Chapter 1

Introduction

Supply-chain attacks have become an increasing threat towards computer systems and networks. The motivation for this kind of attack is relatively easy to understand: One can control entire organizations and even governments and countries if one holds the means to compromise the computer networks and their supply chains. An attacker could acquire enormous monetary funds by manipulating the economic systems or the decentralized financial systems of cryptocurrency blockchains such as NFTs. The attacks are possible when the targets are inherently vulnerable from the trust in their creators and trust in their supply-chain i.e. the hardware constructors and the software authors, the upstream development team, and the dependencies in the supply chain. Software developers normally don't expect an attack on their code while it is being built by the build system. An attacker could put in a backdoor (Trojan horse) in a program or a system as it is being built.

System owners are concerned with confidentiality, integrity, and availability, where availability is at least as necessary as the others. What if documentation gets deleted or destroyed, or as in a chat, it sometimes automatically becomes unavailable after six months? The work with this project, as described in this report, has been to investigate and understand the vulnerabilities and the means of mitigation.

Changing the compilation pipeline of real software without it being seen might sound difficult, but it can be done. In this project, there is an investigation and feasiliibility study and a case study with the design of such an attack. It includes an implementation and evaluatation of the method of defense, named diverse double compiling This report also suggest a few completely novel ideas to reduce the assumptions and reduce the

probability even more that a system is compromised in chapter 2. Cybersecurity can be accomplished primarily in two complementary ways: secure software development, a NIST initiative, and software vulnerability protection.

1.1 Hypothesis Statement

Trusting Trust - The hypothesis being worked with is: To what extent can cryptocurrency software be secure from supply-chain vulnerabilities? There are methods to reduce the attack surface of a computer system in general. The key idea is that to believe in a 100% secure system, every piece of software and hardware must be trustworthy that has ever been used to create your system. During the research of the topic, related work was surveyed and studied, techniques were created, and evaluated for new and established methods of diverse double-compiling in the context of the build system for interacting with the blockchain of the Bitcoin cryptocurrency [35]. To attack and compromise a transaction on a blockchain one might consider a few attack methods and exploits that can be tried and analyzed, for example:

1. Steal a PGP key from a maintainer and using that to sign new versions of Bitcoin-Core
2. Conduct a domain-specific attack where the compiler inserts code that changes the recipient address of a transaction, possibly more difficult to detect if it happens only in 1/10000 Bitcoin transactions.
3. A malevolent maintainer could upload malicious code and hide it. That would get normally caught by a verify signatures script

The hypothesis is: To what extent can we reduce the risk for a cryptocurrency transaction on the blockchain being compromised by supply-chain vulnerabilities?

1.1.1 Examples of Key Attack Techniques

SUNBURST Malware

Certain attacks, such as the SUNBURST attack against the SolarWinds system, happen because malware was put into a vendor's trusted software; then an enemy may attack

all the vendor's client organizations at once. This was a supply chain attack that happened because of trust [42].

XcodeGhost

In 2015 a compiler for Apple Xcode appeared that seemed more easily available in Asia. It was compromised with malware that would inject malware into the output binary. It was named XcodeGhost and was a malware Xcode compiler for Apple macOS. In the incident, a compiler attack referred to as Xcodeghost (2015) constituted a malware Xcode compiler for Apple macOS that can inject malware into output binary. n September 2015 a new compiler malware attack was discovered in China. This attack targeted Apple's Xcode development environment, the official development environment for iOS and OSX development. The attack was a malicious compiler that the user could download. As users in China had slow download speeds when downloading large files from Apple's servers, many users would instead download Xcode from other colleagues or from Baidu Wangpan, a Chinese cloud service created by Baidu. The malicious compiler is believed to have been spread initially through Baidu Wangpan [27]. This malicious version of Xcode replaces CoreServices object files with malicious object files. These object files are used to compile many iOS and OSX applications. It is unknown to the author if the attack did anything to OSX applications. Nevertheless, the attack infected many iOS applications. These applications were then spread through the Apple App Store to the end-users, while neither the users of the applications nor the developers of the applications knew of this. Pangu Team claims the attack infected at least 3418 different iOS applications [25]. Amongst the apps infected were WeChat version 6.2.5, a very popular instant messaging application [22]. In September 2015 WeChat had 570 million active users daily [26]. As iOS had a mobile phone market share of roughly 25At the same time, it is to be expected that the virus can have reached many millions of users [23]. 15 The infected iOS applications will gather system data, encrypt it and send it to a remote server using HTTP [27]. The application also contains the ability to attempt to trick the user into giving away their iCloud password through a crafted dialogue box. Further, the attacker can read and write to and from the clipboard. It can also craft and open malicious URLs, this can also be used for malicious behavior through crafting specific URLs that open other apps with security weaknesses [43]. The attack seems to have spread mostly in China, as the applications infected were mostly Chinese-developed

applications targeting the Chinese market [18]. Nevertheless, applications such as WeChat have been popular in larger regions of eastern Asia. The malware can therefore also spread to larger regions [22]. Pangu Team also released an application to detect malicious applications created through XcodeGhost infected compilers [25].

Win32/Induc.A

Another malware targeting Delphi compilers was called Win32/Induc.A

A relatively recent third incident of this type has been the Win32/Induc. A that was targeting Delphi compilers. W32/Induc is a self-replicating virus that works similarly to the compiler trap door attack. The compiler trap door attack will be further explained in Chapter 3. The virus inserts itself into the Delphi source 13 libraries upon execution, infecting the compiler toolchain. It then inserts itself into all produced executables from the infected toolchain. The virus targets Delphi installations running on a Windows platform. The virus has come in three known variants named Induc-A, Induc-B and Induc-C [39]. It is believed that the two initial runs were testing versions to test the insertion of the virus building up to the release of the more malicious Induc-C virus. Upon executing an infected file, the virus will check for the existence and location of a Delphi installation [39]. Early versions of the virus looked for Delphi installations by looking for a specific registry subkey. Later versions will instead search the hard drive for a compatible Delphi installation. Once the installation is found, the virus will create a backup of the original SysConst.dcu (for earlier versions of the virus) or SysInit.dcu (for later versions of the virus) used for all produced executables [39, 44]. After this, it will copy the SysConst.pas or SysInit.pas file from the object library, modify the source code to include the malicious behavior, and compile the file. It will then be inserted so that it is used instead of the original SysConst.dcu or SysInit.dcu. At this point, the Delphi compiler is infected, and all produced executables from the compiler will also include the virus. Induc-C also can infect any .exe files on the computer [39]. This greatly increases the virus' ability to spread to other computers. The initial versions of the virus (Induc-A and Induc-B) seem not to include any malicious behavior other than self-reproduction [39]. In contrast, Induc-C includes behavior where it downloads and runs other malware. It does this by downloading specific JPEG files containing encrypted URLs in the EXIF sections. It will then download and execute the malware at these locations. Amongst known malware executed is a password stealer. It is also reported that Induc-C

includes behavior that can be used for botnets. The known defense against the attack is antivirus software, which can detect infected executables or infected object files [44]. As of September 2011, over 25were recorded in Russia [39]. For Induc-C most of the detected infections occurred in Russia and Slovakia. 14 This attack is similar to the compiler trap door attack, as they both attack compilers and include self-replicating behavior. The main difference between this and the compiler trap door attack is that instead of attaching the virus to the compiler executable, it inserts itself into object files used for the compilation of all programs using the infected toolchain. The compiler executable itself does not get infected unless it is produced using this toolchain. As the malware isn't specifically attached to the compiler executable, it can be easily delivered through any infected executable and will then further spread itself to all compiled executables. It is reasonable to believe that this the method will resort to a virus that spreads itself faster to more computers, however, it might also be easier to detect

ProFTP Login Backdoor

In 2010 there was an injection of a login backdoor in the software named ProFTP. This was a supply chain attack that will probably be able to automatically detected in the not-so-distant future, while the attack was not merely in the binary code but also in the source code. What happened was that a malware hacker got access to the repository and made a commit that included a backdoor to the next version of the ProFTP software.

A malware can encrypt itself, spread, and replicate as a virus on computers. Packages used in programming languages have recently been high-profile targets for supply chain attacks. A popular package from the Ruby programming language used by web developers was compromised in 2019 when it started to include a snippet of code allowing remote code execution on the developer's machine [60]. What is interesting about this case is that the authors' credentials were compromised, and the malicious code was never found in the code repository. It was only available from the downloaded version installed by the Ruby package manager. https://snyk.io/blog/malicious-remote-code-execution-backdoor-discovered-in-the-popular-bootstrap-sass-ruby-gem/

launchpadhit

In another recent attack, there was a Trojan horse attack on cryptocurrency usage in a codebase that was compromised from a GitHub repository [51].

Cyberattacks against Bitcoin

The three works that are often being cited about Bitcoin are first two mainly technical texts by two different research groups [7] [36]. The third is an article written more from the economical perspective [5] In the context of Bitcoin, there have been a few notable cases of supply chain attacks. The first case is the Bitcoin-Core project, where a malicious developer uploaded a version of the Bitcoin-Core software that included a backdoor. This backdoor allowed the developer to steal funds from any Bitcoin address. The developer did this by changing the code that generated Bitcoin addresses. The second case is the Bitcoin-Core project, where a malicious developer uploaded a version of the Bitcoin-Core software that included a backdoor. This backdoor allowed the developer to steal funds from any Bitcoin address. The developer did this by changing the code that generated Bitcoin addresses. The third case is that of the Bitcoin.org website, where a malicious attacker was able to insert code that would redirect users to a phishing website. The attacker did this by compromising the server that hosted the website. In all of these cases, the attackers could compromise the systems because they had control of the supply chain. In the first two cases, the attackers were able to upload malicious code to the Bitcoin-Core project's code repository. In the third case, the attacker was able to compromise the server that hosted the Bitcoin.org website. In all of these cases, the attackers could compromise the systems because they had control of the supply chain. In the first two cases, the attackers were able to upload malicious code to the Bitcoin-Core project's code repository. In the third case, the attacker was able to compromise the server that hosted the Bitcoin.org website. These cases show it can attack the Bitcoin network by compromising the supply chain. In all of these cases, the attackers could compromise the systems because they had control of the supply chain. In the first two cases, the attackers were able to upload malicious code to the Bitcoin-Core project's code repository. In the third case, the attacker was able to compromise the server that hosted the Bitcoin.org website. All of these cases show that it is possible to attack the Bitcoin network by compromising the supply chain.

1.2 Contributors

In Chapters 2 and forward the state of the art research is presented as well as novel approaches for an implementation of the attack and the defense. Exclude the first chapter and references as well as appendix.