DA2210- Essay

Supply Chain Attacks and Countermeasures

Johanna Loev, Viktor Aryd

May 3, 2022

1 Introduction

Third-party party code is used in the production of most new software, as monolithic application development is becoming obsolete. This has lead to increasing risks of so called software supply chain attacks. The European Union Agency for Cybersecurity (ENISA) defines supply chain attacks as a combination of at least two attacks. The first attack is on a software supplier, which then becomes an attack on one of its customers. In order for the attack to be classified as a supply chain attack both a supplier and its customer must be targeted [1, p. 6].

As the amount of available code and services rises, so does the demand. Products use more and more external code, which makes supply chain attacks a growing threat. Sonatype, a company researching in the area, stated the attacks increased by 650% between 2019 and 2020 [2]. This essay will bring up some common characteristics of these attacks and some preventive techniques.

2 Software Supply Chain

Just like how the production of physical products requires a supply chain of raw materials and smaller components, the production of software nowadays requires a supply chain of third-party party code. Software supply chain refers to the every software necessary to produce new software [3]. It can mean "libraries, code, hardware, and tools that transform code into a final deliverable" [4]. The deliverable can either be a user-face interacting application, a service or a package used in software which other products depend upon.

The software supply chain comes with some significant differences to the traditional manufacturing supply chain:

- Intangible - the software supply chain is an intangible object made up of a series of virtual and digital components that make it extremely difficult to count, discover, and understand their operation end-to-end.
- Mercurial - as more teams reduce their time to market, it is reasonable to expect the shape and structure of a software supply chain to continue to change.
- Iterative reuse - one product's supply chain, is n number of products and nn supply chains, an endless supply of turtles all the way down.[5]

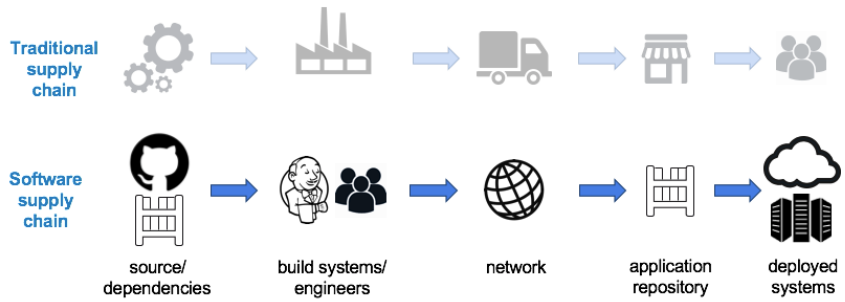

Figure 1: The differences between a Traditional Supply Chain and the Software Supply Chain [5]

Furthermore, the Software Supply Chain should not be confused with Supply Chain Software which refers to software used to manage the supply chain integrating retailers, manufactures, customers, transporters and warehouses, such as SAP SCM or E2Open for instance [6].

3 Software Supply Chain Attacks

There are many types of supply chain attacks. Lots of different attacks can be classified as supply chain attacks, as long as it targets a customer through a supplier. ENISA classifies different supply chain attacks by splitting them into 4 parts: techniques used to compromise supplier, supplier assets targeted, techniques used to compromise customer, and customer assets targeted [1, p. 7-10]. The first two parts are essentially a non-supply chain attack on the supplier, and uses general attack techniques such as Malware Infection, Social Engineering, or Exploiting Vulnerabilities. What differs here is what the attacker is after, which in the case of a supply chain attack is some way to access customers. The second two parts of the ENISA specification constitute a second attack on the customer of the compromised supplier. The attack techniques here differ a bit from other attacks due to the resources gained from the supplier attack. For example, automatic updates can be exploited to deliver malware, or the supplier can be impersonated [1, p. 7-10].

SUPPLIER		CUSTOMER	
Attack Techniques Used to Compromise the Supply Chain	Supplier Assets Targeted by the Supply Chain Attack	Attack Techniques Used to Compromise the Customer	Customer Assets Targeted by the Supply Chain Attack
Malware Infection	Pre-existing Software	Trusted Relationship [T1199]	Data
Social Engineering	Software Libraries	Drive-by Compromise [T1189]	Personal Data
Brute-Force Attack	Code	Phishing [T1566]	Intellectual Property
Exploiting Software Vulnerability	Configurations	Malware Infection	Software
Exploiting Configuration Vulnerability	Data	Physical Attack or Modification	Processes
Open-Source Intelligence (OSINT)	Processes	Counterfeiting	Bandwidth
	Hardware		Financial
	People		People
	Supplier		

Figure 2: The different types of the four parts of a supply chain attack as defined by ENISA. [1, p. 7]

3.1 Solarwinds Orion

One of the recent supply chain attacks which came up multiple times during the research for this essay was an attack targeting the company SolarWinds. The attack was on the Solarwind service Orion, which is a Network Management System. What made this attack especially concerning is the number of customers using Orion. Customers included The US Department of Defense, 425 of the US Fortune 500 companies, as well as 300,000 others [7].

According to ENISA [1], using their way of classifying supply chain attacks, it can be described thusly: To gain access to Solarwinds the attackers exploited software vulnerabilities, social engineering, and a brute force attack. This gave them access to change the source code of Orion. The customers were attacked via their trusted relationship with Solarwinds in the form of a software update containing malware. This finally let the attackers steal data from several customers, 18 000 of them according to Solarwinds themselves [8].

3.2 Codecov

Codecov is as the name implies a code coverage tool used by customers in their CI. In April 2021 a supply chain attack targeting Codecov was discovered by one of their customers by a hash mismatch. It turned out this breach had been ongoing since early January of that same year. While Codecov may not be as widely used as Solarwinds Orion, it still has a large user base. Thus, considering how this attack went undiscovered for 3 months its severity should not be understated. Attackers gained access to Codecov by exploiting a faulty Docker configuration that leaked certain credentials. This let them add a single line to a bash script, which sent all environment variables to the attackers [9]. The fact that only a single line was changed could be one of the reasons the attack

went unnoticed for so long. The main result of the attack was the attackers gaining access to git credentials, which let them clone private repositories and gain access to source code.

Using the Enisa classification, the attackers exploited a configuration vulnerability to gain access to a Codecov code. They then exploited the trusted relationship with Codecovâs customers to include their changes in an update. This let them access the source code of the affected customers.[1, p. 34]

4 Secure the software supply chain

When it comes to securing the software supply chain the two main stakeholders are customers receiving a third-party component or code, and suppliers providing their software products. In the supply chain it is important to understand that the supplier could also be a customer and vice versa. When it comes to securing the software supply chain one can look at the delivery process and the relationship established between the two parties. However, it is important to look at the software development process itself.

4.1 Secure delivery between customer and supplier

According to ENISA [1, p. 27], the customer should identify all suppliers and service providers and continuously monitor for risks and threats associated with each one of them. Furthermore, ENISA states the importance of customers maintaining good management of the relationships with their suppliers. Classifying assets and information which the two parties share and establishing procedures for access and handling of these is key. The customers should also make sure there are appropriate agreements established and that there are well functioning processes to update these supplier agreements. Not the least when there are changes made in tools and technologies used. The customer should also aim to be provided an assurance from the supplier no hidden features or backdoors are knowingly included.

However, the former wonât matter much unless the suppliers take their own responsibility. The supplier must ensure the âinfrastructure used to design, develop, manufacture, and deliver products, components and services follows cybersecurity practices as well as implementing secure engineering processesâ[1, p. 28]. They should to the best extent possible ensure the third party components used within the product offered also uphold security standards. It is important the supplier offers a Conformance Statement to their customers with accepted standards like for instance ISO/IEC 27001, IEC 62443-4-1, IEC 62443-4-2. Most importantly they should implement regular audits to ensure all intended measures are met [1, p. 28].

Furthermore, ENISA believes âsuppliers should implement good practices for vulnerability managementâ [1]. They should monitor security vulnerabilities encountered both internally and externally affecting the software delivered. The supplier should handle the vulnerabilities in the form of patches. A good practice is to use patch verification and testing which the supplier ensures to uphold standardized safety, legal and cybersecurity requirements. They should also maintain policies for treatment of such vulnerabilities and processes for informing customers [1, p. 28].

4.2 Secure the software development process

As argued by ENISA [1, p. 29] and the cloud native computing foundation, CNCF it is not only the relationship between customer and supplier and the delivery process itself which requires good management. Securing the software development process at one site is of equal, if not greater importance. As ENISA states in their report, suppliers must ensure â...the integrity and origin of open source software used within any portion of a productâ [1, p. 29]. However, CNCF further points out it is not only open source but any third-party code which needs to be secured before delivery to customers[5, p. 10]. Supply chain Levels for Software Artifacts (SLSA) and Software bill of materials (SBOM), are two important tools used for improving artifacts integrity and origin.

4.2.1 SLSA

SLSA is a set of technical controls and standards a supplier of code can use in order to improve the integrity of their artifacts. The framework creates an automatic capability to analyze whether artifacts can guarantee the source code is the original, that it is protected from interference in the build and distribution process, and that it is isolated from any hidden vulnerabilities. In the case of code being tampered with it can tell what component of the system is affected [10].

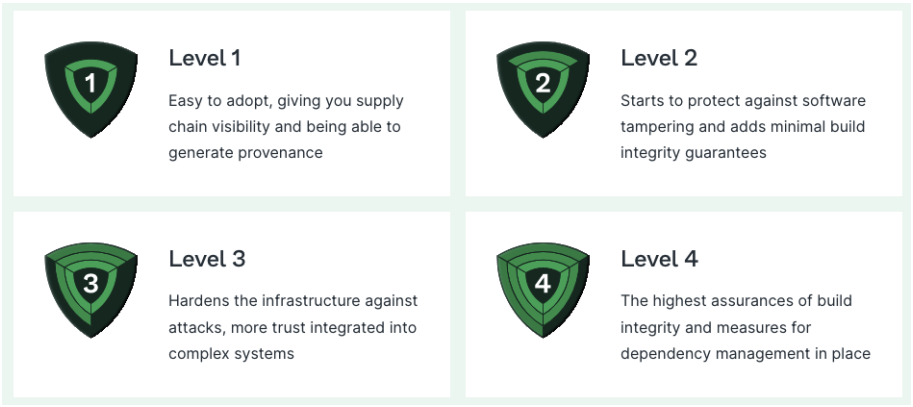

Figure 3: SLSA's four different levels [11]

SLSA offers a way to speak about how secure software artifacts are by using different levels of assurance. The lower steps are basic, easy to adopt and ensure supply chain visibility whereas the higher steps scale up the integrity to protect a system against more advanced threats [11].

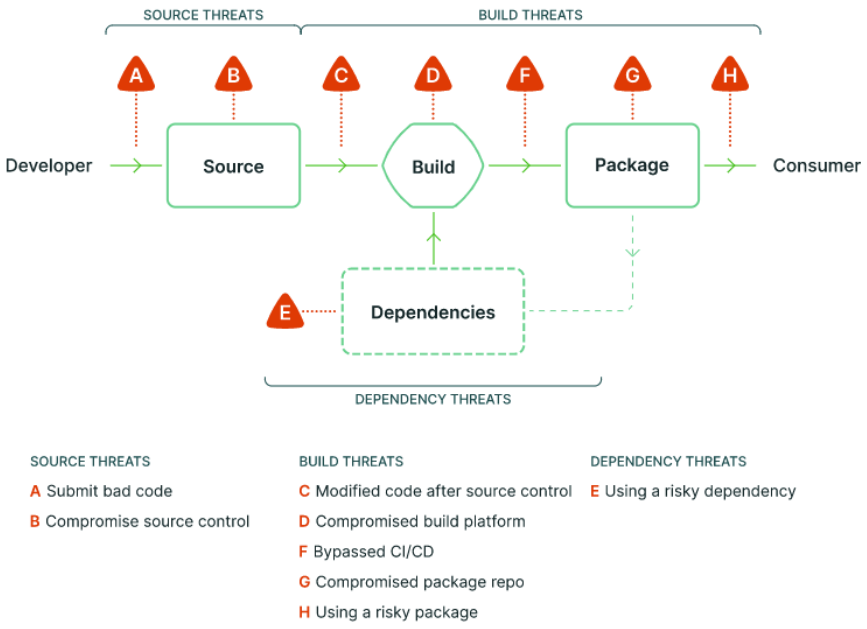

Figure 4: SLSA's framework [10]

SLSA divides up the software creation into three main areas all of which require securement. These are the build, the source and the dependencies. Each level treats all three areas when scaling up the protection. SLSA starts with securing the build, the last step before an artifact is released. It makes sure the software is built from the correct sources and dependencies, at some higher levels it also ensures build platforms are not compromised and CI/CD are not bypassed after source control. SLSA then looks at the source integrity. At this state the framework makes sure all code looks as intended by the software producer. Here it is important the code changeâs history is available for inspection. At a higher level it also ensures the system is protected from bad code submitted without review and that the source control system has not been compromised. Finally the framework looks at the dependencies and applies the same checks on all of these in a recursive fashion[10].

4.2.2 SBOM

Just like BOM is a common concept used in traditional manufacturers as part of the supply chain management, an SBOM is used similarly in the software supply chain. An SBOM is a list of components used in a software. It lists both open source and commercial software components and is used to provide transparency both for the customer and to track down and fix vulnerabilities. Though the term SBOM is just the name of a concept or process there are many companies providing automated tools to create SBOMs. WhiteSource, Sonatype and Anchore are three examples. SPDX which is nowadays used inseparably with SBOM is an open source service such as the SPDX Software Bill of Materials (SBOM) Generator [.]

4.2.3 GitHub Features

Github which is one of the largest platforms for open source code projects offers tools for the open source community in order to assure protection. Github offers not only information on how to secure against supply chain attacks but some tools for the community using Github. They provide features such as a dependency graph and dependency review to identify changes and information to help the reviewer understand how the latest push impacts the system. They also provide so called dependabots alerts and updates which uses the data provided by the dependency graph to search for known vulnerabilities in the GitHub Advisory Database. It generates alerts if vulnerabilities are detected and even runs updates to help update the dependencies with known vulnerabilities [12].

More importantly Github is also aware of the importance of securing the end-to-end supply chain [13]. They state attackers nowadays are not only targeting dependencies you use but focuses more on user accounts and build processes. Therefore, it is important to secure and configure your personal account correctly, Github offers two-factor authentication (2FA) [14]. It also helps you sign builds with encryption keys. You sign the build with your private key and let others use your public key to verify the signature on the build before they use it [**github˙sec˙builds**].

5 Conclusion

As the open source and commercial code continues to grow and the demand for it increases when producing new software the attacks are likely to increase in number and become more complex. Some of the latest data breaches such as Solarwinds and Codecov shows how important it is to not only secure the source code but also secure the build processes. Even though it is impossible to prevent any sort of attack there are few countermeasures recommended by the agencies and companies specialized in this area. Some argue that the relationship between supplier and customer must function well. The customer must ensure the communication to their supplier works effectively, revisions of practices are done regularly and should acquire appropriate systems and processes for patching vulnerabilities as soon as the supplier detects one. At the same time the supplier must ensure good practices and processes of scanning for vulnerabilities in their product, as well as making sure the dependencies used in their software contains no vulnerabilities. Both parties will however benefit from securing their development process. SLSA, SBOM are good frameworks to align with in order to achieve certain levels of security in the supply chain. Github's features may also be used for similar purposes and could benefit the open source community in this area.

References

[1] "Threat Landscape for Supply Chain Attacks". en. In: *ENISA* (). URL: https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks (visited on 05/02/2022).

[2] *Sonatype's 2021 State of the Software Supply Chain*. URL: https://www.sonatype.com/resources/state-of-the-software-supply-chain-2021 (visited on 05/02/2022).

[3] Luke Mcbride. *Software Supply Chains: an Introductory Guide*. en-us. URL: https://blog.sonatype.com/software-supply-chain-a-definition-and-introductory-guide (visited on 05/02/2022).

[4] Ãlvaro Iradier. *Secure software supply chain: why every link matters*. en-US. Nov. 2021. URL: https://sysdig.com/blog/software-supply-chain-security/ (visited on 05/02/2022).

[5] *CNCF Security Technical Advisory Group*. original-date: 2018-03-13T22:30:42Z. May 2022. URL: https://github.com/cncf/tag-security/blob/c400ef76e52376c34c7c75e9ed69ccc3e2b797f1/supply-chain-security/supply-chain-security-paper/CNCF_SSCP_v1.pdf (visited on 05/02/2022).

[6] *Top 15 Supply Chain Management Software in 2022 - Reviews, Features, Pricing, Comparison*. en-US. Apr. 2022. URL: https://www.predictiveanalyticstoday.com/top-supply-chain-management-software/ (visited on 05/02/2022).

[7] *What You Need To Know About the SolarWinds Supply-Chain Attack — SANS Institute*. URL: https://www.sans.org/blog/what-you-need-to-know-about-the-solarwinds-supply-chain-attack/ (visited on 05/02/2022).

[8] *8-K Form NR 001-38711*. en-US. Dec. 2020. URL: https://www.sec.gov/Archives/edgar/data/0001739942/000162828020017451/swi-20201214.htm (visited on 05/02/2022).

[9] *Codecov supply chain attack breakdown*. en. June 2021. URL: https://blog.gitguardian.com/codecov-supply-chain-breach/ (visited on 05/02/2022).

[10] *Introduction*. en. Apr. 2022. URL: http://slsa.dev/spec/v0.1/index (visited on 05/02/2022).

[11] *Supply-chain Levels for Software Artifacts*. SLSA. URL: http://slsa.dev/ (visited on 05/02/2022).

[12] *About supply chain security*. en. URL: https://docs.github.com/en/code-security/supply-chain-security/understanding-your-software-supply-chain/about-supply-chain-security (visited on 05/02/2022).

[13] *Securing your end-to-end supply chain*. en. URL: https://docs.github.com/en/code-security/supply-chain-security/end-to-end-supply-chain/end-to-end-supply-chain-overview (visited on 05/02/2022).

[14] *Best practices for securing accounts*. en. URL: https://docs.github.com/en/code-security/supply-chain-security/end-to-end-supply-chain/securing-accounts (visited on 05/02/2022).