significant advancement in vulnerability analysis, demonstrating the usage of SBOMs in running vulnerability scans. Drawing data from extensive databases— such as the National Vulnerability Database (NVD) and the Distributed Vulnerability Database for Open Source (OSV)—this tool scrutinizes software components against known vulnerabilities, alerting developers to any new vulnerabilities introduced in subsequent builds. This proactive approach is instrumental in identifying and addressing potential security threats before they can be exploited.
- **repology_cli and repology_cve:** command-line clients to repology.org to enable efficient tracking of package versions and vulnerabilities across repositories.
- **nix_outdated:** finds outdated Nix dependencies for a given output path, listing the outdated packages in priority order based on how many others depend on the given outdated package.

Synthesizing these in-house tools with Nix-provided tools has markedly elevated the security and integrity of software artifacts throughout the supply chain at Ghaf. The vulnerability analysis tool, in particular, emerges as a pivotal innovation in preemptive security management. The contribution of these tools to the Nix community reflects an intention to benefit the broader industry, underscoring the importance of collaborative development and shared advancements in software security.

Ghaf's ultimate goal extends beyond providing trusted images or binary caches. It encompasses establishing a reproducible build environment that enables independent verification, building, and sharing of build results. This initiative marks a crucial step toward fortifying the security and reliability of software supply chains, ensuring robust defenses against emerging threats. This effort must, however, be coupled with a robust SCS to achieve the goals.

PKI IN ENHANCING SOFTWARE SUPPLY CHAIN SECURITY

Public Key Infrastructure (PKI) provides a foundation for ensuring the security and integrity of the software supply chain. It helps establish and manage secure digital identities for authentication, encryption, and digital signatures across diverse digital platforms. PKI is vital for automating the security and governance processes required to set up Zero Trust boundaries across the myriad levels of the software development and deployment lifecycle.

COMPONENTS OF PKI

1 **Certificate Authority (CA):**
The CA functions as the pivotal trust entity within PKI, responsible for issuing and managing digital certificates. It validates entities' identities, ensuring the legitimacy and integrity of their public keys.

2 **Digital Certificates:**
These certificates function as digital credentials, linking public keys with entities' respective identities. They comprise essential data, including the entity's name, its public key, the issuing CA's signature, and the certificate's validity period.

3 **Certificate Revocation Systems:**
PKI includes mechanisms for invalidating digital certificates, primarily through Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP).

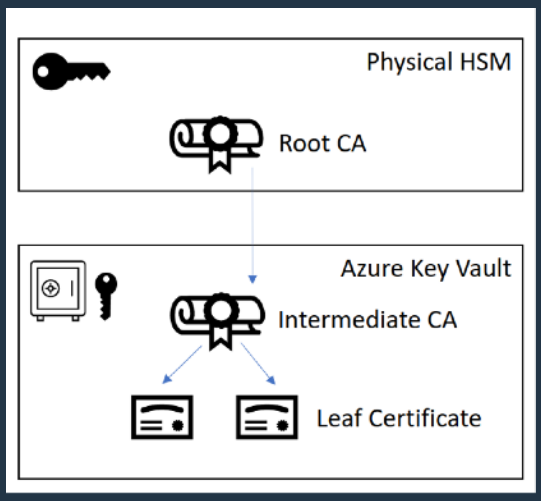
4 **Certificate Chains:**
This hierarchical structure of certificates enables the establishment of trust. Each certificate in the chain is authenticated by the preceding certificate, culminating in a root CA.

PKI IMPLEMENTATION IN GHAF'S SOFTWARE SUPPLY CHAIN

Root CA Secured in Physical HSM:
The Root CA, the cornerstone of trust in the PKI hierarchy, securely stores its private key in a physical Hardware Security Module (HSM). This ensures maximal security for the key, which is pivotal to the entire PKI framework.

Three-Tier CA Structure:
The PKI architecture consists of three tiers: the Root CA, Intermediate CA, and Leaf (or Signing) Certificate. This hierarchy facilitates efficient distribution and management of trust.

- The Root CA is the apex authority, providing the foundation of trust.
- The Intermediate CA is a mediator, issuing certificates assigned to various functions, such as signing build artifacts or websites.
- The signing or Leaf Certificate is directly responsible for signing the artifacts.



BENEFITS OF GHAF'S PKI STRATEGY

1 **Enhanced Security Measures:**
Utilizing a physical HSM for the Root CA's private key significantly mitigates risks of unauthorized access, thereby fortifying the security of the PKI system.

2 **Flexibility in Vendor Choice:**
Storing the Root CA in a physical HSM instead of a cloud-based private certificate authority (PCA) empowers the organization with vendor-independent control. This autonomy in managing the Root CA key facilitates potential transitions between cloud service providers.

3 **Scalability and Management through trusted key vault:**
Leveraging Azure Key Vault for managing Intermediate and ephemeral signing certificates combines the benefits of cloud scalability with robust security, enabling efficient key management.

4 **Establishment of Trust and Reliability:**
The three-tier CA structure, underpinned by the physical HSM and Azure Key Vault, engenders a reliable trust chain from the root to the end-users, augmenting the security fabric of the software supply chain.

Ghaf's integration of a physical HSM with a trusted key vault within its three-tier PKI architecture exemplifies a strategic amalgamation of security, flexibility, and manageability. This approach both elevates software supply chain security and ensures adaptability and trustworthiness, aligning with contemporary digital security exigencies.

IMPROVING CI/CD WORKFLOW

Ghaf's CI/CD (Continuous Integration/ Continuous Deployment) system has been meticulously architected to ensure the security and integrity of software artifacts. Central to this architecture is an advanced software signing process underpinned by the robust three-tier Certificate Authority (CA) structure, with its apex, the Root Certificate securely stored on a physical Hardware Security Module (HSM). At the same time, the Intermediate and ephemeral signing certificates are securely managed within Azure Key Vault.

The process begins when the build-controller triggers a remote builder, compiling the binary and generating a corresponding provenance file. The signer service then signs this dual output using a leaf certificate unique to the builder's identity, ensuring traceability and integrity. The Amazon Web Services Azure SDK facilitates direct signature verification with Key Vault, allowing secure cryptographic operations without exposing sensitive material.

As the binary and provenance record move through the pipeline, their signatures are verified at each stage, reinforcing the security at every step. Finally, the signed artifacts and their signatures are published to a web server. This enables end-users to independently verify the integrity of their downloaded files, using the published certificates.

SECURITY BENEFITS

Enhanced Security:
Ghaf CI/CD significantly mitigates the risks of compromised keys and unauthorized access by storing the Root CA on a physical HSM and utilizing a trusted key vault for other certificates. Moreover, this setup enables us to set up multiple build environments efficiently. We can issue an intermediate certificate for each instance, requiring the HSM to enroll in a new build environment only once; the administrator also needs only a single operation to revoke an intermediate certificate, simultaneously enhancing security and scalability.

Traceability and Accountability:
A unique certificate for each builder provide a clear audit trail from binary creation to deployment.

Secure Signature Verification:
By verifying signatures with Key Vault, the system ensures that cryptographic materials never leave the secure environment.

Ghaf's CI/CD software signing process represents a strategic balance between robust security measures and operational efficiency. Integrating physical and cloud-based security practices ensures that software artifacts are authenticated, verifiable, and secured throughout their lifecycles.

SCALABLE, EPHEMERAL, AND HERMETIC BUILD ENVIRONMENTS

Moreover, ephemeral build environments inherently exist in a state of constant vigilance. With a fresh configuration for each build, these environments mandate continuous security assessments and updates, integral to a culture of perpetual security enhancement. This approach does have its challenges, particularly the robust orchestration and monitoring systems needed to manage the lifecycles of these transient environments. However, security dividends—mitigating long-term vulnerabilities and ensuring the integrity of each release—outweigh the complexities introduced.

Incorporating isolated, hermetic, and ephemeral builds into Ghaf's CI/CD pipeline is a strategic initiative that bolsters our system's security architecture. Isolated builds are executed in environments separated from the developer's local environment and from other builds, reducing the risk of cross-contamination. This isolation ensures that external variables do not influence the build process, producing more consistent and secure outcomes. Furthermore, leveraging Infrastructure as a Service (IaaS) alongside Nix for the build environments themselves ensures traceability even in the construction of the builder, reinforcing our commitment to transparency and accountability throughout the software development lifecycle.

Hermetic builds take this concept further; they ensure that all dependencies are explicitly defined and included within the build environment. They are impervious to changes in the external environment, which significantly reduces the chance of introducing vulnerabilities through dependencies. Given the same source code and dependencies, these builds are reproducible, leading to the same binary output every time: this is a cornerstone of software supply chain security.

To summarize, the security benefits of isolated and hermetic builds include:

Minimized Security Risks:
Minimized Security Risks: By isolating and making the build process hermetic, we reduce the risk of incorporating vulnerabilities from the build environment or external dependencies.

Enhanced Reproducibility:
This approach ensures that the builds can be reproduced identically, facilitating easier debugging and validation.

Predictability and Reliability:
Hermetic builds enhance the predictability and reliability of the CI/CD process, which is essential for maintaining high-security standards.

Ghaf's future integration of ephemeral build environments alongside adherence to SLSA Level 3 standards exemplifies TII SSRC's unwavering commitment to security. This evolution of our CI/CD pipeline, including the planned adoption of isolated and hermetic builds, signifies a strategic move to bolster our defenses against complex software supply-chain risks. Our dedication to security propels us to continually elevate the reliability and integrity of our software, thereby assuring the highest levels of trust and safety for our software products.

CONCLUSION

The path forward is one of vigilance, innovation, and adaptability. Ghaf's integration of cutting-edge technologies, adherence to stringent SLSA Level 3 standards, and proactive steps towards ephemeral and hermetic builds reflect our dedication to the strongest software integrity.

The commitment to isolated build processes and the implementation of Azure-integrated ephemeral environments underscores an unwavering resolve: minimize the risks associated with the software supply chain. Ghaf strives to mitigate potential vulnerabilities through these multifaceted security measures, to foster a trust-centered paradigm in its relationship with stakeholders.

The journey towards a more secure digital ecosystem is ongoing. Ghaf's pioneering strategies serve as a beacon for the industry, guiding the development of secure, reliable, and trustworthy software products for a safer cyber world.