CodeNotary

END-TO-END TRUSTED SUPPLY CHAIN INTEGRITY AND AUTHENTICITY INCLUDING SBOM SOFTWARE BILL OF MATERIAL

Contents

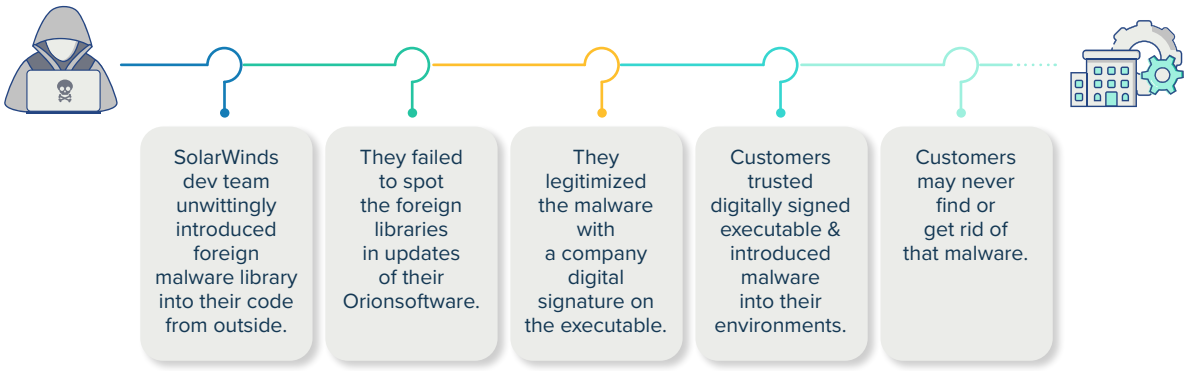
Every Software Supply Chain is Vulnerable

The SolarWinds hack of 2020 continues to make news and every day some more details reach the surface. Fortune 500 companies, large U.S. federal government agencies, security vendors, and operating system vendors were – and continue to be – affected. This is putting the spotlight on how fragile and sensitive today's software development cycle is.

Organizations which use old-world best practices and digital certificates are only nominally compliant, the SolarWinds hack clearly shows the need for additional protection. The bottom line is: keeping your CI/CD pipeline and code supply chain unprotected is like keeping your door wide open to the bad guys. And remember, the money and effort you put into automation, vulnerability scanning, and DevOps training is wasted if you don't protect your software supply chain.

Anatomy of the SolarWinds Attack

What exactly happened and why it's a problem for every organization.



18,000 +Customers Have Been Affected!

vmware intel Microsoft NVIDIA cisco FireEye belkin

SolarWinds, a popular IT monitoring and management vendor, revealed that hackers breached its systems and inserted malware into the software build process of its Orion platform. This stayed undetected for an extended period, and so SolarWinds products were shipped to customers with the vulnerability for several months. That malware in turn introduced a backdoor to access running Orion servers on customers' infrastructures.

Thousands of customers use the SolarWinds Orion platform and by updating to the newest software updates as many as 18,000 organizations installed the backdoor and got breached.

Those affected include Microsoft, the U.S. Department of Homeland Security (DHS), and FireEye, a cyber security vendor which first detected the attack (in December 2020) after the hackers stole proprietary security tools it offers to its clients.

Legitimate Malware

The real problem of this hack is its scope. The attackers found a way to **legitimize the malware** by injecting it into the build pipeline. That build pipeline started to produce digitally signed and trusted software for over 18,000 customers worldwide. Currently it's unclear how many customers have really installed the malware, but none of them would have questioned software signed and introduced by SolarWinds. That is the core of the problem with an untrusted software supply chain because the build pipeline produces builds which are digitally signed with the SolarWinds certificate trusted by the certificate authorities in Microsoft Windows, macOS and your browser. Revoking the digital certificate revokes the good together with the bad software, since you can't differentiate between these any longer in a compromised software supply chain.

Trust in Digital Certificates and Well-known Brands

How can you prevent attacks like the SolarWinds attack in the future?

The vulnerability in this case wasn't a big open firewall port or similar, but rather a very sophisticated attack of the software supply chain. There was also a component to the SolarWinds attach where FTP credentials were stored in a public Git repository. Nevertheless, it's unclear if that FTP account was even part of that attack.

In hindsight, there is clearly a need for a comprehensive, holistic **DevSecOps** strategy, that aims to provide full observability:

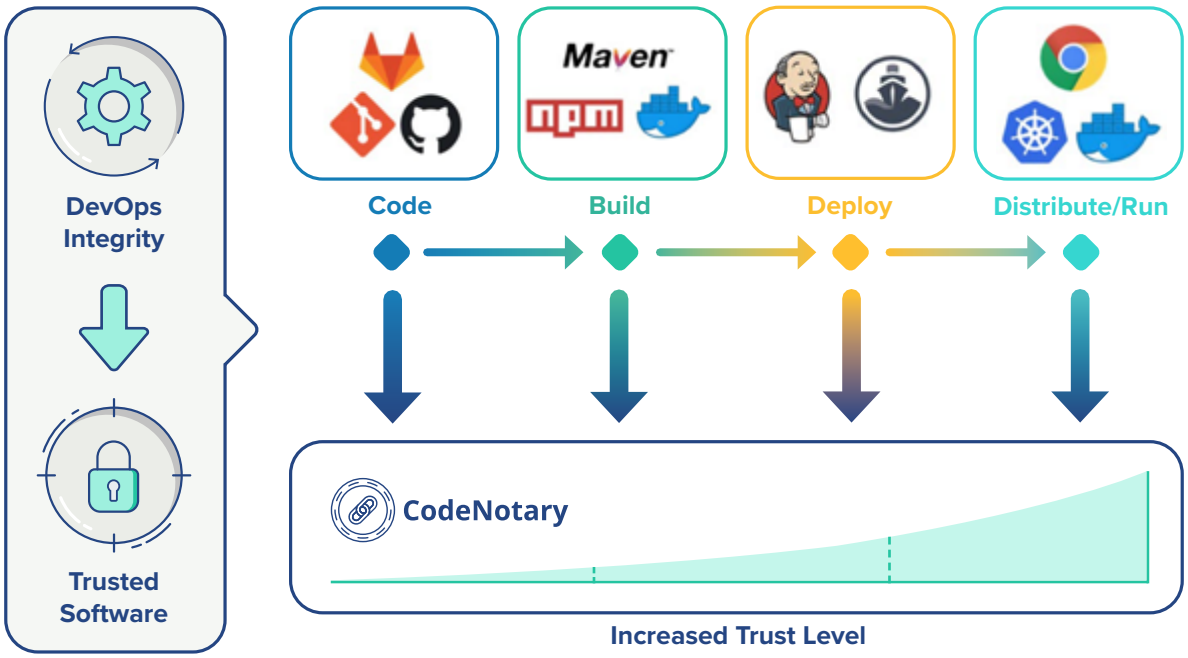
- what components are being used (bill of material)
- when are certain components or open-source projects used?
- what is currently running in my build process?
- what is currently running in my production environment?
- who did introduce what component?
- who did approve which artifacts before being used/deployed?
- what is affected if the component will be updated or **removed**

End-to-End Trusted Supply Chain

There is no doubt that highly sophisticated hackers can use the tiniest flaw to get into systems and won't stop to be very innovative. However, recent advances in immutable data storage and its integration with the software development process provide us with new solutions.

What we need is a trusted CI/CD pipeline where every used open-source project, artifact, or module is uniquely and unequivocally identified, (and compatibility tested) before being integrated into a release.

Our customers use CodeNotary to notarize every step and artifact within the pipeline by people (Developers, QA) or systems (vulnerability checkers, compatibility testing software aso.).



What allows all this to be end-to-end secure, is the unique identity of every single artifact and the bill of material of your build cryptographically verifiably stored on an ultra-fast (up to 10,000,000 transactions per second) immutable ledger.

One very important benefit is the ability to change the trust level of any digital object at any time and that everybody who needs access to that information can verify it. When using digital certificates, you can only revoke that certificate for everything you've ever signed. Whereas using CodeNotary, you can precisely change the status of a certain release or artifact with a *hitherto unseen granularity of revocation*.

Unlike with digital certificates, with CodeNotary you can untrust your release the moment you realize there is a malware or even just an unsanctioned component in it without compromising your whole trust chain.

The net effect: granular observability of your build and release lifecycle, as well as fine grained revocation of untrusted, or obsolete artifacts.

However, the combination of machine and user identity (separate from your SSO or inhouse user management), unique digital artifact identities and trust levels (**including a timestamp**) – while powerful – is worthless if stored in a normal, mutable database.

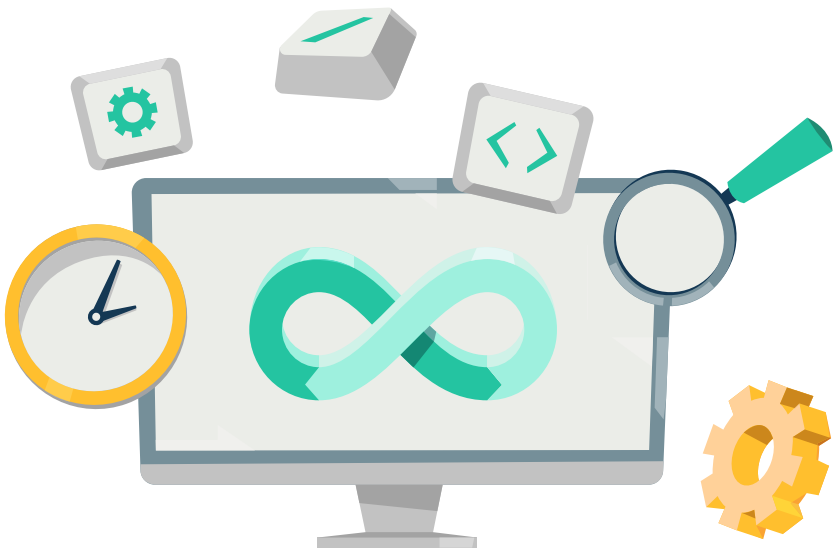
It is of paramount importance to store that information in an immutable ledger that is always accessible and maintains a tamperproof history of changes.

And that is exactly what CodeNotary provides: tamperproof trust at every stage of your pipeline from source to deployment and beyond.

CodeNotary Immutable Ledger is currently the only solution that combines a complete immutable and cryptographically verifiable data ledger integrated to protect the full software lifecycle, software supply chain, and application workload.

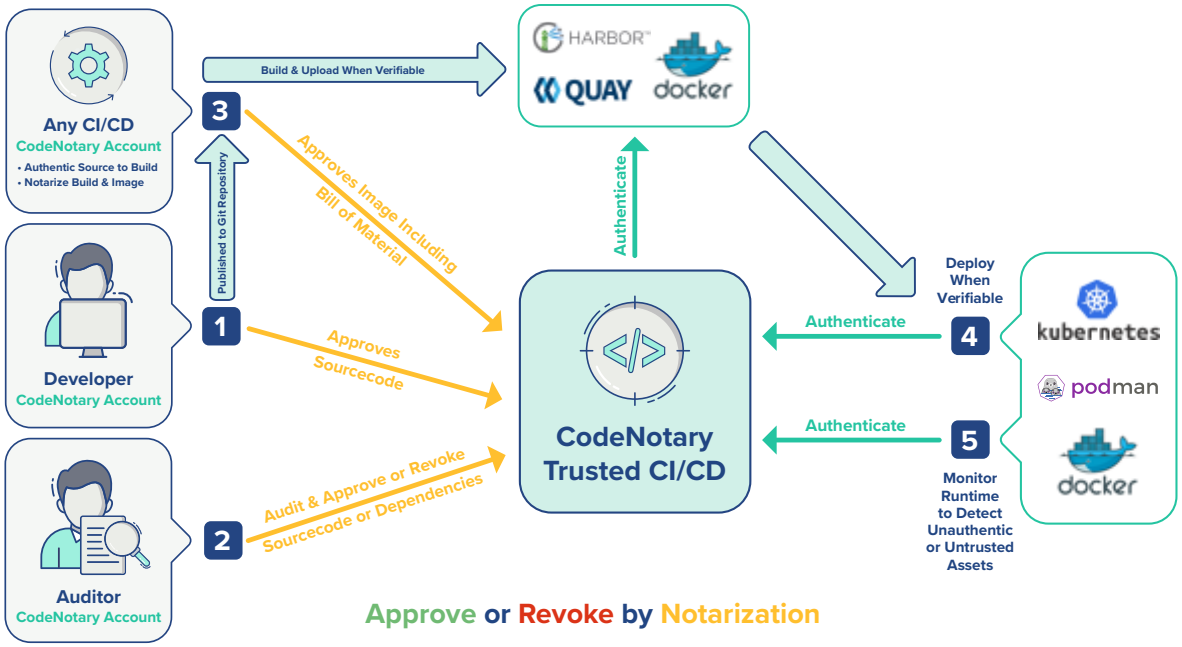
Every timestamped notarization transaction consists of a user or machine identity, the unique digital artifact identity, the trust level (trusted, untrusted, unsupported), and metadata. As every historic transaction is an immutable version, the platform is fully auditable and tamperproof.

The integration with any CI, CI/CD, and DevOps tool is easy using the CodeNotary APIs and/or command-line tools.



Quick introduction CodeNotary Immutable Ledger 2.2

CodeNotary can cover the whole supply chain and instead of signing the final binary, artifacts are notarized or authenticated at every step of the workflow. In today's complex, distributed organizations, multiple persons or machines with different logins are involved and all of them are constantly aware of the status of an object in the supply chain. With CodeNotary, it would be impossible to tamper software products as seen in the Sunburst attack.



The integration of the CodeNotary solution with your CI/CD pipeline gives you for **4 main benefits**:

1. Tamperproof history of all digital artifacts (single artifact AND bill of material of the build)
2. Machines and people can notarize and authenticate every individual artifact inside or outside your premises.
3. Notarization can be done for every individual artifact using different states (Trusted, Untrusted, Unsupported)
4. Authentication can be done publicly by anyone with internet access (i. e. to check the current state of the download or build)

The CI/CD protection can **uniquely identify**:

- git repositories (source code)

- CI/CD pipelines (Jenkins, CircleCI, GitLab, TravisCI, GitHub Actions and many more)
- Container images
- Files and folders (i. e. configuration files, build files, CI/CD recipes, installer binaries, and many more)

CodeNotary Immutable Ledger 2.2 incorporates new features targeted at protecting CI/CD pipelines:

- The *vcn* command line tool integrates with CI/CD pipelines, including an automatic collection of the whole pipeline environment for better and very fast artifact search
- Arbitrary files, eg. security scanner reports, can be attached (immutable) to the notarized artifact
- GitHub actions for notarization and authentication are available for automatic artifact and Git repository processing based on PRs or releases
- Identities (API keys) can be rotated and revoked programmatically
- *vcn* takes into account revoked keys when authenticating assets
- User interface improvements, especially in the CI/CD pipeline and artifact query view
- Multi-ledger query API keys to find artifacts across separated teams

Software Bill of Material - SBOM

President Biden's Executive Order (Section 4) requires every organization to implement an immutable Software Bill of Material (SBOM)

Zero-Trust infrastructures and SBOM are important to comply with the 2021 Cyber Security Executive Order!

We believe that every organization producing software needs to track the bill of material of its software (Software Bill of Material - SBOM or BOM). But it shouldn't just be tracked somewhere and stored in an archive system for later reference; it needs to be actively used. To make that possible, you need a solution that stores your BOM including its full history for every build immutably and tamperproof - so it can become your single source of trust.

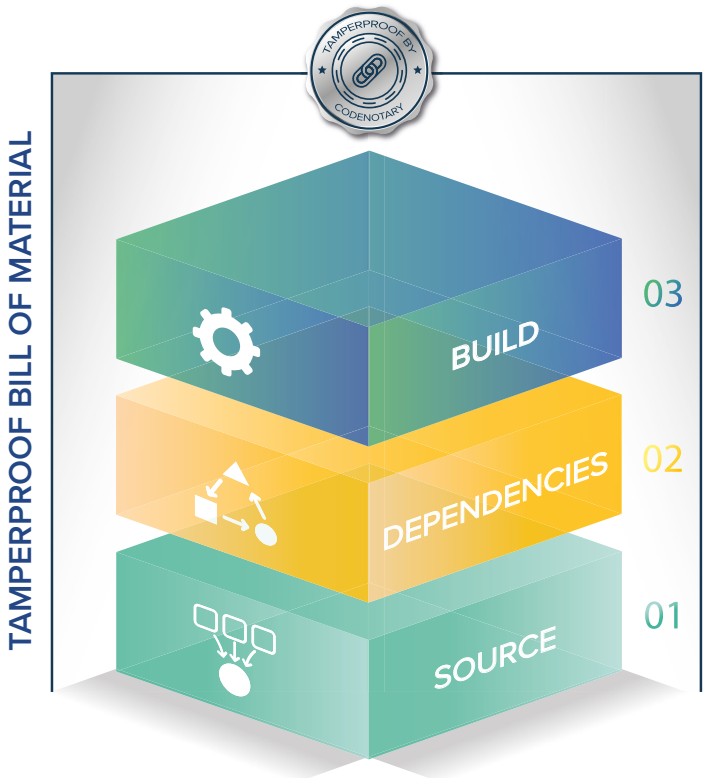
With the 2021 Cyber Security Presidential Executive Order that has just been signed, it becomes obvious that companies need to enhance the software supply chain security as one of the efforts to eventually improve the national cybersecurity standing. That requires an auditable software bill of material using Zero-Trust technologies.

The SolarWinds hack (and many others) made it clear that a vulnerable software pipeline quickly becomes a threat to every user of that software.

The massive increase of software packages being built, distributed, and used must be tracked and it needs to be transparent what packages are used in running workloads. Many packages come with many vulnerabilities and third-party binaries, and open-source projects are pulled from potentially untrusted sources.

In the U.S. and the European Union there is a legal obligation to disclose data and software breaches when companies become aware of it. The more information has been collected - and that includes a complete bill of material, recorded pipeline jobs, and the software approval process -, the better it is for further investigation and to get back to normal business as fast as possible.

CodeNotary provides the platform and the integration required to record a fully searchable and comprehensive catalog or inventory of all your software components including its dependencies and vulnerability check results for internally developed third-party, and open-source code.



What is a Bill of Material / Software Bill of Material or BOM/SBOM?

A **bill of materials** (BOM) as a complete list of all software artifacts that have been used in the build process to produce a software product. That includes the name, version, and ideally the unique checksum for all of your internal components, third-party, and open-source code.

The bill of material will be combined with vulnerability or dependency scanner results that were produced within your software pipeline. Storing all of that data tamperproof including an immutable history makes your BOM easy to use and trustworthy without the fear of having an attacker changing your BOM as well.

You need to keep in mind that many projects in the open-source world exist because of developers that solved a specific problem and shared the software with the community. You cannot expect that every project maintainer spends his spare time on making the code secure and bulletproof.

Therefore, when using these projects, all the quality and security issues are inherited and will be part of the software that you produce and distribute based on these. That puts your software and everyone who is using your software at risk, i. e. your customers or internal departments. In the SolarWinds attack, over 18,000 customers have been affected, and some have no idea that they still use malicious code.

Software developers themselves may be unaware of the vulnerabilities in dependencies within the code they reuse, and as such can’t be blamed.

An immutable SBOM is not just a list of software artifacts stored in the company archive. It's a process that continuously gathers all used software with its dependencies, including versions and known vulnerabilities. CodeNotary provides command line tools to integrate into CI/CD pipelines (GitHub, GitLab, Jenkins, CircleCI, TravisCI, and many more) and GitHub actions to automate the process of collecting the software build dependencies that ultimately leads to a BOM. Our solution CodeNotary Immutable Ledger also stores that information as transactions in an immutable and cryptographically verifiable ledger. These transactions include artifact metadata as well as vulnerability scanner reports, and the involved parties (CI/CD tool, QA engineer, vulnerability servers aso.).

1. Next to a code-to-binary integration, CodeNotary also provides a binary-to-dependency integration for Go and Java applications.
2. It's the only tamperproof and auditable solution on the market that provides a full immutable history of everything that has happened.
3. That way you know who did what when and where it is currently used - and you can prove it!

Software Supply Chain Attacks and the SBOM

Every software delivery relies on a supply chain, like in any other industry, to make sure that all components are used in the right way, at the right time, and in the correct version. This supply chain also covers everything that can affect your delivery of the software.

Therefore, the software supply chain is everything that goes into your code or affects it, from development, the CI/CD pipeline, until it's ready to be used or deployed into production. If you run your own workloads in production, then the software supply chain doesn't stop with the runtime but goes beyond to cover patches and rolling updates as well.

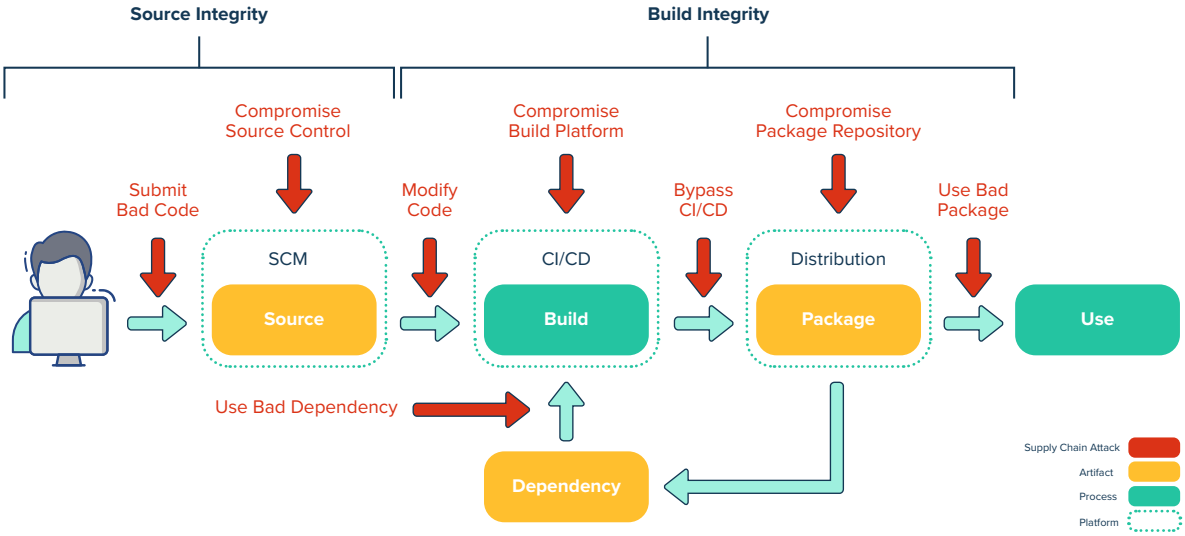
A good supply chain provides reports and information about code, binaries, other components, where they come from and what version or license has been used. Don't forget the security related part and the software component review for security issues, vulnerabilities, compatibilities, approved versions, and compliance.

Think of the who, when, where, and why of everything that went into your app.

The software bill of material is not the only but a very important part of your software supply chain management and proper practice.

The attack surfaces

Most of the software supply chain is automated in today's software development process and is mainly protected using reviews, approval processes, vulnerability scanners, and digital certificates.



Threat	CodeNotary Protects!
Submit bad code	Only accept code changes and pull requests when notarized by multiple, defined reviewer.
Compromise source control or build platform	Source code changes are notarized independently of the Source Control or Build Platform. Unauthorized changes can instantly be identified, and a build process be stopped. Furthermore, build pipeline recipes are protected (notarized and verifiable) the same way as the source code.

Use bad dependency	The integrated SBOM (Software Bill of Material) allows for notarization and verification of each used dependency. Build policies prevent usage of unknown or unauthorized dependencies.
Upload an artifact not produced by the official CI/CD process	All notarization transactions are done using the unique fingerprint of the artifact and the unique user or system identity. Everything is stored fully immutable and tamper evident. Every replacement or change of known and unknown artifacts will be detected.
Compromise package repository	Like the above threat, the package fingerprint will be stored in CodeNotary independently of the package repository. That way the attacker cannot replace packages in the compromised repository unnoticed.
Trick customers into using bad package	CodeNotary Cloud allows for public notarization and verification of any asset. Customers and users can verify if a package is origin and authentic. The publisher can untrust the package immediately if something has been compromised and the user is instantly warned. **The very popular GitHub project Home Assistant is using public CodeNotary notarization to enable full verification of all their artifacts and builds.**

That automation of the CI/CD pipeline is today the exploited attack surface and was also used in the SolarWinds hack.

If the attacker (internal, external, intentionally or by accident) adds software components or changes the process which leads to a malicious software release, the damage is already done. Using digital certificates only protects the software integrity, but not the authenticity or the level of security.

The moment that software is distributed into the runtime or to other customers, the damage spreads like wildfire, and the reputation of the development organization has been destroyed.

How can a SBOM help?

A SBOM can help in 3 ways:

- complete understanding of what went into your app
- act on security, license, or compliance changes of any of your used software components
- present and past view

If someone actively embeds malicious components in your software, having a complete and tamperproof catalog of your whole build process including all components used, still enables you to detect the bad code delayed or for forensic purposes.

But many vulnerabilities are simply part of approved and good software components that are detected way after you started using them. In these cases it's crucial that a software component that suddenly is marked critical by your vulnerability scanner can be detected everywhere it has been used before.

Having a reliable and trustable (ideally using a Zero-Trust technology) bill of material, puts you in the position of being able to trace back everything that has been used in the past, in the current build process, or is already running as a workload. Suddenly a newly detected vulnerability can be the basis for revoking software components close to real-time.

When reading all the news about recent software supply chain attacks, one big question remains:

Where to start?

Integrating CodeNotary in Your Software Pipeline to Get a BOM

The most straightforward way of integration is the build pipeline itself.

We at CodeNotary recommend a strategy where every step verifies the input and notarizes its output. In combination with a proper identity management, you can secure your pipeline in a way that unapproved software cannot be introduced into a build process.

In the build process itself, all used dependencies and requirements are notarized by the build process identity and the produced software build is notarized including all used requirements as well. That way you automatically create a BOM without any

additional effort.

Adding your vulnerability checker results to this notarization as well in our immutable ledger-based platform (uses immudb in its core) provides you with a fully auditable SBOM including an immutable history of past vulnerability or dependency checker results

```
❯ ./vcn bom /github/immudb
github.com/aead/chacha20@v0.0.0-20180709150244-8b13a72661da
github.com/aead/chacha20poly1305@v0.0.0-20201124145622-1a5aba2a8b29
github.com/golang/protobuf@v1.5.2
github.com/grpc-ecosystem/grpc-gateway@v1.16.0
github.com/lib/pq@v1.10.2
github.com/mitchellh/mapstructure@v1.4.1
golang.org/x/net@v0.0.0-20210716203947-853a461950ff
gopkg.in/yaml.v2@v2.4.0
github.com/jackc/chunkreader@v1.0.0
github.com/jackc/pgservicefile@v0.0.0-20200714003250-2b9c44734f2b
github.com/jackc/pgx/v4@v4.12.0
github.com/magiconair/properties@v1.8.5
github.com/o1egl/paseto@v1.0.0
github.com/shurcooL/sanitized_anchor_name@v1.0.0
github.com/golang/glog@v0.0.0-20160126235308-23def4e6c14b
github.com/jackc/pgio@v1.0.0
github.com/jackc/pgtype@v1.8.0
github.com/prometheus/procfs@v0.7.1
github.com/spf13/viper@v1.8.1
github.com/aead/poly1305@v0.0.0-20180717145839-3fee0db0b635
```

You can read more about it here: https://www.codenotary.com/blog/the-trusted-CICD-pipeline

Generating a Bill of Material from Binary Code Using CodeNotary

Most of your apps have already been built, so you only have the binary objects for those, but not the sources from the date of the build. As dependencies can be updated quite often, even builds between Monday and Tuesday can already differ when it comes to the bill of material.

Therefore, having access to the source code or the build protocol doesn't help you need to read the dependencies out of existing binary code.

At CodeNotary, we already have working integrations for Go and Java to gather dependencies including their unique fingerprint by checking existing software binaries to recreate a BOM.

That allows customers to find all used dependencies (name, version, checksum) by simply running our command line tool with the --bom flag and the binary and immediately notarizing them in one step. Alternatively, all used dependencies can be authenticated, to find out if any of them should not be used anymore.

```
SPDXID: SPDXRef-Package-85
PackageVersion: v4.3.2
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA256: d12400d69473b5f4c5c769a24bcb00f7b5efa2815e34e9af51e9c5e28d047158
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageComment: Trusted

PackageName: gopkg.in/src-d/go-git.v4
SPDXID: SPDXRef-Package-86
PackageVersion: v4.13.1
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA256: 491b45c95f0ac5cd143fb6821dc8a338c4063f11d298c38faf3ba5416a259181
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageComment: Trusted

PackageName: gopkg.in/warnings.v0
SPDXID: SPDXRef-Package-87
PackageVersion: v0.1.2
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA256: c055d56c563c0d8e7fc4e7b510289674a0b3665c60b2171854d9011ecb4044c1
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageComment: Trusted

PackageName: gopkg.in/yaml.v2
SPDXID: SPDXRef-Package-88
PackageVersion: v2.4.0
PackageDownloadLocation: NOASSERTION
FilesAnalyzed: false
PackageChecksum: SHA256: 0fcc60c04098ec262fc7e6369f8b01cfddc99fd251bf1762cb2a3c0937ee29a6
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageComment: Trusted

./vcn bom vcn --spdx vcn.spdx
```

Summary

Independently of the new Executive Order and government regulations at the state and federal level that are moving towards more due diligence in the software supply chain, it's in the interest of every single organization today to know the 3 W's (i.e., the Who, When, Where) when it comes to application development, deployment, distribution, or runtime.

CodeNotary is not simply providing a SBOM for your software supply chain, it also securely and immutably tracks every single step in your software pipeline in a tamperproof immutable ledger. It's crucial to have an immutable and absolutely tamperproof ledger to record every build, the SBOM, and the trust level for every single artifact over time.

That way, you can prove and query the present and the past whenever you need to know if and how you're affected by new vulnerabilities or supply chain attacks.

Try our online trial to start protecting your software supply chain: **CodeNotary CNIL Cloud**