Atlantic Council

SCOWCROFT CENTER
FOR STRATEGY AND SECURITY

BREAKING TRUST: SHADES OF CRISIS ACROSS AN INSECURE SOFTWARE SUPPLY CHAIN

Software Supply Chain Attack (SSCA): A software supply chain attack occurs when an attacker accesses and edits software somewhere in the complex software development supply chain to compromise a target farther up the chain by inserting their own malicious code.

Software Supply Chain Vulnerability (SSCV): A software supply chain vulnerability is any software vulnerability that could evolve into a supply chain attack if exploited.

More on Attacks: Modern software products contain a vast number of dependencies on other code, so tracking down which vulnerabilities compromise which products is a non-trivial organizational and technical feat. SSCAs take advantage of established channels of system verification to gain privileged access to systems and to compromise large networks. They undermine foundational tenets of trust in software development. Frequent oversights and failures in patching processes mean that as software supply chains grow in complexity, vulnerabilities will affect more end products, and SSCAs will become an increasingly common attack method—they are on the rise already.

More on Vulns: Because of the multifaceted nature of SSCAs, SSCVs include a wide variety of critical vulnerabilities. SSCVs are limited to those that would enable a malicious code injection, excluding vulnerabilities that would simply augment the harmful capacity of an SSCA payload.

Data Points

- **Date**: Best estimated start date of the attack. When no start date is identifiable, discovery date is used instead.
- **Name**: Name(s) of the attack/incident. Multiple names are included where possible.
- **Attack/Disclosure**: Whether the entry is a verified attack or disclosed vulnerability. Some incidents are a little of both, and it's possible that some vulnerabilities have been taken advantage of without being discovered despite their disclosure.
- **Summary**: Brief summary of the entry, its impact, and technical details.
- **Article(s)**: Links to article(s) about the entry.
- **Affected Code**: What code was modified by attackers, or what code had a vulnerability in it.
- **Code Location/Owner**: Who owned that code, or, if open source, the repository name.
- **Downstream Target**: The end-target of the attack.
- **Affected Codebase**: Categories describing the codebase, product, or service modified by attackers.
    - **1st Party OS/ Applications**: Modified software native to the targeted system
    - **1st Party Firmware**: Modified firmware native to the targeted system
    - **3rd Party Application**: Legitimate application modified
    - **3rd Party Firmware**: 3rd party firmware modified
    - **OSS**: Open source software modified

    - **Attacker Application**: App made by attackers to appear legitimate contains malware
    - **Unknown/NA**: Malicious code's first insert location unknown or not applicable
- **Attack Vector**: How the attacker was able to edit the affected code without detection.
    - **Stolen/purchased certificate**: Legitimate code signing certificate acquired by attacker
    - **Pre-signature insertion**: Attacker modified code before it was signed by developer
    - **Default password exploit**: Default/hardcoded password accesses software
    - **Account access**: Attacker accessed an account with editing permissions
    - **Self-signed/Unsigned**: Attacker signed software, or no signature required
    - **Broken signature system**: Signature check implemented badly or forgeable without private key
    - **Unknown, other, or N/A**: Method unknown
- **Distribution Vector**: How the attacker was able to distribute the modified code.
    - **Typosquatting**: Attackers trick the victims into downloading malware with names resembling legitimate
      apps or packages, often separated by an easy typo or British/American difference
    - **Hijacked updates**: Attacker was able to compromise an update server or otherwise proliferate malware through an established update system
    - **Proprietary application store**: Proprietary application store hosts an infected app, e.g. Google Play Store or Apple's App Store, for consumer download
    - **3rd party application store**: Third party application store, e.g. Tencent MyApp, hosts infected app for consumer download
    - **Open source dependency**: Infected code hosted in an open source library or code repository, so it gets included in code dependent on it
    - **Worm component**: The code can spread itself without user interaction or attacker input
    - **Hardware component**: The code can spread via infected USB sticks or other hardware transfers
    - **Direct download**: The code is directly downloaded by the victim not from open source repository or update server (rare corner case)
    - **Phishing**: The code is directly downloaded by the victim through a phishing scheme other than typosquatting
    - **Development software**: Attackers compromised a development program, allowing them to infect all applications developed in the broken development versions
    - **Supply chain service provider**: Attackers inserted their code in software used by a supply-chain vendor who spread it for them, e.g. HVAC climate control services, firmware providers, server maintainers, etc., excluding previously mentioned vendors
    - **Unknown, other, or N/A**: Method unknown
- **Supply Chain Potential**: For SSCV's, how the vulnerability could have contributed to an SSCA. N/A for SSCAs.
    - **Credential theft**: Would have given attackers code-editing account credentials
    - **Certificate theft**: Would have allowed attackers to obtain and sign code with a legitimate code-signing certificate by obtaining a private key
    - **Cryptography Error**: The vulnerability would have allowed attackers to forge a code signature through the systems reliance on compromised cryptography algorithms or faulty implementation
    - **Firmware Editing**: Would have allowed attackers to insert malicious code into firmware, making it exceptionally hard to detect and delete
    - **Default password**: Would have allowed hackers to edit code they should have been unauthorized to access

    - **Code injection**: Would have allowed attackers to inject code into an interface that would have run or inserted it with strong permissions (like an SQL or XSS vulnerability)
    - **N/A**: For SSCAs
- **Impact**: What the final payload's effect on the downstream target was or could have been.
    - **Data extraction**: The attackers extracted victim data and sent it to a C&C server
    - **Physical systems**: The attackers damaged physical systems
    - **Backdoor access**: The attackers left an access point to a network
    - **Cryptominer**: The attackers installed cryptomining software
    - **Remote command execution/ download**: The attackers enable remote code execution through a C&C server.
    - **Adware**: The attackers left adware on the device, either profiting from inserting their own ads or simply increasing ad traffic on or from the device
    - **Payment diversion**: The attackers diverted payments from victims to themselves
    - **Establish botnet**: The attackers established a botnet targeting the victim
    - **Data damage**: The attackers encrypted, deleted, or hid system data from the victim
    - **Unknown**: It is not known what the malware did once delivered to the final target
- **Attacker Name**: Group name, if attributable. Often contains best guesses.
- **Attacker Type**: Type of group, if attributable. Often contains best guesses.
    - **State**: Any state sponsored or run group
    - **Criminal**: Any international or nonstate group, often involved for profit
    - **Other**: Any group not clearly state or criminal—e.g. ideological nonstate, researcher, etc.
    - **Unknown**: No public attribution in cited sources
    - **N/A**: For disclosures
- **Infectious Potential**: The degree of proximity a given piece of malware needs to spread to a target, from requiring physical access to a piece of hardware, all the way down to worm capabilities allowing it to proliferate through networks autonomously
    - 0: Physical access
    - 1: Product components
    - 2: Store download
    - 3: Network connection
    - 4: Worm
- **Depth of Impact**: The level of meaningful and harmful access a given piece of malware has in an infected machine, from an infected application all the way down to the capability to damage physical systems
    - 1: Application
    - 2: Update
    - 3: OS
    - 4: Firmware
    - 5: Physical system

Read the full report here.