Open-Source Software Supply Chain Attacks: Workflow and Recent Cases*

Yejun Lee and Hyo Jin Jo

Yonsei University, Seoul, South Korea
exit3015@gmail.com, hyojin.jo@yonsei.ac.kr

Abstract

Open-source software (OSS) supports modern technology. Its widespread reuse and complex dependencies create broad supply chain risks. This paper synthesizes the structure and attack surface of OSS supply chains, identifying recurring attack patterns observed across the development lifecycle, informed by recent major incidents (e.g., SolarWinds, XZ Utils, Shai-Hulud). We specify threat origins and trace their spread across repositories, build pipelines, and distribution channels, presenting a concise taxonomy of attack vectors that goes beyond one-off case analyses.

Keywords: Open-Source Software, Supply Chain Security, Supply Chain Attacks

1 Introduction

Open-source software (OSS) is core infrastructure for today's technology ecosystem, and modern software development relies heavily on it. This reliance stems from the reuse of well-vetted components that accelerate development and improve quality, as well as the benefits of standardization, cost efficiency, transparency, and customization. Such reliance, however, makes the OSS supply chain a prime target for adversaries. Against this backdrop, complex OSS-based supply chains expose multiple entry points—code repositories, CI/CD pipelines, and package registries—through which adversaries can inject malicious code and propagate it downstream. In recent years, software supply chain attacks have surged, and they are now widely recognized as a major threat [**?**]. Real-world incidents illustrate the far-reaching impact of these risks: SolarWinds involved a compromised build system; XZ Utils exploited the development workflow; and Shai-Hulud—a self-replicating worm—drove the widespread distribution of malicious npm packages [**?**, **?**, **?**]. Accordingly, this paper systematically synthesizes recent trends and well-documented cases in OSS supply chain attacks. This review strengthens awareness of OSS supply chain security and provides insights to guide future research on detection and defense.

2 Supply Chain Attack

Software supply chain attacks exploit the trust embedded in software development, build, and distribution processes. Attackers insert malicious code into legitimate artifacts or tamper with distribution channels, enabling large-scale downstream propagation. Unlike traditional vulnerability exploitation, these attacks leverage the entire supply chain as an attack vector. As shown in Figure **??**, examining prominent attack instances further clarifies how adversaries target specific points within the software supply chain. Because these attacks compromise the foundational trust of the software ecosystem, their impact extends well beyond individual projects or

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. P-5, December 16-18, 2025, Sapporo, Japan.

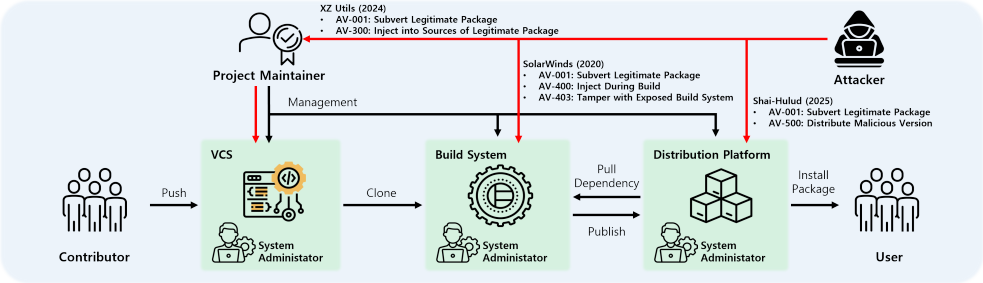


Figure 1: OSS supply-chain workflow with selected attack examples

organizations. These trends highlight the necessity of securing not only software products but also the broader ecosystem of dependencies, build systems, and distribution infrastructure.

SolarWinds (2020). Attackers compromised the SolarWinds Orion build system, inserting a backdoor into digitally signed updates. The tainted updates were widely distributed to government agencies and enterprises, propagating the compromise downstream. The incident demonstrated the system-wide impact of build-environment trust failures [**?**].

XZ Utils (2024). Attackers subverted the development and distribution workflow of XZ Utils, a widely used Linux compression library. They abused compromised maintainer privileges to inject a backdoor into official release packages. The case highlights how maintainer trust and routine workflows can themselves become attack vectors [**?**].

Shai-Hulud (2025). This self-replicating worm infected hundreds of npm packages by exploiting compromised developer accounts to embed malicious code in postinstall scripts. It then self-propagated using stolen credentials. The emergence of such supply-chain worms underscores automation-driven propagation risks across dependency graphs [**?**].

3 Conclusion

This paper presents a systematic characterization of the OSS supply chain's structure and attack surface and, through analysis of recent incidents and trends, demonstrates their systemic risk. It strengthens developer awareness of supply chain security and provides guidance for future research on detection and defense.

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2025-24534502).

References

[1] Matias Martinez Piergiorgio Ladisa, Henrik Plate and Olivier Barais. Sok: Taxonomy of attacks on open-source software supply chains. In *IEEE Symposium on Security and Privacy (SP)*, pages 1509–1526, 2023.

[2] Piotr Przymus and Thomas Durieux. Wolves in the repository: A software engineering analysis of the xz utils supply chain attack. In *IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*, pages 91–102, 2025.

[3] Surya Rao Rayarao and Naga Donikena. The shai-hulud npm supply chain attack: A comprehensive analysis of self-replicating malware in the javascript ecosystem. Preprint, 2025.

Open-Source Software Supply Chain Attacks: Workflow and Recent Cases

Yejun Lee[1] and Hyo Jin Jo[1]
[1]Yonsei University, Seoul, South Korea

MobiSec 2025

Introduction

- Open-source software drives modern development.
- However, heavy dependency increases supply-chain risks.
- Attackers target repositories, builds, and registries.
- Incidents like SolarWinds, XZ Utils, and Shai-Hulud show the scale.
- We analyze their workflows and shared weaknesses.

Supply Chain Attack

- Supply-chain attacks exploit trusted development stages.
- Malware spreads through legitimate software updates.
- Targets include source, build, and distribution stages.
- Tracing the attack flow reveals core weak points.
- The figure below outlines these critical paths.

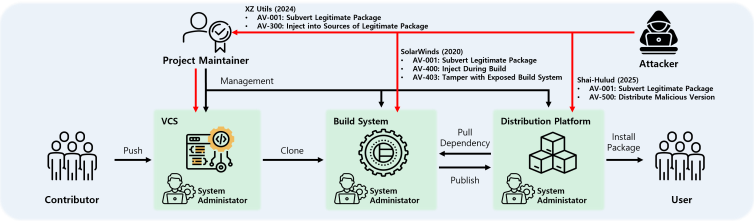


Figure. Open-Source Software Supply Chain Workflow

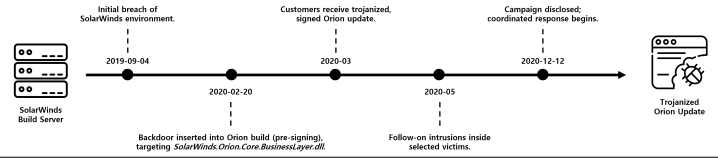
Conclusion

- OSS supply chains face trust-based vulnerabilities.
- Mapping cases to the SoK taxonomy reveals patterns.
- We identify recurring threats across all stages.
- Future defense must secure builds and maintainers.

Contributions: lifecycle SoK map, prerequisites & vector checklist, n=3 snapshot.
Defense: provenance & MFA · SLSA L3+ · signed artifacts & verification policy.

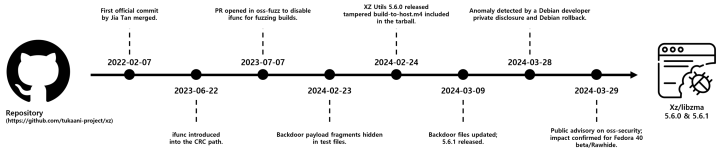
SolarWinds (2020)

- Build systems were compromised to inject code.
- Signed updates distributed malware to thousands.
- The case exposed CI/CD trust vulnerabilities.
- It remains a defining build-stage attack example.



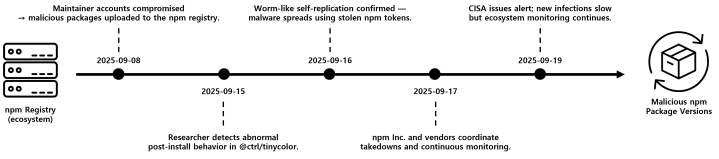
XZ-Utils (2024)

- Attackers gained maintainer roles over time.
- A backdoor was added to official releases.
- Normal workflows became infection channels.
- It highlights maintainer verification as essential.



Shai-Hulud (2025)

- Stolen maintainer accounts spread infected packages.
- Malicious post-install scripts propagated automatically.
- The worm moved through dependency chains.
- It shows a new self-replicating OSS attack form.



Recent Open-Source Software Supply-Chain Attacks Mapped to SoK Taxonomy*

Case	Year	Key prerequisite	Primary attack vector	Detail
SolarWinds	2020	AV-700: Compromise of Build System or CI/CD Infrastructure	1) AV-001: Subvert Legitimate Package 2) AV-400: Inject During Build 3) AV-403: Tamper with Exposed Build System	1) Tampered the build system to inject malicious code into artifacts 2) Distributed widely via signed official updates
XZ Utils	2024	AV-800: Role Assignment or Privilege Elevation of Malicious Actor	1) AV-001: Subvert Legitimate Package 2) AV-300: Inject into Sources of Legitimate Package	1) Committed a backdoor using granted maintainer privileges 2) Propagated to users through official releases/distribution channels
Shai-Hulud	2025	AV-600: Credential or Account Compromise of Maintainers	1) AV-001: Subvert Legitimate Package 2) AV-500: Distribute Malicious Version	1) Used compromised maintainer accounts to add malicious post-install code to existing npm packages 2) Stole credentials and self-propagated across packages

* Reference: Ladisa et al., IEEE S&P 2023 — "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains."