DELIVER UNCOMPROMISED: SECURING CRITICAL SOFTWARE SUPPLY CHAINS

PROPOSAL TO ESTABLISH AN END-TO-END FRAMEWORK FOR SOFTWARE SUPPLY CHAIN INTEGRITY

by Charles Clancy, Joseph Ferraro, Robert Martin, Adam Pennington, Christopher Sledjeski, and Craig Wiener

 #21-0278 3-9-2021

Executive Summary

A series of actions, if taken by the software development community and the larger information technology ecosystem, can significantly reduce the risk of compromise, exploitation, exfiltration, or sabotage from software supply chain attacks.

While no silver bullet exists, establishing and implementing an end-to-end framework for software supply chain integrity will reduce risks from too-big-to-fail applications that are central to private sector enterprises, governments, and the critical capabilities they rely upon each day.[1]

The current state of practice in software supply chain security lacks systematic integrity. There are insufficient interoperable tools for preventing, detecting, or remediating software supply chain attacks that go beyond tools available for general cybersecurity threats. Given the potential impacts from software supply chain attacks, we cannot treat them as just another cybersecurity breach.

Within this paper we propose the following framework be developed to bolster the integrity of our software supply chains:

- The software industry must adopt a standard scalable, interoperable Software Bill of Materials (SBOM)-based supply chain metadata approach that can track composition and provenance of every component in a software product, provide metadata integrity for each software component and its pedigree, and use that metadata to systematically characterize and manage risk.
- Cryptographic code signing and associated validation infrastructure needs to mature to reflect the complexity and diversity of today's software supply chains, and prepare for the rapid deployment of expected new standards for post-quantum digital signatures.
- Systems involved in building and distributing software and software updates, at a minimum, must meet higher levels of assurance, such as National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53 Rev 5.[2]

NIST should update their existing supply chain standard, NIST SP 800-161,[3] to include this framework.

THE CURRENT STATE OF PRACTICE IN SOFTWARE SUPPLY CHAIN SECURITY LACKS SYSTEMATIC INTEGRITY.

The United States (U.S.) federal government should require this framework be implemented by vendors, second- or third-party resellers, and integrators as it acquires services and supplies, and use this framework as part of selecting appropriately trustworthy suppliers, supplies, and services.[4] For example, the Department of Defense's Cybersecurity Maturity Model Certification (CMMC)[5] program should include use of this framework as part of its criteria.

Longer-term, industry standards such as the International Organization for Standardization (ISO) 27001[6] should be updated to include this framework.

Contents

Introduction

In 2017, the United States (U.S.) Office of the Director of National Intelligence (ODNI) released a short paper depicting the vast threat from software supply chain attacks.[7] A software supply chain attack is defined as the compromise of software code through cyberattacks, insider threats, or other close access activities at any phase of the supply chain to infect an unsuspecting customer.[8] ODNI recognized that:

> "Hackers are circumventing traditional cyber defenses to compromise software and delivery processes to enable successful, rewarding and stealthy methods to subvert large numbers of computers through a single attack. Cyber experts predicted the use of this attack vector because (1) many software development and distribution channels lack proper cyber and process protections, and (2) other cyberattack paths become less optimal as system owners improve the overall cybersecurity posture of their networks, components and computers. Adversaries can use these generalized attacks to target specific victims to conduct extortion campaigns or exfiltrate, manipulate or destroy data for some targeted, deliberate purpose."[9]

Software supply chain attacks can be relatively simple or complex. For example, a simple mode of attack is conducted by corrupting a vendor's patch site by placing malware files similarly named to authorized code, in the hopes that the malware file is downloaded (e.g., ACME.xxx vs ACMEupdate.xxx). A more complicated or complex attack would typically include a foreign intelligence or military intelligence service infiltrating a software company's code base to insert malware before the code is compiled or electronically signed.[10]

The software supply chain compromise of the SolarWinds Orion Platform that occurred in 2020, a nearly ubiquitous product used for Information Technology (IT) infrastructure management, revealed to the public at large the power and potentiality of this technical approach against networked systems for espionage, sabotage, and warfighting. Based on the depth, breadth, and scope of this subversion, it is clear that a foreign intelligence service compromised a large array of government agencies, critical infrastructure entities, and private sector organizations at least as early as March 2020, most likely through an insertion of malicious code into the foundational source code library at SolarWinds.

Preparatory compromises at SolarWinds date back to October 2019, according to at least one recent report.[11, 12]

Although the subversion of the SolarWinds enterprise-level software appears to have a variety of parallels to earlier supply-chain attacks, and in some cases leveraged strikingly similar Tactics, Techniques, and Procedures (TTPs), the main distinguishing feature of this operation appears to be the scope, scale, and mission of the approximately 18,000 organizations targeted and impacted by the

A SOFTWARE SUPPLY CHAIN ATTACK IS DEFINED AS THE COMPROMISE OF SOFTWARE CODE THROUGH CYBERATTACKS, INSIDER THREATS, OR OTHER CLOSE ACCESS ACTIVITIES AT ANY PHASE OF THE SUPPLY CHAIN TO INFECT AN UNSUSPECTING CUSTOMER.

campaign. The importance of these entities to the core functioning of government and industry IT systems in the U.S. and other Western countries—the main targets for a foreign intelligence service—whether for straightforward espionage purposes or a prelude to something even more sinister cannot be overstated. Clearly, many Fortune 500 companies and government organizations embraced the functionality of the SolarWinds software for their discrete enterprises for a variety of very good reasons. However, the software supply chain attack against the Orion platform illuminates the potential vulnerabilities of a wide variety of ubiquitous software applications. For example, software that supports critical cloud service infrastructures, if successfully subverted, would potentially create even farther reaching and wider spread disruptions for economic functioning and national security matters. Unfortunately, this is only one example.[13] While the full impact of the SolarWinds breach is not yet fully understood, the security implications of this particular software supply chain attack vector are clear and require software developers and distributors to begin the process of systemic redressal.

A Brief Overview of Previous Software Supply Chain Attacks

Well-documented software supply chain subversions continue to proliferate unabated. In 2017 alone, there were at least seven major software supply chain attacks discovered or announced. Below are some historical examples through 2020,[14] concluding with the SolarWinds breach.

Havex

At least as far back as 2014, Russian state-sponsored advanced persistent threat (APT) actors trojanized update installers on a minimum of three industrial control systems (ICS) vendor web sites to advance Havex malware into ICS. Havex is a Remote Access Trojan (RAT) that uses a Command and Control (C&C) server to deliver additional payloads to compromised systems. According to the Department of Homeland Security (DHS), the basic Havex payload gathered information on Class Identification (CLSID), server name, Program ID, Open Platform Communications (OPC) version, vendor information, running state, group count, and server bandwidth. OPC is widely used in industrial process control, manufacturing automation, and other applications.[15] Though Havex was not observed altering ICS system parameters, ICS-CERT testing revealed that in addition to gathering ICS system information, the Havex payload could cause multiple common OPC platforms to intermittently crash, possibly resulting in a denial-of-service condition on ICS networks dependent on OPC communications. Havex actors utilized a combination of techniques for initial access including watering hole-style attacks and phishing emails.[16]

Kingslayer

Announced publicly in 2017, but occurring in 2015, Chinese APT-19 cyber actors[17] targeted system administrator accounts to steal credentials to replace legitimate application updates with a malware version containing an embedded backdoor. Specifically, for at least two weeks, the actors compromised the website and update server of a company that sells software to help Windows system administrators review and interpret Windows event logs. The actor-controlled website hosted subverted, but signed, versions of the application service executable, and an installer package file that contained the trojan. Once installed, the software would attempt to load secondary malicious payloads.[18, 19]

According to RSA, the actors specifically targeted Windows operating system administrators of large organizations. The victims included: five major defense contractors; four major telecommunications providers; more than ten western military organizations; more than two dozen Fortune 500 companies; 24 banks and financial institutions; and at least 45 higher educational institutions.[20]

CCleaner

In 2017, Cisco Talos detected Chinese APT[21] cyber actors using download servers intended to distribute CCleaner, a legitimate software package, to deliver malware to over 2.27 million endpoint users for over a month.[22] The legitimately signed version of CCleaner 5.33 distributed by Avast contained a multi-stage

malware payload with a Domain Generation Algorithm (DGA) as well as hardcoded C&C.[23] The Chinese APT actors likely compromised the development or build environment. Compromised versions returned the computer's name, Internet Protocol (IP) address, a list of installed software, a list of active software, and list of network adapters to a C&C server. Based on this information, a secondary customized malware payload was downloaded to the infected machine. A third stage backdoor associated with Chinese APT 17, dubbed ShadowPad, was used to capture keystrokes, credentials, and remotely control infected computers. CCleaner is legitimately used to perform system maintenance including temporary file clean up, performance optimization, and centralized application management. In 2016, CCleaner was downloaded 2 billion times.

NetSarang

For 17 days in August 2017, Chinese APT 17 actors embedded ShadowPad malware in the source code of a Windows server management product used by hundreds of organizations, including banks and energy companies, under NetSarang Computer's Xmanager Enterprise 5.0 Build 1232, Xmanager 5.0 Build 1045, Xshell 5.0 Build 1322, Xftp 5.0 Build 1218, and Xlpd 5.0 Build 1220.[24] The backdoor was placed in a version of the file nssock2.dll and was signed with the private key from NetSarang utilizing a legitimate certificate. Several layers of encrypted malicious code were only decrypted and activated when the C&C server sent the compromised machine a special packet. Until the activation packet was received, compromised machines sent only basic configuration information, every eight hours. Like the CCleaner compromise, the ShadowPad backdoor was used to capture keystrokes, credentials, and remotely control infected computers leading to the theft of information from hundreds of companies in the energy, financial services, manufacturing, pharmaceuticals, telecommunications, and transportation industries.[25, 26]

ASUS

In 2019, Chinese cyber actors compromised and accessed the ASUS update infrastructure and infected over a million users to advance targeted malicious updates to specific computers/users of interest according to open-source reports. This highly targeted espionage campaign delivered additional malware payloads to 583 specific computers identified through the target computer's media access control (MAC) address, a unique identifier assigned to a network interface controller (NIC) for use as a network address for communications within a network segment. When the malware found a target address of interest, it reached out to a C&C server that installed additional malware. Asus is a Taiwanese original equipment manufacturer (OEM) and computer, phone, hardware, and electronics company. As of 2020, Asus is the world's sixth-largest personal computer (PC) vendor by unit sales.[27, 28, 29, 30, 31]

SolarWinds Breach, December 2020

According to publicly available reports, the subversion of the SolarWinds Orion platform began as early as October 2019, with the first set of malicious code introduced in March 2020, through at least June 2020 in a variety of updates released by the unwitting vendor.[32, 33]

The initial enabling action was likely the threat actor's compromise of an account in the software development environment of SolarWinds or a

compromise of the build environment itself, although this is currently under further forensic investigation.[34, 35] This type of compromise would allow the actors to review the Orion platform code, design, and architecture. According to recently published reports, the actors likely inserted a few lines into a dynamic-link library (DLL) file to provide an entry point in any enterprise that subsequently downloaded and ran this file through the SolarWinds update process. Inserting malicious logic in the development stage allows clandestine modifications to be signed and secured giving the subverted code the appearance of legitimate SolarWinds software.[36]

The integrity of the software was most likely violated by compromising a legitimate developer's account, thus making the malicious actor's changes appear as though they came from the developer. This leads to questions regarding the lack of active monitoring in the development environment and traceability of changes made to the SolarWinds source code. The most likely answer is that the typically manual and idiosyncratic nature of software development allowed this code alteration to pass unnoticed. Based on observed tradecraft, it seems that the attacker was unsure if this modification would be detected and engaged in a test run by inserting empty classes to the software code to determine if they were noticed prior to injecting the actual subversions into the development environment codebase.[37]

More recent analysis provides insights into how the actors leveraged the update process, using an iterative approach to identify infected targets worthy of additional exploitation efforts.[38] The actors began by infecting the SolarWinds build server with Sunspot malware. The build server, also known as a continuous integration server, is used to test and integrate smaller portions of a software application into larger applications.[39] According to researchers, from the vantage point of the build server, Sunspot then awaited build commands. When build commands were issued, Sunspot replaced source code files within Orion with Sunburst malware. This malicious code was eventually downloaded as part of a legitimate software update. Sunburst then collected information from victim networks for the hackers to evaluate and prioritize additional exploitation activities.[38]

Common Adversary Tactics, Techniques, and Procedures and Target Opportunities

Adversary Tactics, Techniques, and Procedures

While each of the supply chain attacks exemplified above have unique implementations and details, the following are highlights of adversary TTPs that appear to be common across this array of examples. Abuse of trust is a core underpinning principle.[40]

Epic Scale, Focused Targeting

The adversaries behind recent software supply chain attacks conducted very thoughtful and deliberate targeting of commercial solutions and likely chose specific software packages based on functionality and lists of customers that are reliant on these products. Once identified, the potential scale of compromise given market penetration into key segments represents a fundamental enabler that is then used to target specific entities of high interest. Fortune 500 and critical government organizations were targets in many cases.

Capitalizing on Trust

Since software updates have prima facie legitimacy and are assumed to be safe and trustworthy, customers install updates without question as they have been conditioned to assume that the risk of doing so is low to non-existent based on known and previously dependable sources.[41] Furthermore, these types of enterprise-wide updates are propagated using accounts that operate with significant levels of privilege, typically via administrator accounts. Well placed software supply chain attacks allow an adversary to install malicious capabilities and conduct operations with an almost unrestricted level of access to achieve a cascading array of objectives.

Built From Scratch

Insertions of malicious code into the development chain or immediately after the completion of the development chain, but before software signing, is a highly effective strategy to subvert the legitimate software production process. This facilitates the propagation of the subversion through traditional, and trusted, software supply chain distribution pathways.

DevSecOps

Modern supply chains leverage DevSecOps environments and practices to accelerate the development and deployment of new capabilities that are more operationally relevant. However, without additional security measures, this practice also allows the adversary to inject unintended code into the trusted baseline. Additionally, a DevSecOps developmental approach provides software updates that are more frequent in nature and designed to provide incremental capabilities that require more communication with customer enterprise networks. If not understood and managed correctly, this rapid rate of software updating and deployment allows attackers increased access to target networks, potentially increasing risk.

Current Approaches to Detecting Supply Chain Injections

As shown in the last section, a broad set of organizations were victims of multiple successful software supply chain compromises of increasing complexity over the last several years, from a variety of suspected nation state intelligence services. The resulting intrusions involved all aspects of modern IT and Operational Technology (OT) enterprises (from

on-premises IT software, to ICS/SCADA networks, to cloud computing environments and managed service providers) and continue unabated to this day. In this section, we provide a brief overview of the current approach to detecting supply chain injections, which is necessary but insufficient to comprehensively defend against this type of cyber espionage method.

Integrity Checking Mechanisms Are Important...

In line with current industry best practices, MITRE ATT&CK's®[42] entry on software supply chain injections[43] recommends verifying compiled code binaries against known good hashes, or other integrity checking mechanisms. The two primary current approaches to achieve these verifications are the distribution of a software's cryptographic hash[44, 45] through independent trusted mechanisms, and code signing. In both cases, software binaries and/or distribution packages are input into a cryptographic hash function as part of a software release process. With independent distribution of the hash value itself, software recipients can verify that their software matches a published value. As a result of code signing, a hash value is created by using the signer's private key and the integrity can be verified by a third party utilizing a common root of trust.

But Insufficient to Stop SolarWinds Style Supply Chain Attacks

In many previous widely known compromises of the software supply chain, the distributed software did not match any integrity checking mechanism created by the developer and would have been revealed via either of these practices. However, it appears that neither of these practices would have been successful against the recent SolarWinds breach. Current practices rely upon verification that a distributed piece of software matches what a developer created and signed, but SolarWinds believes that their build process itself was modified, which allowed the adversary to insert malicious code undetected.[46] Another technique increasingly leveraged by adversaries is the theft of code signing private keys,[47] allowing them to create malicious software that passes verification checks.

Identifying Adversary Tactics, Techniques, and Procedures During Exploitation Operations Offers Some Early Detection Opportunities

Defenders still have other potential opportunities for detecting a breach[48] when current strategies against supply chain injection itself fail. While precise details of how the SolarWinds Orion build environment was modified are currently unknown, considerably more is known about the breaches that resulted from the malicious code. Software supply chain injections can provide an adversary initial access to an enterprise, but usually just represent a beachhead and do not accomplish adversary end goals, such as espionage, sabotage, or destruction by themselves. To achieve their end goals, an adversary frequently will have to remotely perform several additional actions to gain access to their final target after the malware executes. For example, reporting[49] on the breach resulting from the trojanized SolarWinds product has described at least 45 MITRE ATT&CK techniques[50] that were leveraged by the malware and the exploiter in early stages of the breach.[51] In addition to gaining initial access to the target environment via their software supply chain compromise, the SolarWinds exploitation team also surveyed the environment and took steps to ensure their persistence. While current practices for detecting the supply chain injection itself were likely inadequate, several of these 45 techniques represent additional opportunities for detection of the resulting

breach. Since currently published reporting only describes a small portion of this intrusion, it is likely that these techniques and associated opportunities for detection will only increase as additional information becomes available.

These types of detection opportunities are common across other software supply chain attack incidents. Supply chain injection itself is a relatively rare method, but after successfully implementing this approach, adversaries leveraged common intrusion tactics and many additional ATT&CK techniques. Our analysis of reporting on the Havex, Kingslayer, CCleaner, Netsarang, and ASUS supply chain injections identified at least 45 ATT&CK techniques performed on victim systems in addition to the initial access via "Supply Chain Compromise: Compromise Software Supply Chain." A little more than half of these 45 ATT&CK techniques overlapped with the techniques seen in reporting related to the SolarWinds breach, as referenced above. With properly tuned data source collection, behavioral analytics, and alerting, it is possible to detect and defend against an advanced APT adversary early on in a breach.

Despite the best efforts to ensure software integrity through a variety of means and utilize TTP detection methodologies, software supply chain attacks like CCleaner and SolarWinds have continued to succeed. A comprehensive framework that is focused on robust software supply chain integrity throughout the design, build, and delivery process is clearly needed today.

Proposed End-To-End Framework for Software Supply Chain Integrity

Within this section we detail our proposed end-to-end framework for software supply chain integrity, see Figure 1. This framework relies on (1) managing risk through standardized Software Bill of Materials (SBOM)-based supply chain metadata, (2) improving code and component signing infrastructure, and (3) hardening the software build and distribution infrastructure. Widespread adoption and implementation of these imperative solutions will dramatically reduce the risks and associated impacts of software supply chain attacks depicted in the prior sections.

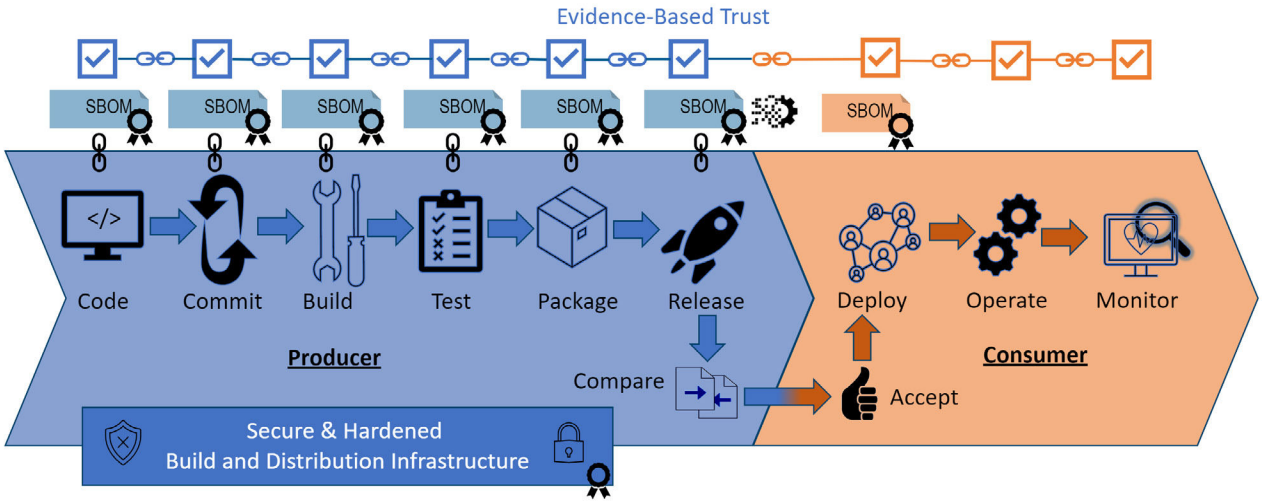

FIGURE 1. A FRAMEWORK FOR SOFTWARE SUPPLY CHAIN INTEGRITY[52]

Verifiable Composition and Process Integrity through Evidence-based Metadata Software Releases Using Standardized SBOM-based Supply Chain Metadata

In today's software producer-consumer culture, an individual or enterprise end user detrimentally relies on the integrity and functionality of the software deployed within their enterprise. An end user should have appropriate, measurable, and verifiable insight into the composition and critical attributes of the software or software as a service they are purchasing and insight into critical risks posed by that software when deployed on their system. Metadata information attesting to the composition, provenance, and integrity of software produced to the right of the Build step, as shown in Figure 1, can be used for verification prior to deployment in an operational environment. This cryptographically traceable metadata therefore becomes the basis of measuring trustworthiness, in accordance with organizational tolerances and policies, to make a determination if the software meets acceptable risk thresholds for the target network. This insight, based at least in part on producer attestable standards-based evidence, could support

an automatable decision to deploy, remove, or mitigate concerns through complimentary controls.

The Tool-to-Tool Software Bill of Materials (3T-SBOM), Grafeas, and "in-toto" projects are collaborative initiatives that will provide software supply chain metadata and integrity checks (articulated with evidence) to build and deliver software in a trustworthy manner despite threats illuminated by the SolarWinds compromise.

Today, tracking and reviewing software is a manual and labor-intensive endeavor. Therefore, espionage campaigns targeting software supply chains can subvert the efforts of a large team of developers. The software development and orchestration process require better recordkeeping, including the processes and sources of code used in the creation of products. The use and adoption of standard supply chain metadata that captures details such as the creation and creator (author & timestamps), the tools used and their options when creating the code (pedigree), the source of the code and any third-party components (provenance), as well as the integrity of the ensemble (secured identity and integrity) would make malicious modifications that circumvent the code creation processes much more difficult to achieve.

The establishment and broad adoption of standard supply chain metadata will allow software ecosystems to inoculate themselves from the type of subversion demonstrated against SolarWinds and other variations of similar supply chain attacks. This approach is a part of the Department of Commerce's National Telecommunications and Information Administration (NTIA) Software Component Transparency Initiative's Framing Group's approach for "High Assurance" capabilities of an SBOM. NTIA has adopted an approach that builds on existing data formats and standards that are already being deployed in different aspects of the software supply chain, from the open source community to modern software development organizations to the medical device community.[53] This approach explicitly acknowledges that the diversity of the software world means that a single exogenous solution is unlikely to succeed, and emphasizes modularity and integration with other existing solutions and tools. The community has identified formats with existing user bases, including SWID tags, SPDX, and the OWASP-related CycloneDX.[54] By layering high assurance details on top of this, organizations can follow an incremental, evolutionary approach. This effort has broad international cross-sector support building on stakeholder consensus, and the active engagement of several parts of the U.S. government.

THE SOFTWARE DEVELOPMENT AND ORCHESTRATION PROCESS REQUIRE BETTER RECORDKEEPING, INCLUDING THE PROCESSES AND SOURCES OF CODE USED IN THE CREATION OF PRODUCTS.

The 3T-SBOM Exchange standards joint working group et al. is nearing completion of such a standard. However, the pervasive use of supply chain metadata needs to be coupled with understanding and management of the integrity of the software development and supply chain flow within and between organizations.

To adequately address the type of malicious code insertion discovered in the SolarWinds breach, there needs to be "assurance to downstream consumers," (see Figure 1). This assurance needs to show that any given piece of software has "completed expected/required steps, and that no unexpected and potentially malicious steps have been inserted into the documented supply chain sequence."[55] Integrity of the software supply chain development sequence is one of nine usage scenarios driving the creation of the 3T-SBOM standard.[56]

Grafeas ("scribe" in Greek) is one example of an open-source artifact metadata application programming interface (API) that provides a uniform way to audit and govern software supply chains. The Digital Bill of Material (DBOM) effort is another.[57] As the 3T-SBOM standard is being established, these projects are moving to include 3T-SBOM standards as one of the metadata types they capture and convey.

Furthermore, to provide for the integrity of a release, "in-toto"[58] offers a framework that captures additional evidence and checkable policy metadata. "In-toto" is designed to ensure the integrity of a software product from initiation to end-user installation. It does so by making it transparent to the user what steps were performed, by whom, and in what order and will utilize the 3T-SBOM's pedigree and provenance capturing capabilities. As a result, individuals or teams creating software can share the specific processes, activities, and participants in creating and releasing software in a way that, utilizing "in-toto," allows the user to verify what steps in the supply chain were performed, and that the steps were performed by the right actor.[59]

The 3T-SBOM effort and the broader NTIA community are working closely with members of DBOM, Grafeas, "in-toto," and others, to be able to create a common supply chain metadata standard and SBOM vision that each can use to align efforts and build products with a common integration underpinning. Thus, the Grafeas, DBOM, and "in-toto" efforts will leverage the SBOM format to capture and convey the metadata within a software supply chain. Accelerating the convergence and maturity of these complementary efforts will address a massive capability gap that is sorely needed to achieve an overall articulated framework.

These evidence-based characterizations, when anchored in strong roots of trust, provide a powerful solution to produce, measure, and make risk-informed decisions about the many attributes of a software supply chain from a producer to an end consumer. Additionally, this evidence-based approach allows third-party verification to be conducted in a highly automated manner, provides an evidence chain that can be used to demonstrate and measure trust, pedigree, and provenance. Based on this approach, a SBOM provides irrefutable forensic data to identify the origin of malicious behavior targeting the supply chain. As more embrace and move toward a DevSecOps model, there will need to be native support and use of SBOMs in associated tool pipelines. Infrastructure as Code[60] (IaC) is a critical enabler to achieving this goal and would allow others to leverage and recreate results with high confidence for any part of the development chain. Finally, the approach above can be combined with The Internet Engineering Task Force Software Updates for Internet of Things Working Group (IETF SUIT WG)[61] to ensure that additional dependencies beyond the core software to be installed provide the appropriate level of verifiable pedigree as well.

Maturing Code Signing for Software Integrity, Including Quantum Readiness

Use of signatures to cryptographically assure the authenticity of software continues to evolve. Early approaches focused on the risk of accidental corruption and provided a file hash that could be validated after software had been downloaded. In 2005, Microsoft launched what at the time was a revolutionary tool to combat the growing corpus of exploits: Windows Update. Windows Update facilitated the launch of a broader code signing ecosystem, particularly for things like device drivers. The mobile ecosystem further pushed code signing into the application domain to combat malicious apps.

The 2016 attack on Internet infrastructure company Dyn by the Mirai botnet[62] highlighted the need to fundamentally reform security on embedded devices that were proliferating as part of the broader Internet of Things (IoT) movement. Since Mirai, the sector experienced a huge push to adopt secure code update infrastructure, primarily for IoT firmware and software. Building on standards like RFC 4108,[63] standards bodies are off and running to secure IoT devices through efforts like Institute of Electrical and Electronics Engineers (IEEE) 802.1AR,[64] IETF SUIT, and IETF Remote Attestation Procedures (RATS).[65] While embedded devices were the target in the Mirai event, what occurred in 2016 is representative of a larger issue that holds modern software and firmware at risk.

A comprehensive code signing approach is needed to account for the complexity of current software supply chains. For example, third-party modules integrated into software packages often have their own signatures, but those signatures are lost when the software package is bundled and resigned. This metadata and cryptographically robust provenance must be retained.

A COMPREHENSIVE CODE SIGNING APPROACH IS NEEDED TO ACCOUNT FOR THE COMPLEXITY OF CURRENT SOFTWARE SUPPLY CHAINS.

As part of the 3T-SBOM Integrity working group effort, guidance is being created with respect to how cryptographically strong identity and integrity mechanisms can be used with a standard SBOM to convey authoring identity and assure the integrity of the SBOM itself within and across the various communities of software creators and distributors. This work is leveraging efforts by the IETF and "in-toto" and is focused on the two widely used cryptographic signing standards: PKIX (RFC 5280) and PGP (RFC 4880). A premise for 3T-SBOM's integrity is to enable signature providers to use keys and signatures that are pre-existing in these two ecosystems.

While adopting these robust approaches to integrity through public key infrastructure (PKI) are urgently needed, a looming concern for the entire cryptographic community is the risk that quantum computing will reach a level of maturity that can undermine public key encryption. The broader IT, software, cryptography, security, and standards community needs to begin planning for a shift from current ciphers to Post-Quantum Cryptography (PQC) ciphers that are immune to being broken by quantum computing algorithms. This demands that code signing infrastructure should be rolled out to be quantum ready.

National Institute of Standards and Technology (NIST) is currently evaluating candidate ciphers.[66] Once new ciphers are selected, a whole range of standards must be updated to accommodate these changes and

require underlying cryptographic software libraries to be updated, tested, and released. Next, networking libraries need to be updated to incorporate these changes, and applications that use those libraries need to require them.

Additionally, the broader PKI ecosystem that protects web-based transactions needs new quantum-safe trust anchors. Millions and millions of digital certificates must then be replaced with new quantum-safe certificates. Trust anchors are often baked into software distributions, particularly for embedded devices, and this can only be accomplished through software or firmware updates.

The migration to PQC represents an upcoming disruptive event for not only our software supply chains, but also the delivery mechanisms that secure that software supply chain. With many emerging technologies such as 5G[67] shifting to PKIs for their security, the need to prepare for this is even more urgent.

As an industry we should plan now to shift as early as possible to post-quantum digital signatures for software and software update integrity. NIST is in the late stages of selecting one or more post-quantum digital signature algorithms for general use and standardization,[68] and is currently in the process of approving Stateful Hash-Based Signature Schemes.[69] The IETF has published two standards:

1. **RFC 8391** – Extended Merkle Signature Scheme (XMSS)[70]
2. **RFC 8554** – Leighton-Micali Hash-Based Signatures[71]

The use of stateful hash-based signature schemes involves a risk that an error in tracking the state of the digital signing key could lead to an attacker being able to forge signatures and additional strategies may be needed to keep these high value stateful keys protected. Additionally, traditional software interfaces for signature creation and key handling will likely not work out of the box for existing systems and will need to be modernized.

The fact that NIST is in the process of approving these algorithms despite the articulated risk, even for use cases in which deployment cannot wait for the general-purpose process to complete, shows that NIST expects substantial interest in accelerated transition to post-quantum digital signature algorithms.

Stakeholders should immediately begin working together to prepare for rapid deployment of general purpose post-quantum techniques for code signing once the NIST process results in at least one final standard. Such preparation should include (1) establishment of a post-quantum certification authority to anchor trust, (2) planning modernization of client-side software update mechanisms for operating systems and applications to incorporate one or more post-quantum digital signature validation methods, and (3) planning modernization of software update distribution platforms to incorporate post-quantum digital signing capabilities. Early stakeholders should include, but are certainly not limited to, Microsoft, Apple, Google, Linux Foundation, and the Free Software Foundation.

Software Build and Distribution Infrastructure Hardening

Hardening of build environments has not been uniformly addressed across the industry. As evidenced by a variety of software supply chain attack methods, many developer systems have prioritized ease of configuration and use over security. Developer

systems represent highly lucrative targets to nation state intelligence services engaged in cyber espionage via software supply chain attacks. These development systems are integral to providing critical inputs to the capability baseline, often allow developers to operate at a high level of privileged access as well have incredibly broad and far-reaching access to code and resources. These attributes make these systems and users supremely attractive—unfortunately, they are often times improperly secured and defended.

Existing standards, such as the recently updated NIST Special Publication (SP) 800-53 (Rev. 5),[72] should be more rigorously applied as a critical security protection baseline for ensuring that the build and distribution systems are adequately protected. Government and private acquirers should be deploying software that has been developed to a Moderate-Impact or High-Impact standard[73] depending on the criticality of the infrastructure and work it supports to achieve mission or business needs. While producers of software have migrated to a DevSecOps software development approach and are much more security conscious than ever before, there is still a tremendous need to incorporate controls within NIST SP 800-53 (Rev. 5) for both the development pipeline infrastructure and processes. Producers of software all need to ensure the code output itself is compliant with these controls. Additionally, maturity models such as CMMI for Development,[74] a part of CMMI v2, are still being synchronized with evolving development practices. Furthermore, while highly applicable, standards such as International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) 27001:2013,[75] need to be updated to included enhanced controls as captured in NIST SP 800-53 (Rev. 5).[76] Both NIST SP 800-53 (Rev. 5) and NIST SP 800-161 should be updated to reflect the recommendations within this framework as well.

The following seven items represent opportunities to apply and advance the hardening of build and distribution environments, leveraging NIST SP 800-53 (Rev. 5) controls as well as foundational Cyber Resiliency Engineering Framework techniques.[77]

1. **Follow Best Practices** – Establish different roles and separate accounts, define the workflow, align account privileges, align permissions to separate functions, and configure systems to only provide functionality needed for the respective accounts and their roles.[78]
2. **Criticality Analysis** – Rigorously identify Crown Jewels[79] within the development, build, and distribution infrastructure to develop a priority-based strategy for protecting, monitoring, verifying, and restoring critical system components that could hold the system and outputs at risk.
3. **Continuous Red Teaming** – Conduct continuous penetration testing and red teaming of development and build environments to validate that configuration settings, security controls, and mitigating functions

WHILE PRODUCERS OF SOFTWARE HAVE MIGRATED TO A DEVSECOPS SOFTWARE DEVELOPMENT APPROACH AND ARE MUCH MORE SECURITY CONSCIOUS THAN EVER BEFORE, THERE IS STILL A TREMENDOUS NEED TO INCORPORATE CONTROLS WITHIN NIST SP 800- 53.

are having the intended effect on a highly iterative basis. The catalyst for this testing should be both time-based and event-driven based on system change. This technique needs to be employed both on the output of the system and the system itself. A highly automatable approach with measurable, reproducible, and auditable results to attest to the environment is the driver.

4. **Segmentation and Micro Segmentation** – Implement a segmentation architecture approach to protect the build and distribution infrastructure to ensure the protection and efficient auditing of resources. For greater control, micro segmentation can be used to protect at the application level.[80]
5. **Advance the Creation and Fidelity of Supply Chain ATT&CK-focused TTPs** – Mature the current supply chain related TTPs for detection to be as mature as other attack chain areas. Focusing on supply chain centric TTPs will allow the broader community to more comprehensively sense, detect, and share threat data to address this style of attack more rapidly.
6. **Sensoring and Analytical Monitoring** – Correlate development and build pipelines as well as distribution infrastructure to detect adversary behaviors earlier and throughout the supply chain. A robust and detailed criticality analysis can identify additional critical monitoring points within the system to focus the application of additional defensive measures. Adversary ATT&CK TTPs can provide a systematic approach to monitoring for adversary patterns of behavior. Where TTPs may not be fully documented, the Common Attack Pattern Enumeration and Classification (CAPEC™)[81] can be used to help prioritize monitoring of critical IT infrastructure.
7. **Heuristic Analysis** – Employ a role-based access approach for source code that commits to the version control repository correlated to trouble ticket or development assignments. This method would illuminate the potential insertion of unintended code into the software baseline to be flagged for additional review and verification.

Conclusions

As we have shown, establishing and implementing an end-to-end framework for software supply chain integrity as part of an overall supply chain security strategy will reduce risks from too-big-to-fail applications that are central to private sector enterprises, governments, and the critical capabilities they rely upon each day. This end-to-end framework must include the rapid adoption of an implementable standard for a software bill of materials, a cryptographic code signing approach that is quantum ready, and ensuring systems involved in building, distributing, and updating software are hardened to higher levels of cybersecurity assurance as called for in NIST SP 800-53 Rev 5. Without these changes, massive deficits across the industry today will continue to persist.

The community must accelerate the maturation and formalization of standard supply chain metadata and their underpinning standards with the intent to rapidly adopt this practice, which will allow practitioners to measurably track and attest for the composition, provenance, and integrity metadata for every component in a piece of software, to include the supporting infrastructure. As more embrace and move toward a DevSecOps model, there will need to be native support and use of supply chain metadata in their supporting tool pipelines. This will provide acquirers and end users long sought-after transparency in the content of the software they use, which will result in informed risk decisions pertaining to deployment, updating, and disposition of their entire application library. These attestable artifacts can also be verified by third parties such as National Information Assurance Partnership (NIAP) to further increase confidence.

Cryptographic code signing and associated validation infrastructures need to mature to reflect the complexity and diversity of today's software supply chains and initiate the rapid universal deployment of post-quantum digital signatures upon final selection and standardization by NIST. Modernized cryptographic signing not only will advance integrity capabilities for current systems but will also be a critical component of signing and attesting to SBOM evidence-based artifacts.

IMPLEMENTING THIS END-TO-END APPROACH WILL REQUIRE THE SOFTWARE COMMUNITY, PRIVATE SECTOR, AND GOVERNMENTS TO EXPEDITIOUSLY ADDRESS THE VULNERABILITIES WITHIN THE OVERALL SOFTWARE ECOSYSTEM AS SOON AS POSSIBLE.

Last, but certainly not least, further hardening of software build and distribution infrastructure is both critical and fundamental to providing integrity to software. Without the steps we have outlined to achieve this hardening, all other recommendations will be less likely to succeed. While implementing SBOM and cryptographic code signing is necessary, software cybersecurity assurance will remain insufficient without a secure platform that can be better defended by implementing controls delineated in NIST SP 800-53 v5 and NIST SP 800-161.[82]

Maturity Models such as the Cybersecurity Maturity Model Certification (CMMC)[83] can and should be used as an approach to demonstrate measurable advancement in an organization's security practices. Implementing this end-to-end approach will require the software community, private sector, and governments to expeditiously address the vulnerabilities within the overall software ecosystem as soon as possible. This is especially critical as it pertains to software that runs on systems that are so widely used or central to the functioning for business or national security concerns that its failure would be disastrous to the functioning of society, or to state it more simply, are "too big to fail."

Abbreviations and Acronyms

TERM	DEFINITION
3T-SBOM	Tool-to-Tool Software Bill of Materials
API	Application Programming Interface
APT	Advanced Persistent Threat
C&C	Command and Control
CAPEC	Common Attack Pattern Enumeration and Classification
CLSID	Class Identification
CMMC	Cybersecurity Maturity Model Certification
DBOM	Digital Bill of Materials
DGA	Domain Generation Algorithm
DHS	Department of Homeland Security
DLL	Dynamic Link Library
IaC	Infrastructure as Code
ICS	Industrial Control Systems
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
MAC	Media Access Control
NIAP	National Information Assurance Partnership
NIC	Network Interface Controller
NIST	National Institute of Standards and Technology
NTIA	National Telecommunications and Information Administration
ODNI	Office of the Director of National Intelligence
OEM	Original Equipment Manufacturer
OPC	Open Platform Communications
OT	Operational Technology

PC	Personal Computer
PKI	Public Key Infrastructure
PQC	Post Quantum Cryptography
RAT	Remote Access Trojan
RATS	Remote Attestation Procedures
SBOM	Software Bill of Materials
SP	Special Publication
SUIT	Software Updates for Internet of Things
TTP	Tactics Techniques and Procedures
U.S.	United States
WG	Working Group
XMSS	Extended Merkle Signature Scheme

Endnotes

1. It is acknowledged that there are software risks beyond the scope of this paper to be addressed to include the design, architecting, and production of robust software that is an output of the supply chain under scrutiny.
2. National Institute of Standards and Technology (NIST), "Security and Privacy Controls for Federal Information Systems and Organizations," SP 800-53, Revision 5, September 2020. [Online]. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf. [Accessed January 26, 2021].
3. National Institute of Standards and Technology (NIST), "Supply Chain Risk Management Practices for Federal Information Systems and Organizations," SP 800-161, April 2015. [Online]. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161.pdf. [Accessed January 26, 2021].
4. The MITRE Corp., "Trusting Our Supply Chains: A Comprehensive Data Driven Approach," January 2021. [Online]. https://www.mitre.org/publications/technical-papers/trusting-our-supply-chains-a-comprehensive-data-driven-approach. [Accessed January 26, 2021].
5. Cybersecurity Maturity Model Certification (CMMC), Version 1.02, March 18, 2020. [Online]. https://www.acq.osd.mil/cmmc/docs/CMMC_ModelMain_V1.02_20200318.pdf. [Accessed January 26, 2021].
6. International Standards Organization (ISO), "Information technology—Security techniques—Information security management systems—Requirements," ISO/IEC 27001:2013, October 2013. [Online]. https://www.iso.org/standard/54534.html [Accessed January 26, 2021].
7. Office of the Director of National Intelligence (ODNI) "Software Supply Chain Attack Graphic" DNI.GOV Website. [Online]. https://www.dni.gov/files/NCSC/documents/supplychain/20190327-Software-Supply-Chain-Attacks02.pdf. [Accessed December 18, 2020].
8. ibid
9. ibid
10. ibid
11. U.S. Department of Homeland Security (DHS), Cyber Security and Infrastructure Security Agency (CISA), Alert (AA20-352A) "Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations," December 17, 2020. [Online]. https://us-cert.cisa.gov/ncas/alerts/aa20-352a. [Accessed December 18, 2020].
12. Peričin, Tomislav "SunBurst: the next level of stealth: SolarWinds compromise exploited through sophistication and patience," December 16, 2020. [Online]. https://blog.reversinglabs.com/blog/sunburst-the-next-level-of-stealth. [Accessed December 18, 2020].
13. Recent reports indicate investigators are examining whether unit testing and code quality analysis software—that underpins the development of software code itself—may be an even broader and more lucrative target of subversion. See: N. Perlroth, D. E. Sanger, and J. E. Barnes "Widely Used Software Company May Be Entry Point for Huge U.S. Hacking." [Online]. https://www.nytimes.com/2021/01/06/us/politics/russia-cyber-hack.html. [Accessed January 7, 2021].
14. The history of software supply chain attacks goes back even further. For example, in 2000, it was reported that "unknown hackers were believed to have gained access to the source code" of Microsoft's "most valuable software, including latest versions of Windows and Office." CNN Money, "Microsoft: big hack attack," Oct. 27, 2000. [Online]. https://us-cert.cisa.gov/ics/alerts/ICS-ALERT-14-176-02A. [Accessed January 26, 2021]. This article reports that Microsoft found signs the intrusion "may have come from St. Petersburg in Russia."
15. Department of Homeland Security (DHS), US CERT, ICS Alert (ICS-ALERT-14-176-02A), ICS Focused Malware (Update A), June 27, 2014. [Online]. https://us-cert.cisa.gov/ics/advisories/ICSA-14-178-01. [Accessed December 18, 2020].

16. Department of Homeland Security (DHS), US CERT, ICS Advisory (ICSA-14-178-01), ICS Focused Malware, June 30, 2014. [Online]. https://us-cert.cisa.gov/ics/advisories/ICSA-14-178-01. [Accessed December 18, 2020]. .
17. The MITRE Corp., "Deep Panda," April 17, 2020. [Online]. https://attack.mitre.org/groups/G0009/. [Accessed December 18, 2020].
18. RSA, "Kingslayer: A Supply Chain Attack," (Whitepaper), February 2017. [Online]. https://www.rsa.com/en-us/offers/kingslayer-a-supply-chain-attack [Accessed December 18, 2020].
19. Krebs on Security, "How to Bury a Major Breach Notification," February 2017. [Online]. https://krebsonsecurity.com/2017/02/how-to-bury-a-major-breach-notification/. [Accessed December 18, 2020].
20. NCSC, "Software Supply Attacks," March 27, 2019. [Online]. https://www.odni.gov/files/NCSC/documents/supplychain/20190327-Software-Supply-Chain-Attacks02.pdf. [Accessed December 18, 2020].
21. Dark Reading, "Chinese APT Backdoor Found in CCleaner Supply Chain Attack," March 12, 2018. [Online]. https://www.darkreading.com/endpoint/privacy/chinese-apt-backdoor-found-in-ccleaner-supply-chain-attack/d/d-id/1331250. [Accessed December 18, 2020].
22. L. Tung, ZDNet, "Hackers hid malware in CCleaner PC tool for nearly a month," September 18, 2017. [Online]. https://www.zdnet.com/article/hackers-hid-malware-in-ccleaner-pc-tool-for-nearly-a-month/#:~:text=Updated%3A%202.27%20million%20users%20had,machines%2C%20CCleaner%20maker%20Piriform%20said.&text=Hackers%20modified%20versions%20of%20the,of%20PCs%20with%20a%20backdoor. [Accessed December 18, 2020].
23. CISCO Talos, "Ccleanup: A Vast Number of Machines at Risk," September 18, 2017. [Online]. https://blog.talosintelligence.com/2017/09/avast-distributes-malware.html. [Accessed December 18, 2020].
24. Dark Reading, "Chinese APT Backdoor Found in Ccleaner Supply Chain Attack," March 12, 2018. [Online]. https://www.darkreading.com/endpoint/privacy/chinese-apt-backdoor-found-in-ccleaner-supply-chain-attack/d/d-id/1331250. [Accessed December 18, 2020].
25. ARS TECHNICA, "Powerful backdoor found in software used by >100 banks and energy cos." August 15, 2017. [Online]. https://arstechnica.com/information-technology/2017/08/powerful-backdoor-found-in-software-used-by-100-banks-and-energy-cos/. [Accessed December 18, 2020].
26. J. Henderson, Supply Chain, "Cyber hackers targeting software supply chains, says report" August 1, 2018. [Online]. https://www.supplychaindigital.com/technology/cyber-hackers-targeting-software-supply-chains-says-report. [Accessed December 18, 2020].
27. Zdnet, "Researchers publish list of MAC addresses targeted in ASUS hack," March 29, 2019. [Online]. https://www.zdnet.com/article/researchers-publish-list-of-mac-addresses-targeted-in-asus-hack/. [Accessed December 18, 2020].
28. Vice, "Hackers Hijacked ASUS Software Updates to Install Backdoors on Thousands of Computers," [Online]. https://www.vice.com/en/article/pan9wn/hackers-hijacked-asus-software-updates-to-install-backdoors-on-thousands-of-computers. [Accessed December 18, 2020].
29. Wired, "Hack Brief: How to Check Your Computer for Asus Update Malware," March 25, 2019. [Online]. https://www.wired.com/story/asus-software-update-hack/. [Accessed December 18, 2020].
30. Bleeping Computer, "MAC Addresses Targeted by the ASUS Supply Chain Attack Now Available," March 29, 2020. [Online]. https://www.bleepingcomputer.com/news/security/mac-addresses-targeted-by-the-asus-supply-chain-attack-now-available/. [Accessed December 18, 2020].

31. Cyware, "Researchers publish the list of 583 MAC addresses impacted by recent ASUS hack," March 29, 2019. [Online]. https://cyware.com/news/researchers-publish-the-list-of-583-mac-addresses-impacted-by-recent-asus-hack-e8f88086. [Accessed December 18, 2020].

32. U.S. Department of Homeland Security (DHS), Cyber Security and Infrastructure Security Agency (CISA), "Active Exploitation of SolarWinds Software," December 13, 2020 | Last revised: December 14, 2020. [Online]. https://us-cert.cisa.gov/ncas/current-activity/2020/12/13/active-exploitation-solarwinds-software. [Accessed December 21, 2020].

33. U.S. Department of Homeland Security (DHS), Cyber Security and Infrastructure Security Agency (CISA), "Alert (AA20-352A) Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations," updated December 19, 2020. [Online]. https://us-cert.cisa.gov/ncas/alerts/aa20-352a [Accessed December 21, 2020].

34. ibid

35. Note: "CISA has evidence that there are initial access vectors other than the SolarWinds Orion platform. Specifically, we are investigating incidents in which activity indicating abuse of Security Assertion Markup Language (SAML) tokens consistent with this adversary's behavior is present, yet where impacted SolarWinds instances have not been identified. CISA is working to confirm initial access vectors and identify any changes to the TTPs. CISA will update this Alert as new information becomes available." U.S. Department of Homeland Security (DHS), Cyber Security and Infrastructure Security Agency (CISA), "Alert (AA20-352A) Advanced Persistent Threat Compromise of Government Agencies, Critical Infrastructure, and Private Sector Organizations," updated December 21, 2020. [Online]. https://us-cert.cisa.gov/ncas/alerts/aa20-352a [Accessed December 21, 2020]; Also see: N. Perlroth, D. E. Sanger, and J. E. Barnes "Widely Used Software Company May Be Entry Point for Huge U.S. Hacking" https://www.nytimes.com/2021/01/06/us/politics/russia-cyber-hack.html. [Accessed January 7, 2021].

36. Microsoft 365 Defender Research Team, "Analyzing Solorigate, the compromised DLL file that started a sophisticated cyberattack, and how Microsoft Defender helps protect customers," December 18, 2020. [Online]. https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defender-helps-protect/. [Accessed December 18, 2020].

37. SolarWinds, "SolarWinds Security Advisory," December 31, 2020. [Online]. https://www.solarwinds.com/securityadvisory. [Accessed January 7, 2021].

38. C. Cimpanu, ZDNet, "Third malware strain discovered in SolarWinds supply chain attack," January 12, 2021. [Online]. https://www.zdnet.com/article/third-malware-strain-discovered-in-solarwinds-supply-chain-attack/. [Accessed January 25, 2021].

39. Continuous Integration | IBM [Online] https://www.ibm.com/cloud/learn/continuous-integration. [Accessed January 26, 2021].

40. Atlantic Council, "Breaking trust: Shades of crisis across an insecure software supply chain," July 26, 2020. [Online]. https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/. [Accessed January 26, 2021].

41. In fact, not installing various software version changes or security updates is itself deemed a high-risk behavior that can lead to compromises.

42. MITRE ATT&CK is a globally-accessible knowledge base of adversary behaviors that is widely used by the public and private sector as a framework for delineating, tracking, and developing defenses against adversarial techniques. [Online]. https://attack.mitre.org/. [Accessed January 26, 2021].

43. MITRE ATT&CK, "Supply Chain Compromise: Compromise Software Supply Chain." [Online]. https://attack.mitre.org/techniques/T1195/002/. [Accessed January 26, 2021].

44. Cryptographic hashes are mathematical functions that map arbitrary amounts of data to a fixed-size value. Functions are intended to be one-way; it should be infeasible to create another set of data that maps to a given value. Handbook of Applied Cryptography, Chapter 9, "Hash Functions and Data Integrity." [Online]. http://cacr.uwaterloo.ca/hac/about/chap9.pdf. [Accessed January 26, 2021].

45. National Institute of Standards and Technology (NIST), "Hash Functions," June 22, 2020. [Online]. https://csrc.nist.gov/projects/hash-functions. [Accessed January 26, 2021].

46. On December 14, 2020, SolarWinds filed a Current Report on Form 8-K. [Online]. https://www.sec.gov/ix?doc=/Archives/edgar/data/1739942/000162828020017620/swi-20201217.htm. [Accessed January 26, 2021].

47. MITRE ATT&CK, "Subvert Trust Controls: Code Signing." [Online]. https://attack.mitre.org/techniques/T1553/002/. [Accessed January 26, 2021].

48. The MITRE Corp., "TTP-Based Hunting," July 2020. [Online]. https://www.mitre.org/publications/technical-papers/ttp-based-hunting. [Accessed January 26, 2021].

49. Center from Threat-Informed Defense, MITRE ATT&CK®, "Tracking UNC2452-Related Reporting." [Online]. https://github.com/center-for-threat-informed-defense/public-resources/blob/master/solorigate/README.md. [Accessed January 25, 2021].

50. Descriptions of specific adversary behaviors combined with defensive recommendations for each. The MITRE Corp., "MITRE ATT&CK®: Design and Philosophy," MP180360R1, Originally Published July 2018, Revised March 2020. [Online]. https://attack.mitre.org/docs/ATTACK_Design_and_Philosophy_March_2020.pdf. [Accessed January 26, 2021].

51. M. Malone, J. Williams, J. Burns, and A. Pennington, MITRE ATT&CK®, "Identifying UNC2452-Related Techniques for ATT&CK," December 22, 2020. [Online]. https://medium.com/mitre-attack/identifying-unc2452-related-techniques-9f7b6c7f3714. [Accessed January 25, 2021].

52. We do not specifically address the post acceptance deployment of software by consumers in this paper but have included it here for completeness.

53. National Telecommunications and Information Administration (NTIA) initiative of Software Component Transparency [Online] https://www.ntia.doc.gov/SoftwareTransparency and the published consensus documents at https://ntia.gov/SBOM. [Accessed January 26, 2021].

54. National Telecommunications and Information Administration (NTIA) "Survey of Existing Formats and Standards," (2019) https://www.ntia.gov/files/ntia/publications/ntia_sbom_formats_and_standards_whitepaper_-_version_20191025.pdf. [Accessed January 26, 2021].

55. CISQ/OMG Joint SBOM Working Group, "Standardizing SBOMs—post-Nashville – 09-29-19", pages 2-3. [Online]. https://drive.google.com/file/d/1lCWlv_5tXPeAJeuT5sAk6LIqUrmmucgN/view?usp=sharing. [Accessed January 26, 2021].

56. The Consortium for Information & Software Quality (CISQ)/Object Management Group (OMG) Joint SBOM Working Group effort. [Online]. https://drive.google.com/drive/folders/1Vg_OEUMPdh1jt0u-nIGbbvfgNdmu1tI0; [Accessed January 26, 2021]. Other elements of 3T-SBOM include pedigree and provenance capturing capabilities.

57. Control System Cyber Security Association International, "The Time Has Come to Automate Supply Chain Security," [Online]. https://www.cs2ai.org/post/the-time-has-come-to-automate-supply-chain-security. [Accessed January 26, 2021].

58. https://in-toto.io/ [Accessed January 26, 2021].

59. in-toto, "What is in-toto?" [Online]. https://in-toto.io/in-toto/. [Accessed December 26, 2020].

60. S. Guckenheimer, Microsoft, "What is Infrastructure as Code?" April 4, 2017. [Online]. https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code. [Accessed January 26, 2021].

61. Internet Engineering Task Force, "Software Updates for Internet of Things (SUIT)," Working Group, [Online]. https://datatracker.ietf.org/wg/suit/about/. [Accessed January 26, 2021].

62. Department of Energy Argonne National Laboratory Cyber Team "Mirai Attack on Dyn Internet Infrastructure" November 30, 2016. [Online]. https://coar.risc.anl.gov/mirai-attack-dyn-internet-infrastructure/. [Accessed January 7, 2021].

63. R. Housley, "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages," Internet Engineering Task Force (IETF) RFC 4108, Internet Standard, August 2005. [Online]. https://tools.ietf.org/html/rfc4108. [Accessed January 26, 2021].

64. Institute of Electrical and Electronics Engineers, "Secure Device Identity," IEEE Std 802.1AR-2018, February 2017. [Online]. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8423794. [Accessed January 26, 2021].

65. Internet Engineering Task Force, "Remote Attestation Procedures (RATS)," Working Group, [Online]. https://datatracker.ietf.org/wg/rats/about/. [Accessed January 26, 2021].

66. National Institute of Standards and Technology (NIST), NISTIR 8309, "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process," July 2020. [Online]. https://csrc.nist.gov/publications/detail/nistir/8309/final. [Accessed January 26, 2021].

67. C. Clancy, R. McGwier, and L. Chen, "Post-Quantum Cryptography and 5G Security: Tutorial," ACM Wireless Security Conference, May 2019. [Online]. https://www.nist.gov/publications/post-quantum-cryptography-and-5g-security-tutorial. [Accessed January 26, 2021].

68. National Institute of Standards and Technology (NIST), NISTIR 8309, "Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process," July 2020. [Online]. https://csrc.nist.gov/publications/detail/nistir/8309/final.[Accessed January 26, 2021].

69. National Institute of Standards and Technology (NIST), "Recommendation for Stateful Hash-Based Signature Schemes," SP 800-208, October 2020. [Online]. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-208.pdf. [Accessed January 26, 2021].

70. A. Huelsing, D. Butin, S. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme," Internet Research Task Force, RFC 8391, May 2018. [Online]. https://tools.ietf.org/html/rfc8391. [Accessed January 26, 2021].

71. D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-Micali Hash-Based Signatures," Internet Research Task Force, RFC 8554, April 2019. [Online]. https://tools.ietf.org/html/rfc8554. [Accessed January 26, 2021].

72. National Institute of Standards and Technology (NIST), "Security and Privacy Controls for Federal Information Systems and Organizations," SP 800-53, Revision 5, September 2020. [Online]. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r5.pdf. [Accessed January 26, 2021].

73. ibid

74. CMMI® Development, [Online]. https://cmmiinstitute.com/cmmi/dev. [Accessed January 26, 2021].

75. International Organization for Standardization (ISO), "Information technology—Security techniques—Information security management systems—Requirements," ISO/IEC 27001:2013, October 2013. [Online]. https://www.iso.org/standard/54534.html. [Accessed January 26, 2021].

76. National Institute of Standards and Technology (NIST), SP 800-53, "Revision 5 Control Mappings to ISO/IEC 27001," December 2020. [Online]. https://csrc.nist.gov/CSRC/media/Publications/sp/800-53/rev-5/final/documents/sp800-53r5-to-iso-27001-mapping.docx. [Accessed January 26, 2021].

77. D. Bodeau, R. Graubart, J. Picciotto, and R. McQuaid, "The Cyber Resiliency Engineering Framework," The MITRE Corp., January 2012. [Online]. https://www.mitre.org/publications/technical-papers/cyber-resiliency-engineering-framework. [Accessed January 26, 2021].

78. CSO, "5 steps to simple role-based access control [RBAC]," January 2, 2019. [Online]. https://www.csoonline.com/article/3060780/5-steps-to-simple-role-based-access-control.html. [Accessed January 26, 2021].

79. The MITRE Corp., "Crown Jewels Analysis." [Online]. https://www.mitre.org/publications/systems-engineering-guide/enterprise-engineering/systems-engineering-for-mission-assurance/crown-jewels-analysis. [Accessed January 26, 2021].

80. Cisco, "What Is Network Segmentation?" [Online]. https://www.cisco.com/c/en/us/products/security/what-is-network-segmentation.html. [Accessed January 2, 2021].

81. The MITRE Corp., Common Attack Pattern Enumeration and Classification (CAPEC™), [Online]. https://capec.mitre.org/index.html. [Accessed January 26, 2021].

82. National Institute of Standards and Technology (NIST), "Supply Chain Risk Management Practices for Federal Information Systems and Organizations," NIST SP 800-161, April 2015. [Online]. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-161.pdf. [Accessed January 26, 2021].

83. Cybersecurity Maturity Model Certification (CMMC), Version 1.02, March 18, 2020. [Online]. https://www.acq.osd.mil/cmmc/docs/CMMC_ModelMain_V1.02_20200318.pdf. [Accessed January 26, 2021].

Acknowledgements

The authors would like to thank our reviewers:

Robert Case, Technical Fellow, The MITRE Corporation

Russ Housley, Founder and CEO, Vigil Security, LLC

Moses Liskov, Infosec Scientist, The MITRE Corporation

Robert S. Metzger, Attorney (Rogers Joseph O'Donnell, PC), Co-Author, "Deliver Uncompromised"

Kay Williams, Principal Program Manager, Azure Office of the CTO, Microsoft

MITRE's Mission

MITRE's mission-driven teams are dedicated to solving problems for a safer world. Through our public-private partnerships and federally funded R&D centers, we work across government and in partnership with industry to tackle challenges to the safety, stability, and well-being of our nation.

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™