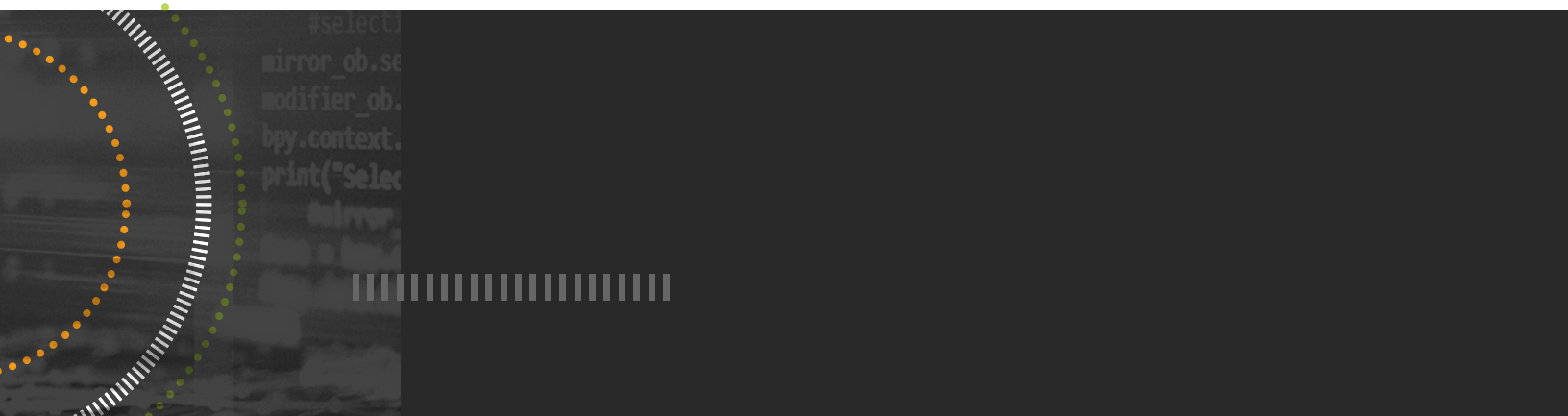
solarwinds

WHITEPAPER

Setting the New Standard in Secure Software Development: The SolarWinds Next-Generation Build System

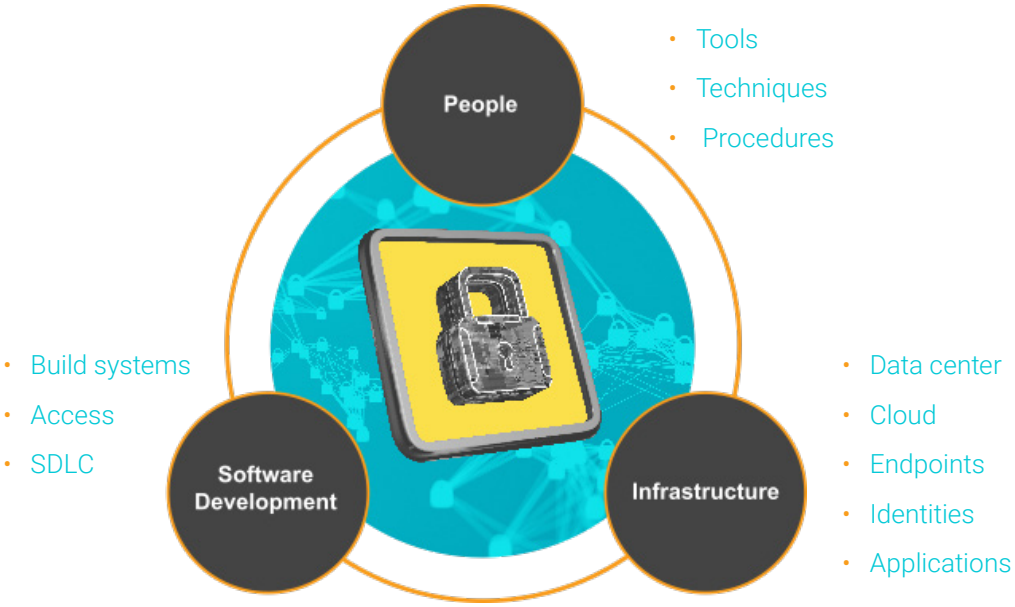


INTRODUCTION

The December 2020 SUNBURST cyberattack on the SolarWinds software build environment illustrates a concerning new reality for the software industry and illuminates the increasingly sophisticated threats made by outside nation-states to the supply chains and infrastructure on which we all rely. SolarWinds remains committed to sharing our learnings broadly given the common development practices in the industry and our belief that transparency and cooperation are our best tools to help prevent and protect against future attacks.

As SolarWinds President and CEO Sudhakar Ramakrishna laid out in a webcast with Krebs Stamos Group founding partner Alex Stamos in February 2021, Secure by Design is our guiding set of principles, with a focus on people, infrastructure, and software development. This document focuses on software development and the build process improvements we've made in an accelerated timeline this year.

SECURE BY DESIGN PRINCIPLES TO SECURE THE ENTERPRISE



The SUNBURST attack required unauthorized modification of the build supply chain and insertion of malicious code into the build environment. The design of our build system at the time of the attack was common in the software industry, and the more we learned during our investigation, the more we knew these industry-standard ideas would no longer serve today's threat landscape. With this in mind, we've completed a number of incremental changes to strengthen the integrity of the environment.

- **Phase I**, released in SolarWinds Orion Platform version 2020.2.4 on January 25, 2021, introduced a dual-build verification into our process. This enabled us to take compiled binaries back to the source code files with the associated hashes and compare those hashes with the files in source control, thus ensuring no alteration or insertion occurred within the build pipeline.
- **Phase II,** completed in July 2021, incorporates recreating the build environment in our AWS environment and adding additional security controls.
- **Phase III** is the **SolarWinds Next-Generation Build System**. The next generation-build system is in active use today, with initial products released in July and October 2021, and additional products to be released throughout 2022.

As we've designed, built, and documented this next-generation build system, we've worked and exchanged ideas with cybersecurity experts, open-source thought leaders, customers, engineers, and developers. The speed with which we've been able to bring this initiative from concept to reality has been possible only through the commitment and dedication of scores of Solarians around the globe and in partnership with all those who seek to strengthen our collective security posture—all with the goal of setting a new standard in secure software development—one we will share for the betterment of the communities we all serve.

THE SOLARWINDS NEXT-GENERATION BUILD SYSTEM

The next-generation build system needed to meet four tenets to support the Secure by Design software development principles, to help ensure resiliency against future attacks, and to provide a great developer experience. These tenets are as follows:

- **Base the system on ephemeral operations** leaving no long-lived environments available for attackers to compromise. Instead, we designed a system to spin up resources on-demand and destroy them when they complete the discrete task to which they've been assigned, thereby removing the opportunity for attackers to establish a "home base" in our systems and making it even harder for threat actors to attempt an attack.
- **Ensure build products can be produced deterministically**, for a given set of inputs, so building an artifact more than once will produce outputs which are bit-for-bit identical.
- **Build in parallel**, producing multiple, highly secure duplicates of our new build system and building all artifacts in parallel, across all systems, at once, to establish a basis for integrity checks. We refer to the products of such a system as consensus-attested builds.
- **Record every build step**, creating an immutable record of proof and providing complete traceability.

Let's look at what we designed and built to satisfy each of these tenets.

BASE THE SYSTEM ON EPHEMERAL OPERATIONS

Containerized operations have become the dominant compute paradigm across much of the SaaS world, replacing long-lived, stateful virtual machines for many kinds of workloads. Cloud-based CI/CD systems—such as CircleCI and GitHub Actions—work primarily by executing build steps inside Linux containers, which are spun up on demand and then destroyed when the relevant build job completes. It's frequently easier to control the security context of containers than those of stateful virtual machines, but simply by their very nature as ephemeral, short-lived processes, stateful virtual machines offer a significant security advantage.

Early in our design efforts, we crystallized what we call the Golden Rule of our next-generation build system: "developers shall have **fine-grained control** over the things they build—but have **zero control** over how those things are validated and secured." After evaluating CircleCI and GitHub Actions, we determined both offer container-based ephemerality and a highly desirable level of developer control over build details, but neither could give us the security controls we required, because 100% of the actions executed during a build are defined by developers, in configuration files living inside the source repository. We knew we needed to be able to subject all builds for all SolarWinds projects to a rigorous and standardized collection of security checks unable to be evaded by a sufficiently determined "inside threat" rogue actor. CircleCI, GitHub Action, and other commercial tools did not offer a us method to achieve this goal.

Golden Rule of the SolarWinds Next-Generation Build System

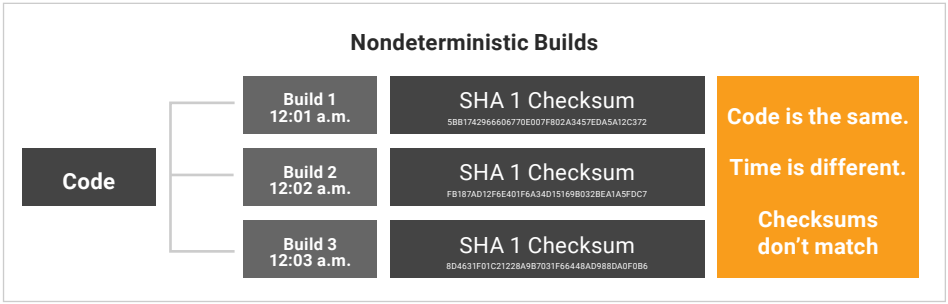
Developers shall have **fine-grained control** over the things they build—but have **zero control** over how things are validated and secured.

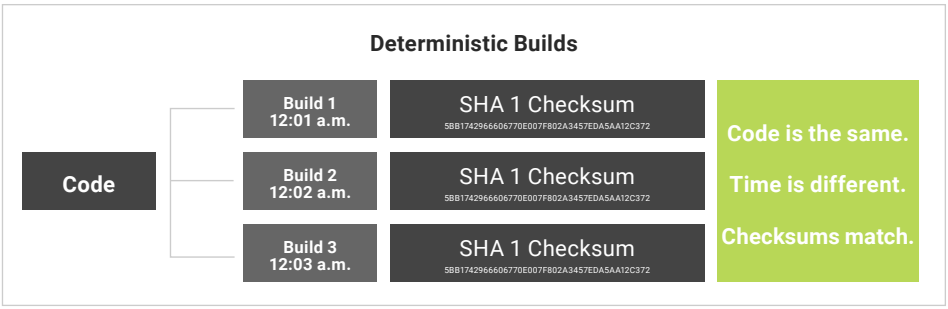
Contemporary thinking in information security identifies the insider threat as one of the most important attack vectors to mitigate against, but even so, SolarWinds has over 500 engineers at the time of this writing (and growing). A system based on trusting developers never to alter their build configurations to evade controls or elude necessary telemetry emissions simply wouldn't scale.

So, we determined we would create a next-generation build service we could eventually turn over to the open-source community, which satisfies our four tenets defined above, supports on-premises and SaaS deployments, and supports Windows and non-Windows technology stacks. Fortunately, we found a good starting point in the Tekton Project, a Kubernetes-based CI/CD framework from Google and IBM. Although it was not a complete solution for our needs, it gave us a great foundation upon which to build. Over the last several months, we created the new build service, combining a unique process design with Tekton technology to support our more than 40 products, and internally developed software, which we intend to contribute back to the Tekton Project. We aspire to make this a development standard in the software industry.

PRODUCE DETERMINISTIC ARTIFACTS

As many technical readers of this document may know, it is possible, trivial, and routine to compute a **digest** for any binary artifact (e.g., Word document, JPEG, Java archive, Microsoft installer – literally anything). This digest (also frequently called a **checksum**, though the term has other meanings) offers an unforgeable fingerprint of the artifact in the form of a unique string of letters and numbers. If even one bit of the software changes, the digest of the same file will be different.





Purveyors of both for-profit and open-source software publish digests for software components as part of their release processes. Users are invited to subject that binary file to the same trivial digest-checking process (possible on any modern computer/operating system) on their own systems to verify they've received the final product intended by the creators. If the string of letters and numbers you produce on your own system matches the one the creator published on their website, you can have a very high degree of confidence no one has subjected the artifact to tampering, and you can feel confident running it.

However, if you're a vendor trying to produce a consensus-attested build to prove supply chain integrity, there's a wrinkle: many technologies will **not** produce a bit-for-bit identical file with the same digest for the same set of inputs, particularly if timestamps or similar unique data is introduced in the build process. This means while you, as a vendor, can publish the digests for your final product to benefit your end user, the end user isn't able to build the same thing multiple times and get the same digest string, meaning these files are said to be **nondeterministic**.

As in the timestamp example, if I build the same source code now and two minutes from now, any computed timestamps inserted into the resulting files will be different, and result in different digest computations, which prevents using digests to compare the results of builds from two parallel systems.

This is the state the software development world has been in for decades and most just accepted. We can't do so any longer, because of the many high-profile supply chain attacks and new mandates in President Biden's Executive Order on Improving the Nation's Cybersecurity.

Fortunately, the software industry is moving fast to address this limitation. For Java projects using the Maven build system (which is standard for all Java at SolarWinds), there's a runtime option that can be used to strip timestamps and other non-deterministic data, resulting in a fully reproducible JAR file. For container images, such as those used in our SaaS product offerings, there are several solutions. And for .NET, Microsoft continues to progress MSBuild will also be capable of honoring a similar runtime flag to produce fully deterministic artifacts.

There are emerging standards like SLSA (Supply-Chain Levels for Software Artifacts) and CNCF's Software Supply Chain Best Practices which take this state-of-play into account and require all artifacts with a feasible path to reproducibility to use it, while acknowledging it's not universally available. In cases where it's not available, we can do things like compute digests of componentry.

For example: our Database Performance Analyzer (DPA) product, which shipped from the new build system in September 2021, is based on Java. This means the components we use to build it (the Java Archives or JAR files) have been built deterministically. However, we deliver Microsoft Installers (MSI) to customers, which **cannot** be built deterministically. We've developed tooling for cases like this, in which we record the digests of the top-level artifact (the MSI, in this example) as well as the reproducible digest available from the componentry. This gives us a basis for consensus attestation in our system, because all the actual executing code is in the JAR files, and **those** have been reproduced successfully by the consensus system.

We help drive supply chain initiatives by participating in emerging standards and contributing to open-source projects. We actively participate in SBIC with Microsoft, SLSA with Google, and OpenSSF Digital Attestation Working Group.

Recent community contributions include:

- CycloneDX module for .NET, which creates a valid CycloneDX SBOM document containing an aggregate of all project dependencies
- Tekton Chains support for writing In-Toto ITE-6 attestations
- Sigstore added support for signing with AWS KMS
- Grafeas improved gRPC support
- TektonCD support for Windows, where we helped drive requirements

BUILD IN PARALLEL

As mentioned previously, our consensus attestation system is based on TektonCD. This system is implemented as a collection of Kubernetes controllers, and we've built other controllers and Kubernetes-resident software to complement the functionality we get from the upstream project. Because it's based on Kubernetes, Tekton is very easy to deploy in multiple identical environments. We leverage this fact to have three logical environments for our next-generation build service as follows:

- **Standard Pipeline** – this is the normal day-to-day build system with which developers and security operations personnel interact. It executes build definitions resident in project repositories on GitHub and performs a variety of activities to validate them for correctness and security. Using the Tekton Chains sub-project for **supply chain security** (to which we've contributed extensively), we record each build step taken in each pipeline run, cryptographically sign it, and store it in an immutable ledger database.
- **Validation Pipeline** – Almost no one has access to this highly secure environment, which has zero ingress or egress to the internet. All build jobs are handed to the validation environment over a message bus – there's no way to interact with the system other than through highly secured and audited channels available only to DevOps personnel. The purpose of this environment is to do the **exact same build process**, producing attested build steps, just as are produced in the standard environment.
- **Security Pipeline** – The security pipeline is a collection of security checks performed at two different levels: before merge from topic branch to the main development branch, and before the development branch merge to the master branch.

RECORD EVERY BUILD STEP

Attestations are cryptographically signed statements of fact. Tekton Chains produces these for each Tekton Task executed in the pipelines, meaning for any given build, both standard and validation environments produce **receipts** of the activities taken and sign them with a key. These attestations go into a common database which can then be queried to discover whether a given source-code commit has produced the same artifacts in both systems.

ORIGINAL SOFTWARE KIT BOUND FOR OPEN SOURCE

As part of our next-generation secure build system, we've created several pieces of original software, which we intend to open source in 2022:

- A private GitHub app for validating and extending developer-defined workflows, making it far simpler and safer to integrate Tekton with GitHub. This has allowed us to rapidly scale our pilot implementation to several dozen repositories, taking advantage of the ephemeral tokens available to GitHub apps.
- A Kubernetes controller to watch Tekton Tasks and report status and log back to the GitHub Checks API, enabling a developer to see all the continuous integration (CI) results of their pull request (PR) in one place, a similar experience to GitHub Actions.
- A mutating webhook to respond to Tekton Task annotations and insert new steps, to do things like perform digest computations on build artifacts.
- A GitHub proxy that enforces secure access back to GitHub's API in the case when a Tekton Task needs to talk back to the originating repository.

SECURITY CONTROLS UNRELATED TO REPRODUCIBILITY

In addition to the consensus-attested build architecture, we have a range of other controls we've implemented, augmenting our DevSecOps practices with the concept of a **security pipeline** outside of application developer control. Our DevSecOps team has fine-grained, modular control over the content of this pipeline. Each Pull Request is subject to rapid static analysis with tools like SemGrep and dependency analysis driven by **software bill of materials** (SBOM) files.

We generate SBOM files at build time and use them in the build process in three ways:

1. To validate third-party dependencies haven't changed underlying code while still retaining previous version numbers. This helps to mitigate a third-party library code compromise.
2. To provide a comprehensive picture of the transitive dependency tree available on a current build and historical basis for auditing and forensics purposes.
3. To power a process wherein we use JFrog Xray to perform build-time checks and enforce policies based on CVSS scoring – we only attest to the security of a dependency tree if no dependencies include vulnerabilities with high CVSS scores.

OTHER SOURCES OF SUPPLY CHAIN RISK

So far, we've covered the mitigations for the kind of attack SolarWinds suffered—a trusted component of our build system was compromised. However, there's another kind of supply chain attack, which is when a trusted piece of **third-party** componentry is compromised and added to the build process by unsuspecting developers. This kind of attack was illustrated quite dramatically in January 2021 when the Codecov tool was compromised, resulting in subsequent compromise of a large number of downstream consumers of that tool.

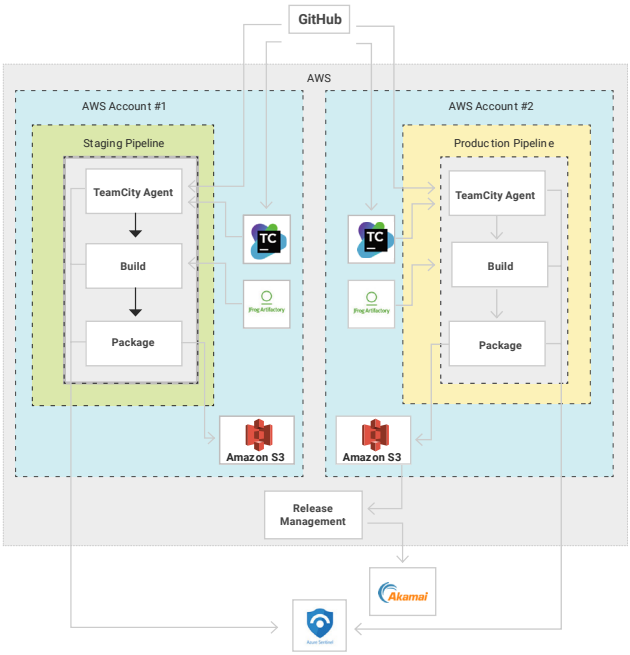
Among the lessons we learned from this compromise is, in general, you should arrange your system, so you never build directly from the internet without any intervening scanning tool in place to validate the dependencies you bring into your builds. To this end, we use an instance of JFrog Artifactory, not the cloud service, to host our dependencies, which is the only valid source for any software artifacts bound for staging, production, or on-premises releases.

SUMMARY

SolarWinds and the software industry have evolved dramatically since December 2020. Four of the lessons we've learned are as

1. Build more than once, and in parallel, to be sure you know releasing.
2. Don't build directly from the internet, and ensure you dependencies—including third-party componentry—continuously evaluate them.
3. Continue to evolve your build service in alignment with standards.
4. Leverage the power of open-source community to evolve your build service.

We anticipate we'll release the SolarWinds next-generation build service to the open-source community in 2022.

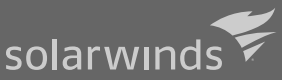


A simplified diagram of the basic architecture of our next-generation pipeline

Security is an ongoing journey with no finish line, but we feel SolarWinds, and the software industry has become smarter—and stronger—through the learnings from attacks like what we faced. These learnings, and the advancements they reveal, help make all of us stronger and more secure.

ABOUT SOLARWINDS

SolarWinds (NYSE:SWI) is a leading provider of simple, powerful, and secure IT management software. Our solutions give organizations worldwide—regardless of type, size, or complexity—the power to accelerate business transformation in today's hybrid IT environments. We continuously engage with technology professionals—IT service and operations professionals, DevOps and SecOps professionals, and Database Administrators (DBAs) – to understand the challenges they face in maintaining high-performing and highly available IT infrastructures, applications, and environments. The insights we gain from them, in places like our THWACK community, allow us to address customers' needs now, and in the future. Our focus on the user and commitment to excellence in end-to-end hybrid IT management has established SolarWinds as a worldwide leader in solutions for observability, IT service management, application performance, and database management. Learn more today at www.solarwinds.com.



For additional information, please contact SolarWinds at 866.530.8100 or email sales@solarwinds.com.
To locate an international reseller near you, visit http://www.solarwinds.com/partners/reseller_locator.aspx