sonatype

Understanding Open Source Risk

How open source risk impacts your SDLC

The risks associated with using open source software (OSS) are diverse, and not all are created equal.

An outdated library may cause compatibility issues that disrupt your development process. Legal complexities of OSS licensing can lead to compliance challenges that, if overlooked, could result in significant legal and financial repercussions. A security vulnerability in an OSS component can open the door to malware attacks.

Given this diversity, it's crucial to approach open source risk with a comprehensive strategy that addresses the full spectrum of potential threats.

Types of open source risk

While vulnerabilities and malware may seem to be similar or even interchangeable terms, they are different concepts.

Knowing their distinct characteristics is essential for building effective strategies to optimize your software supply chain and reduce your open source risk.

Vulnerability

A vulnerability is an unintentional weakness in software that can be exploited to gain unauthorized access to systems or networks. While they can result in damage or manipulation to resources or operations, vulnerabilities are not inherently malicious and may result from software development oversights.

When identified, they are assigned a Common Vulnerabilities and Exposures (CVE) number to help developers recognize and address these issues. In many cases, using a different version of the affected software component that has been patched can mitigate your open source risk.

Malware

In contrast, malware is software explicitly designed to inflict harm. Malware exists in many forms, each with specific harmful functionalities such as the following:

- Viruses attach to legitimate programs and replicate, spreading upon program execution.
- Worms replicate themselves to spread across computing systems.
- Trojans disguise themselves as legitimate software and subsequently perform harmful activities.
- Spyware secretly monitors and collects user information.
- Ransomware encrypts data, demanding a ransom for its release.
- Adware displays unwanted advertisements, often without the user's consent.
- Botnets act as networks of infected computers controlled remotely by threat actors.
- Rootkits conceal the presence of other malware by modifying operating systems.

Differences between vulnerabilities and malware

INTENT

- **Vulnerabilities:** Unintentional, arising from errors in the software development process.
- **Malware:** Deliberately created to deceive, exploit, or harm users.

DISCOVERY

- **Vulnerabilities:** Identified through security testing, code reviews, or reports, and can then be patched or mitigated.
- **Malware:** Disguised as legitimate software and discovered through analysis, reports, or scans.

IMPACT

- **Vulnerabilities:** Varies based on severity if exploited by a threat actor.
- **Malware:** Immediate and intentionally harmful actions often upon installation.

Understanding attack vectors

In order to mitigate your open source risk, you need to understand the role attack vectors play in software supply chain attacks. An attack vector is a method or pathway a threat actor uses to access and exploit vulnerabilities in a system, application, or network. A vector contains the techniques employed to execute an attack and deliver malware.

Below you will find examples of a few common attack vectors in modern software supply chain attacks.

Compromised accounts

Compromised accounts involve threat actors gaining unauthorized access to accounts, typically of maintainers of open source projects. Threat actors inject malware into legitimate components, which are then distributed to unsuspecting users. The trust placed in maintainers and their components is exploited, making this approach particularly damaging.

An example of this is if a threat actor took control of an account and attacked via a version of event-stream, an npm component that contained code designed to steal Bitcoin from users' wallets. This attack affected numerous applications and developers who relied on the component, underscoring the critical importance of securing open source maintainer accounts.

Dependency confusion

Dependency confusion is an attack vector that exploits the mechanisms repositories use to resolve software dependencies.

This attack sequence capitalizes on the ambiguity between private and public repository feeds:

1. Threat actors craft malware and upload it to public repositories under the same names as private components used within an organization.
2. By assigning a higher version number to the malware, threat actors exploit the repository's default behavior of preferring the highest version available.
3. This can lead an installer script, intending to pull a file from a private repository, to inadvertently download the malware from the public repository instead.

This not only compromises the software but also opens the door for further exploitation such as code execution and data theft.

One real-life example is when Sonatype detected a dependency confusion attempt targeting John Deere, where a threat actor uploaded a component named "johndeer" to the PyPI repository. Designed to mimic an update, the component aimed to be automatically downloaded by John Deere's systems due to its higher version number. Sonatype's automated tools identified and flagged this activity, preventing a potential security breach.

Repository hijacking

In repository hijacking, a threat actor gains control of a repository, often by exploiting inactive or abandoned open source repositories. Once in control, the threat actor can push malicious updates that may go undetected due to the established trust in the repository, making this a significant open source risk. These malicious updates, once published, distribute malware to unsuspecting users who download or update components from the hijacked repository.

An example of repository hijacking involved a popular npm component for generating QR codes. The component's original repository had been abandoned, making it vulnerable to takeover. A researcher claimed this inactive repository, gaining control to demonstrate a significant security vulnerability in npm. This ethical demonstration highlighted the ease with which threat actors could exploit similar vulnerabilities to distribute malware via widely used components.

Trojan

Designed to deceive users into installing them, Trojans may appear as innocuous files or updates and can be bundled with legitimate software. But once installed, they can steal sensitive data, create backdoors for remote system access, or download malware. Trojans typically operate stealthily without showing clear signs of infection, making them hard to detect and remove.

One example of a Trojan is when a threat actor uploaded over 450 malicious components to PyPI. Once installed, these components executed a Windows Trojan that fetched additional malware from a Dropbox URL, evading initial detection and deploying its payload.

Typosquatting

Typosquatting is an attack vector where threat actors create malware with names that closely resemble those of legitimate components. By exploiting common typing errors, these deceptively named components trick developers into downloading them instead of the intended components. This deceptive strategy can lead to unintended code execution and data theft as the malware is integrated into build systems.

A real-life example is when Sonatype detected a case of typosquatting on PyPI, where malware, misnamed to impersonate the widely used "Requests," tricked developers into downloading them. Once installed, these components released ransomware, jeopardizing developers' systems and risking extensive data theft and damage.

Common attack types

An attack type is the specific category or nature of a software supply chain attack, focusing on the impact or outcome of the attack rather than the method used to deliver it. While an attack vector embodies the "how" of an attack, an attack type details the "what" of an attack, describing the effects after execution.

Backdoor

A backdoor is an attack type that bypasses normal authentication to gain unauthorized access to systems, often installed by malware for continuous access.

A good example of a backdoor is from the 2020 SolarWinds attack that involved "Sunburst," a backdoor enabling extensive infiltration of government and corporate networks.

Code injection

Code injection is an attack that inserts malware into a program, enabling threat actors to execute arbitrary operations.

A real-life example of code injection was back in 2022. The Log4j component experienced the Log4Shell vulnerability that allowed remote code injection, affecting millions worldwide.

Credential exfiltration

Credential exfiltration is a method of unauthorized extraction of user credential information, leading to further system exploitation.

A good example of a credential exfiltration is the 2012 LinkedIn breach that involved the theft of over 6.5 million hashed passwords.

Crypto stealer or miner

A crypto-currency stealer or miner is malware that either steals cryptocurrency wallets or covertly mines cryptocurrency using infected systems.

An example is the “python-dateutils” package, which masqueraded as a legitimate Python library, targets Windows, Linux, and macOS systems to mine Monero cryptocurrency and steal AWS credentials. This package employs techniques like Base64 encoding and ROT13 cipher for obfuscation, and even exfiltrates sensitive data using a Discord webhook.

Data exfiltration

Data exfiltration is unauthorized data transfer from a system or network.The npm package “speedy-ts-compiler” served as a distraction technique for data exfiltration by pretending to install another empty package, while actually capturing the user’s IP address and system username through manipulated npm commands.

Denial of service (DoS)

Denial of service (DoS) attacks aim to overwhelm networks with illegitimate requests to make services unavailable.

CVE-2020-13935 is an example of a denial of service (DoS), and is a vulnerability in the Apache Tomcat Websocket component that allows attackers to trigger an infinite loop by sending payloads with invalid lengths. This flaw can severely disrupt services by consuming excessive CPU resources until the affected application becomes unresponsive.

Hijacker

Hijacker malware redirects user traffic or alters system settings to display unwanted advertisements.

An example of a hijacker attack is ChromeLoader malware, which is distributed through VHD files disguised as illegal game hacks for platforms like Nintendo and Steam, hijacked users’ browsers by installing malicious Chrome extensions. These extensions redirect users to advertisement websites and collect browsing data, including browser credentials and settings modifications.

Phishing

Phishing fraudulently attempts to obtain sensitive information by posing as a trustworthy entity.

In this example of a phishing campaign targeting Office 365 users, hackers employed a malicious proxy server to bypass multi-factor authentication (MFA), capturing users’ credentials and session cookies to gain unauthorized access to email accounts. This adversary-in-the-middle (AiTM) technique facilitated business email compromise (BEC) and payment fraud schemes, leveraging compromised accounts for financial gain.

Protestware

Protestware is software modified by developers to include elements of protest, often disrupting functionality or distributing messages to draw attention to social or ethical issues.

A good example of protestware is when the npm libraries "colors" and "faker" were intentionally sabotaged by their maintainer in protest, causing them to disrupt thousands of applications by entering infinite loops or deleting useful code. This act highlighted the ongoing issues of open source sustainability and the pressures on unpaid developers.

Ransomware

Ransomware is an attack type that encrypts a victim's files, blocking access until a ransom is paid, often spreading through phishing or security vulnerabilities.

A real-life example of ransomware can be found in the PyPI packages "requesys," "requesrs," and "requesr," identified as typosquats of the legitimate "requests" library. These packages contain ransomware that encrypts files on a user's computer. The ransomware does not demand payment. Instead, it uploads the decryption keys to a Discord server, making them available without cost, underlining a more experimental or educational intent by the author.

Strengthening your defense against open source risks

Understanding the distinctions between vulnerabilities, malware, and the various attack vectors and types that exploit them is crucial to secure your software supply chain and reduce your open source risk.

While vulnerabilities are unintentional weaknesses that can be mitigated with patches and updates, malware is intentionally harmful and requires swift, decisive action to prevent damage.

By familiarizing yourself with these key concepts and the specific attack vectors outlined in this guide, you can better defend against the growing open source risk landscape targeting modern software development. A proactive and informed approach is essential to protect your organization's software supply chain from evolving threats.

Headquarters
8161 Maple Lawn Blvd, Suite 250
Fulton, MD 20759
USA • 1.877.866.2836

European Office
168 Shoreditch High St, 5th Fl
London E1 6JE
United Kingdom

APAC Office
60 Martin Place, Level 1
Sydney 2000, NSW
Australia

Sonatype Inc.
www.sonatype.com