

Final Assignment 7.1: Supply Chain Attack Investigation

Joshua A. Wortz

University of San Diego

CSOL-570

Prof. Melanie Thomas

10/18/2025

Executive Summary

In late 2024, 1inch, a leading decentralized exchange (DEX), became one of several organizations indirectly impacted by a sophisticated supply chain attack on the open-source Lottie Player web component. The attacker did not directly penetrate 1inch's internal systems; instead, they exploited the trust relationships inherent in the software supply chain. By compromising the Lottie Player library hosted on a content delivery network (CDN), the threat actor injected malicious JavaScript code that propagated to multiple cryptocurrency-related websites, including 1inch, through legitimate software updates.

The malicious script created a pop-up wallet drainer, tricking users into signing transactions that transferred funds to the attacker's wallet. This led to the theft of approximately 10 wrapped Bitcoin (wBTC) on the Avalanche blockchain from impacted users (Mitrade, 2024). This event underscores the operational and reputational risks associated with unvetted or unmonitored dependencies on third-party open-source libraries.

This report provides an overview of the attack, an internal investigation summary, analysis of organizational gaps, remediation recommendations, a recovery plan, and post-mortem lessons to help prevent similar incidents in the future. The findings reveal critical opportunities for 1inch to strengthen supply chain visibility, improve detection of third-party code tampering, and implement governance practices aligned with modern DevSecOps principles.

Internal Investigation Summary

Threat Actor and Targeting Rationale

While no specific nation-state or criminal group has claimed responsibility, threat intelligence suggests that the attacker's primary motive was financial theft from users of cryptocurrency services. Lottie Player was selected because it is widely used across Web3 applications, particularly for animated wallet interfaces and onboarding flows. Its popularity made it an ideal target for watering hole-style attacks, in which adversaries compromise a shared dependency to reach many downstream victims.

Open-source components like Lottie Player are attractive targets because they often lack centralized governance, code signing, or controlled distribution channels. Unlike proprietary software vendors, which may have dedicated security teams, open-source projects depend on community vigilance. This decentralized model creates an asymmetry of effort: a single attacker compromising one widely used library can impact thousands of downstream applications, while defenders must secure each implementation individually. This dynamic aligns closely with trends identified in the Open Source Security Foundation (OpenSSF) reports, which highlight software supply chain attacks as one of the most rapidly growing categories of cyber incidents globally.

In this case, the malicious actor's campaign displayed hallmarks of strategic timing and operational stealth. The injection occurred shortly before a widely anticipated Lottie Player update, increasing the likelihood that developers and CDNs would cache and distribute the compromised version without scrutiny. This reflects a level of planning beyond casual opportunism and suggests the actor may have been monitoring the project's commit history or release cycles to identify optimal injection points.

This attack mirrors techniques documented in prior software supply chain breaches, such as EventStream (2018) and SolarWinds (2020). Instead of breaching each target individually, the actor compromised a single high-leverage point (the library) to silently affect dozens of organizations simultaneously.

Evidence and Actions of the Breach

According to public reporting and lab analysis, the attacker inserted malicious JavaScript into the Lottie Player package hosted on a CDN. When a user visited a website that embedded the compromised script, it executed code that displayed a phishing pop-up designed to look like a legitimate wallet interaction. Unsuspecting users signed transactions that transferred tokens to the attacker's wallet (TryHackMe, n.d.).

Key technical indicators:

- **Malicious script injection** in the minified JavaScript bundle (hosted on cdn.jsdelivr.net).
- **Popup payload** activated on pages containing Web3 wallet interactions.
- **Exfiltration address** identified on the Avalanche blockchain.
- Impact on websites including **1inch**, **Mover**, **Bubble**, and **TryHackMe** (Mitrade, 2024).

Further investigation revealed that the attacker leveraged obfuscation techniques to conceal the malicious pop-up code within legitimate animation functions. Static code analysis alone would not have easily flagged the payload; its behavior was triggered only under specific runtime conditions involving wallet interaction scripts. This indicates that the attacker had a clear understanding of how cryptocurrency platforms, including 1inch, integrate Lottie Player into their web onboarding and wallet signing flows.

This mirrors tactics seen in Magecart-style attacks, where malicious JavaScript is injected into checkout or wallet pages to harvest sensitive data at the point of user interaction. In both cases, the attacker avoids noisy network intrusion methods and instead compromises the trust boundary between the user and the application interface.

During internal log review, 1inch's team discovered that content security policies (CSP) allowed execution from the compromised CDN. No alarms were triggered by endpoint protection or web application firewalls, since the malicious code was delivered as part of legitimate application content.

Observations and Organizational Gaps

The incident exposed several gaps in 1inch's internal security posture regarding third-party dependencies:

1. **Lack of integrity checks:** 1inch did not implement **Subresource Integrity (SRI)** hashes for CDN-loaded JavaScript. This allowed the modified code to execute without detection.
2. **Insufficient supply chain visibility:** There was no automated inventory or Software Bill of Materials (SBOM) to track which libraries were in use and their versions.
3. **Limited runtime monitoring:** Web application security tools focused primarily on inbound traffic and did not monitor changes in client-side script behavior.
4. **No pre-deployment code scanning for external libraries:** Security scans were performed on internal codebases but not on third-party resources.

Detection feasibility:

Had integrity checks or runtime behavioral monitoring been in place, the injection could have been detected quickly. Web integrity platforms or modern RASP (Runtime Application Self-Protection) solutions may have flagged unexpected script changes.

CVE Reporting and Business Impact

The malicious modification qualifies as a supply chain compromise rather than a traditional software vulnerability. However, depending on disclosure guidelines, it could result in a Common Vulnerabilities and Exposures (CVE) entry for the affected Lottie Player version.

Reporting this could have dual implications for 1inch:

- **Positive:** Demonstrates transparency, contributes to community awareness, and accelerates ecosystem remediation.
- **Negative:** Could lead to regulatory scrutiny and potential loss of user confidence, especially given the financial impact on users.

Remediation Suggestions

To address these gaps, the following measures are recommended:

1. **Implement Subresource Integrity (SRI)** on all externally loaded scripts to ensure only approved code executes.
2. **Adopt a software composition analysis (SCA)** tool and **generate SBOMs** to track third-party dependencies.
3. **Integrate real-time script monitoring** through Content Security Policy reporting endpoints and RASP solutions.
4. **Require cryptographic signing** or host critical libraries internally to reduce reliance on external CDNs.

5. **Enhance developer security training** to include secure handling of open-source dependencies.
6. **Establish formal supply chain security governance**, including regular reviews of dependencies and patching cadences.

These remediation steps align with the principles outlined in the NIST Secure Software Development Framework (SSDF), which emphasizes integrating security throughout the development lifecycle, ensuring the integrity of software components, and maintaining robust response mechanisms for supply chain risks (National Institute of Standards and Technology [NIST], 2022).

Recovery Plan

Operational Timeline and Activities

1inch's recovery effort focuses on two parallel tracks: technical remediation and regulatory/user response. Estimated timeline:

- ***Immediate (0–24 hours)***
 - Revert to the last known clean version of Lottie Player.
 - Implement emergency CSP rules to block the compromised CDN path.
 - Publicly disclose initial findings.
- ***Short-term (1–7 days)***
 - Perform forensic analysis on affected web servers and logs.
 - Identify and notify all users who may be impacted.
 - Implement SRI checks across production systems.
 - Conduct an external security audit of all frontend dependencies.
- ***Medium-term (1–4 weeks)***
 - Integrate SBOM tooling and automate dependency scanning.

- Update incident response playbooks for supply chain scenarios.
- Roll out enhanced monitoring in production.

Complete restoration to “business as usual” is estimated at 3–4 weeks, including forensics, remediation, and communication efforts.

Regulatory and Legal Considerations

Because the breach involved financial transactions, 1inch must assess reporting obligations under multiple frameworks, including:

- **EU GDPR / ePrivacy** for affected EU users (if personal data was exposed through interactions).
- **U.S. state data breach notification laws** vary depending on jurisdiction.
- **Financial regulators** (e.g., FinCEN or its EU equivalents) were concerned about cryptocurrency transactions being stolen.

Proactive engagement with regulators can mitigate potential penalties and support user restitution.

Additionally, 1inch must consider emerging software liability and supply chain disclosure regulations. In the United States, Executive Order 14028 and subsequent Office of Management and Budget (OMB) memos emphasize the importance of SBOMs and software provenance. While these regulations currently focus on federal suppliers, private-sector organizations handling financial transactions are increasingly expected to adhere to similar transparency standards. Failure to disclose in a timely manner or to provide adequate assurances

of software integrity could expose 1inch to civil litigation from affected users or regulatory inquiries into due diligence practices.

Globally, regulatory frameworks such as the EU's Digital Operational Resilience Act (DORA) and the Markets in Crypto-Assets Regulation (MiCA) impose additional obligations on cryptocurrency platforms operating in European jurisdictions. These regulations emphasize rapid incident notification, third-party risk management, and proactive security measures. A supply chain breach like this, even if originating externally, falls within the scope of operational resilience expectations. Therefore, 1inch's legal and compliance teams must collaborate closely with cybersecurity leadership to ensure reporting timelines and disclosure language meet applicable legal thresholds.

Customer Communications

Affected users must be informed promptly. Recommended actions:

- Issue **transparent breach notifications** via email, blog posts, and social media.
- Provide **instructions for revoking malicious approvals** on Avalanche and other chains.
- Offer **customer support escalation paths** for impacted users.
- Consider **compensation or fee credits** to rebuild trust.

Post-Mortem and Lessons Learned

This incident highlights that even organizations with strong internal defenses remain vulnerable through their dependencies. Key lessons for 1inch:

1. **Trust but verify third-party code:** Popular libraries are prime targets for adversaries; automated integrity validation is essential.

2. **Shift-left supply chain security:** Integrate SCA, SBOM, and code signing early in the development lifecycle, aligning with established secure development frameworks such as the NIST SSDF (NIST, 2022).
3. **Establish runtime guardrails:** Client-side security tools can detect unexpected changes that traditional perimeter defenses miss.
4. **Collaborate with the open-source community:** Report issues, support maintainers, and contribute to secure development practices.

Update incident response plans: Traditional IR plans often neglect supply chain scenarios; these must now be first-class citizens.

By treating open-source components as part of the extended attack surface, linch can significantly reduce its exposure to similar incidents in the future.

References

Mitrade. (2024, October 31). *Lottie Player hit with a supply chain attack, stealing 10 wrapped*

BTC from Avalanche wallet. Retrieved from

<https://www.mitrade.com/insights/news/live-news/article-3-442401-20241031>

TryHackMe. (n.d.). *Supply Chain Attacks*. Retrieved from

<https://tryhackme.com/room/supplychainattacks>

National Institute of Standards and Technology (NIST). (2022). *NIST Secure Software*

Development Framework (SSDF), SP 800-218. U.S. Department of Commerce.