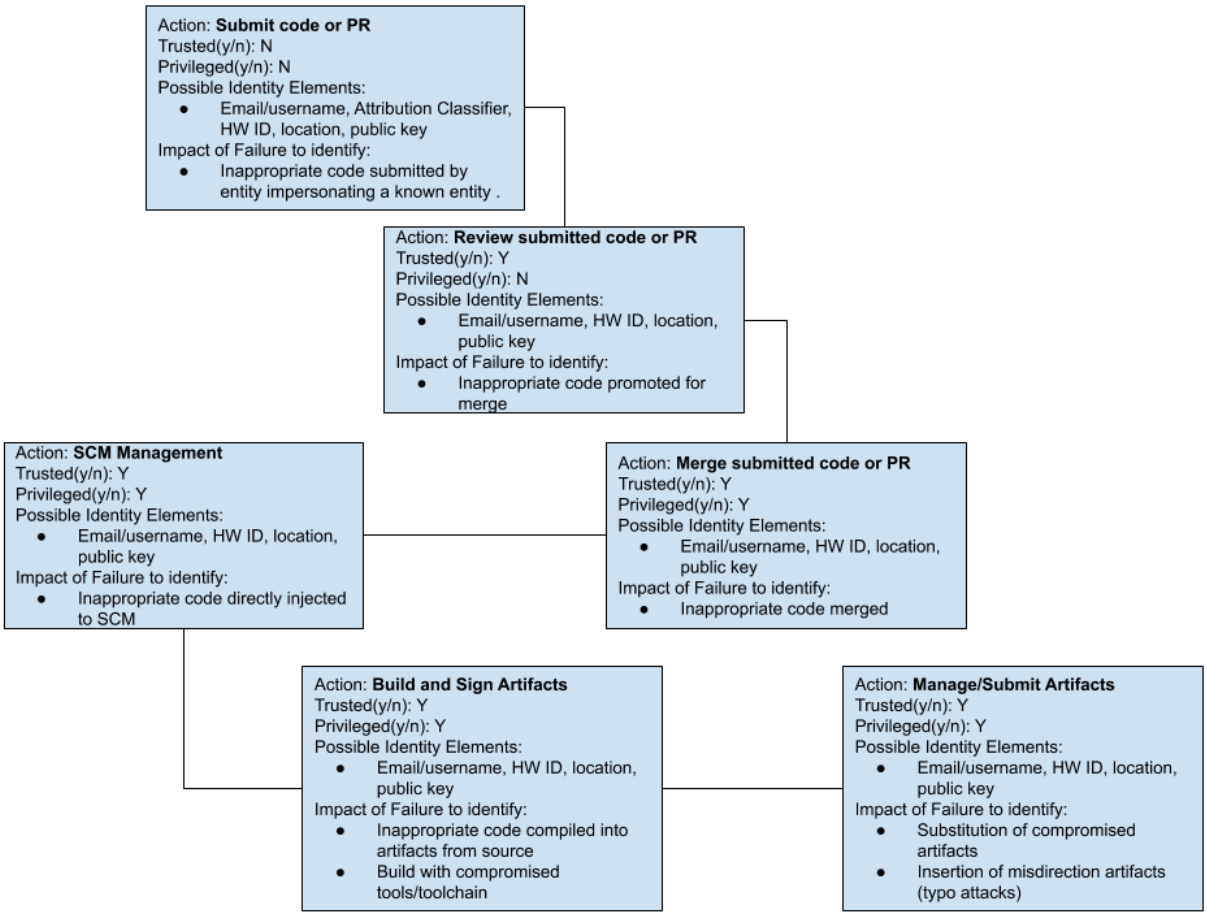
Digital Identity Supply Chain Threat Models

Author: dlorenc@google.com, WG members
Created: August 19th, 2020

Role of Identity in Supply Chain Attacks



Actions

This section lists out the different actions from the diagram above, and covers the threat models we collected below. Please add your name next to the ones that interest you and we can start working together on ways to address them.

Concepts & Terminology

- Trusted: Processes give more weight to actions/inputs by virtue of “knowing” the entity. Eg. a “known” expert reviewer within a community.
- Privileged: The entity attached to a specific identity is allowed to perform privileged functions within a system/process - eg., merge rights to a repository.
- Identity Elements: Direct or indirect data points used to attempt to independently verify the identity of an entity
- Impact of failure to identify: Potential attacks exposed by failure to correctly identify an entity

Submit code or PR

Interested parties: [klewandowski, dlorenc]
Trusted(y/n): N
Privileged(y/n): N
Possible Identity Elements:

- Email/username, attribution classifier, HW ID, location, public key

Impact of Failure to Identify:

- Inappropriate code or new dependency submitted by entity impersonating a known entity (account hijacking)
- Reputation harm to impersonated entity
- Potential licensing/copyright infringement (improper attribution)

Review submitted code or PR

Interested parties: [klewandowski, dlorenc]
Trusted(y/n): Y
Privileged(y/n): N
Possible Identity Elements:

- Email/username, HW ID, location, public key

Impact of Failure to Identify:

- Inappropriate code promoted for merge
- Unilateral access (bypassing multi-party review)
    - Eg. “Identity A” accepts code from “Identity B” and they’re either the same person or working together maliciously

SCM (source code management)

Interested parties: [klewandowski,]
Trusted(y/n): Y
Privileged(y/n): Y
Possible Identity Elements:

- Email/username, HW ID, location, public key

Impact of Failure to Identify:

- Inappropriate code directly injected to SCM
- SCM Repository Subversion
- Unilateral access (bypassing multi-party review)

Merge submitted code or PR

Interested parties: [ ]
Trusted(y/n): Y
Privileged(y/n): Y
Possible Identity Elements:

- Email/username, HW ID, location, public key

Impact of Failure to Identify:

- Inappropriate code merged

Build and sign artifacts

Interested parties: [klewandowski,]
Trusted(y/n): Y
Privileged(y/n): Y
Possible Identity Elements:

- Email/username, HW ID, location, public key, machine/system identity?

Impact of Failure to Identify:

- Inappropriate code compiled into artifacts from source
- Build with compromised tools/toolchain (developer tooling backdoor)
- Build infrastructure compromise
- Reputation harm to signing entities

Manage/submit artifacts

Interested parties: [klewandowski,]
Trusted(y/n): Y
Privileged(y/n): Y
Possible Identity Elements:

- Email/username, HW ID, location, public key

Impact of Failure to Identify:

- Substitution of compromised artifacts (artifact repository subversion, freeze attacks)
- Insertion of misdirection artifacts (cg)

List of Threat Models

Instructions: Please add threat models below! We'll start by collecting them here, then reviewing, curating and grouping them into chunks we can tackle as a group. Please leave your name/email on them to help us give you credit!

<Threat Model>

Authors: <you>
Description:
Examples:

An attacker forges/tampers release artifacts

Authors: David Wheeler, Dan Lorenc,
Description: An attacker publishes a release, pretending it is coming from a trustworthy source. This could happen in a variety of ways, including some outlined below (compromise of hosting systems, typo-squatting), and others - phishing, etc.

SCM Repository Subversion

Authors: David Wheeler
Description: An entire SCM repository can be subverted, resulting in the ability for attackers to tamper with code. This happened with the Linux kernel repository in 2011. GitHub, SourceForge, Apache, Haskell repositories have all been compromised in the past - want to limit the damage.

Compromise of hosting system management keys

Earlier in 2020 some Microsoft keys for managing GitHub repos were compromised by an insider. We want to reduce the likelihood or impact of this.

Artifact Repository Subversion

Authors: David Wheeler, Dan Lorenc, lhinds
Description: An entire artifact repository can be subverted, resulting in the ability for attackers to tamper with or publish new artifacts directly to users. This could take the form of an attack on a specific package or an attack on the entire repository.

Subresource Integrity

Authors: Srikanth Suresh
Description: CDNs and other transport layer systems can modify/tamper with artifacts during transmission.

Typo-squatting

Authors: Joshua Lock <jlock@vmware.com>
Description: malicious packages are published with names similar to popular packages, for example: through misspelling, additional name components, homographs.

Malicious Maintainer

Authors: Joshua Lock <jlock@vmware.com>
Description: a maintainer – perhaps an unknown developer who volunteers to take over maintaining a project, or a known credible maintainer – uses their privileges to introduce malicious code or dependencies. This happened in 2018 with the npm event-stream package.

Malicious Contributor

Authors: Dan Lorenc <dlorenc@google.com>
Description: Someone sends a PR with malicious code inside of it, possibly obfuscated, hoping it will be merged.

Developer Tooling Backdoor

Authors: Joshua Lock <jlock@vmware.com>
Description: a compromised software component in the developer toolchain, i.e. a compiler backdoor, introduces malicious behaviour (malware) into software at build time. This happened in 2015 with a compromised version of Apple's XCode, dubbed XCodeGhost.

Build Infrastructure Compromise

Authors: dlorenc@google.com, jlock@vmware.com
Similar to developer tooling backdoor above, but on a shared build or release system.

Account Hijacking

Authors: Dan Lorenc <dlorenc@google.com>

A maintainer can have their credentials stolen, machine stolen, giving an attacker direct access to SCM systems, CI/CD systems or artifact management systems.

Insecure code

Authors: Matt Thompson <m@thompson.gr>
Description: Flaws introduced by poor coding standards (e.g. SQL injection)

Dependency availability

Authors: Matt Thompson <m@thompson.gr>
Description: Apparently covered in other projects (https://github.com/ossf/wg-securing-critical-projects) - A dependency becomes unavailable or unpatchable meaning the project is compromised.

Developer Privacy subverted

Some developers may want to remain pseudonymous. That should be supported, though making it clear that it's a pseudonym (so projects/users may choose to reject if they want).

Insider person/organization

A developer or developing organization may be malicious. Eg. if two developers are required to review and merge code, the malicious developers could be working together to achieve their goals. This could scale out beyond two devs.

Inability to verify developer/organization properties

Some projects require developers to have certain properties, e.g.

- 18 years or older (to determine if they can make legal agreements).
- Who is their employer? Are they a prohibited entity in some country?
- Country of origin

Inability to verify that two different identities are two different people. Probably need third party attestation for this

Developer Identity Theft

Authors: zdlorenc@google.com
A malicious individual masquera des as a well-known contributor, using their reputation/trust/goodwill to publish code or artifacts.

Signing work (commits, artifacts) can help mitigate this, but only with another form of verification (PKI, TOFU, etc.), and only if it is used. The attacker does not have the original person's credentials, but they are pretending to be that person through other methods.

Freeze Attacks

Attacker makes it so the user doesn't see updates. See TUF.

SCM Repository TakeDown

Authors: Kengo Suzuki <kengoscal@gmail.com>
Description: Similar to "SCM Repository Subversion" on top of the list, but more focused on the threat that may harm the availability of the repository, such as DoS/DDoS attack. This could take the form of an attack on a specific package by encrypting and locking down the package.

See other related works for threat model info

TUF

E.g., "Freeze attacks" (malicious mirrors trick you into not knowing about updates)
https://theupdateframework.io/security/#attacks-and-weaknesses

"Threats, Risks & Mitigations"

"Threats, Risks, and Mitigations in the Open Source Ecosystem" - Michael Scovetta, Microsoft in collaboration with the Open Source Security Coalition
https://github.com/ossf/wg-identifying-security-threats/blob/main/publications/threats-risks-mitigations/v1.1/Threats%2C%20Risks%2C%20and%20Mitigations%20in%20the%20Open%20Source%20Ecosystem%20-%20v1.1.pdf

SafeCode Whitepaper

https://safecode.org/tactical-threat-modeling/
https://www.safecodedev.wpengine.com/wp-content/uploads/2017/05/SAFECode_TM_Whitepaper.pdf ← broken link

"Improving Trust and Security in Open Source Projects" (TSI)

See here for some ideas:

https://www.linuxfoundation.org/wp-content/uploads/2020/02/improving_trust_security_in_oss_projects.pdf
(Note: Some are overly strongly stated, Wheeler tried to tone it down, but this has some ideas that are probably worth mining. In particular we must acknowledge the need for user privacy.)

Securing The Public Go Module Ecosystem

https://go.googlesource.com/proposal/+/master/design/25530-sumdb.md

Breaking trust: Shades of crisis across an insecure software supply chain

https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/

(Linux kernel proposal) Header-Based Patch Attestation

“Header-Based Patch Attestation” is a proposal by Konstantin Ryabitsev on how to handle attestation within the Linux kernel development process, which is maling list based.

A draft of this is here:
https://git.kernel.org/pub/scm/linux/kernel/git/mricon/patch-attestation-poc.git/tree/README.rst

Requirements

Must be easy to use

Must provide easy/obvious benefits

Portability: Identity can port between systems

Old Comments

Account Hijacking