The SolarWinds Attack and Farm-to-table Methods in the Development Process: Notes

Mitigating disasters through supply chain security

Felix Pojtinger

2022-01-06

Topic: The “Solarwinds” attack and farm-to-table methods in the development process - Mitigating disasters through supply-chain security

Part 0: The SolarWinds Attack (Highly Evasive Attacker Leverages SolarWinds Supply Chain)

Part 0: The SolarWinds Attack (Highly Evasive Attacker Leverages SolarWinds Supply Chain)

- Summary
  - On 13 December 2020, FireEye detected a large supply chain attack targeting SolarWinds Orion
  - The actors behind the attack (tracked as UNC2452) gained access to data and control over both private and public institutions
  - Trojanized updates were used to get access to SolarWinds Orion since as early as Spring 2020
  - Result of the attack is lateral movement and data theft
- Backdoor
  - SolarWinds.Orion.Core.BusinessLayer.dll is a signed component of Orion which communicates with external servers using HTTP
  - After laying dormant for about two weeks, it receives and executes commands ("jobs")
  - Network traffic is masqueraded as the Orion Improvement

Part 1: Overview (Risks in the Software Supply Chain)

Part 1: Overview (Risks in the Software Supply Chain)

- As the SolarWinds attack has shown, supply chain attacks on any step of the supply chain can lead to significant breaches
- Let's take a look at the potentials vulnerabilities in a supply chain
- Security is a lifecycle issue:
  - Mission thread
  - Threat analysis
  - Abuse cases
  - Architecture and design principles
  - Coding rules and guidelines
  - Testing, validation and verification
  - Monitoring
  - Breach awareness
- Historically, software development didn't have a supply chain
  - Software was limited in size, function and audience
  - Each organization had their own developers
  - Each organization created their own software

Part 2: Framework (Socio-technical Framework for Threat Modeling a Software Supply Chain)

Part 2: Framework (Socio-technical Framework for Threat Modeling a Software Supply Chain)

- Now that we've analyzed the risks associated with supply chains, let's take a look at how to model its vulnerabilites from a social and technical perspective
- Software supply chains are similar to traditional supply chains
- A supply chain is created by deploying and using a product directly or reproducing it as a new product in repetition
- Traditional supply chains can have risks
  - Late product delivery
  - Counterfeits
  - Human errors
- Software supply chains have risks too, i.e. faulty code (intentional or unintentional)
- Risk management is used to counteract these known vulnerabilities
- The first step is to create a threat model of the system

Part 3: Implementation (in-toto: Providing farm-to-table guarantees for bits and bytes)

Part 3: Implementation (in-toto: Providing farm-to-table guarantees for bits and bytes)

- Using this social and technical abstract, let us now take a look at a concrete implementation of a supply chain security system, in-toto
- Examples of supply chain attacks
  - Version control systems: Linux kernel, Gentoo and Google
  - Build systems: Fedora, which allowed for signing backdoored version of security packages
  - Build environment: CCleaner
  - Software updaters: Microsoft, Adobe, Google and Linux distros
  - Are now also used by nation states against foreign states and own citizens
- Current state
  - Supply chain security is limited so securing individual steps
  - Git commit signing: Controls which devs can modify what in a repo
  - Reproducible builds: Enables building the software multiple

Part 4: Evaluation (in-toto: Providing farm-to-table guarantees for bits and bytes)

Part 4: Evaluation (in-toto: Providing farm-to-table guarantees for bits and bytes)

- Finally, let's analyze the results that the in-toto maintainers provided following some initial usage
- Debian rebuilders
  - Reproducible builds are bit-by-bit reproducible, so it is possible to build a package on a separate host and get the same hash on the result
  - A apt—transport for in-toto is used to provide attestations of the resulting builds using link metadata
  - Allow cryptographically asserting that a Debian package has been reproducibly built by k out of n rebuilders and the Debian build farm
  - Modification of a package would require breaching at least k out of n rebuilders, which the client can verify
- Cloud native builds with Jenkins and Kubernetes
  - Cloud-native/containerized environments require high levels of automation