# **Event-Stream NPM Supply Chain Attack (2018)**

### **1. Core Issue**

The Event-Stream incident highlighted the dangers of **abandoned open-source projects and unvetted contributors**. Attackers gained control of a popular but lightly maintained Node.js library, *event-stream*, by offering to help with maintenance. Once trusted, they injected a malicious dependency designed to steal cryptocurrency wallet credentials from apps using the library. The weakness was not in the code itself but in the **governance of open-source ecosystems**, where dependency trust is often informal.

### **2. Who Was Attacked**

The primary entry point was the *event-stream* package in the **NPM (Node Package Manager) registry**, a widely used repository in the JavaScript ecosystem. The maintainer handed over access to a new contributor who turned out to be malicious.

### **3. Who Was Affected**

* Developers and organizations using *event-stream* in their applications, particularly those dealing with cryptocurrency.
* Affected downstream projects included **Copay**, a popular Bitcoin wallet, where attackers attempted to siphon user funds.
* While the malicious code was targeted at a narrow set of users, the sheer popularity of *event-stream* (downloaded millions of times) created significant risk across industries.

### 

### 

### **4. Exploit Chain Details**

1. **Maintainer Fatigue** – Original maintainer of *event-stream* accepted an offer from a “volunteer” to maintain the package.
2. **Insertion of Dependency** – The new maintainer added a malicious dependency (*flatmap-stream*) containing an obfuscated backdoor.
3. **Targeted Payload** – The injected code attempted to steal private keys from cryptocurrency wallets built using Copay.
4. **Distribution via NPM** – Since *event-stream* was trusted and widely downloaded, developers unknowingly installed the compromised versions.
5. **Discovery** – A security researcher noticed suspicious code in *flatmap-stream* in November 2018, leading to disclosure and removal of the package.

### **5. Prevention / Protection Steps**

* **Strict Package Governance**: Limit maintainer access and require multi-party review before publishing updates.
* **Dependency Scanning**: Use automated tools (e.g., npm audit, Snyk) to flag suspicious dependencies.
* **Supply Chain Monitoring**: Track sudden maintainer changes or unusual version updates.
* **Crypto Application Hardening**: Isolate and secure key management logic away from external dependencies.

### **6. Fixes & Vendor Response**

* NPM removed the malicious *flatmap-stream* package.
* Copay released patched versions, removing the compromised dependency.
* Security advisories were issued to warn developers about potential key theft.
* Broader discussions began in the open-source community about **sustainability and maintainer trust**.

### **7. If No Fix Available**

* Developers should immediately pin dependencies to known safe versions.
* Conduct audits of open-source libraries, especially for sensitive applications.
* Monitor systems for signs of credential theft or anomalous blockchain activity.

### **8. Reference Material**

* npm Blog – Statement on Event-Stream Compromise:  
   https://blog.npmjs.org/post/180565383195/details-about-the-event-stream-incident
* GitHub Issue – Event-Stream Malicious Package Discussion:  
  <https://github.com/dominictarr/event-stream/issues/116>
* Malwarebytes Blog – NPM Package Event-Stream Compromised:  
   https://blog.malwarebytes.com/threat-intelligence/2018/11/npm-package-event-stream-compromised-to-steal-cryptocurrency/
* The Register – Event-Stream Supply Chain Attack Coverage:  
   https://www.theregister.com/2018/11/26/npm\_repo\_bitcoin\_stealer/
* Snyk Security Advisory – Event-Stream Malicious Dependency (flatmap-stream):  
   https://security.snyk.io/package/npm/event-stream/3.3.6
* GitHub Advisory Database – Event-Stream flatmap-stream Trojan:  
  <https://github.com/advisories/GHSA-8g7p-vh8c-9pp9>

### 

### 

### **9. Further Reading**

* ENISA Threat Landscape for Supply Chain Attacks (2021):  
   https://www.enisa.europa.eu/publications/threat-landscape-for-supply-chain-attacks
* MITRE ATT&CK – Supply Chain Compromise (T1195):  
   https://attack.mitre.org/techniques/T1195/
* OWASP Software Supply Chain Security Guide:  
   https://owasp.org/www-project-software-supply-chain-security/
* OpenSSF Best Practices for npm Ecosystem Security:  
   https://openssf.org/working-groups/supply-chain-integrity/
* SANS Institute – Lessons from the Event-Stream NPM Supply Chain Attack:  
   https://www.sans.org/blog/lessons-from-the-event-stream-npm-supply-chain-attack/

### **10. Tooling**

* npm Audit – Detect vulnerable or malicious dependencies:  
   https://docs.npmjs.com/cli/v8/commands/npm-audit
* GitHub Dependabot Alerts – Identify compromised packages early:  
   https://docs.github.com/en/code-security/dependabot
* Sigstore / Cosign – Sign and verify npm artifacts:  
   https://sigstore.dev/
* Snyk CLI – Scan npm projects for vulnerabilities:  
   https://snyk.io/
* Zeek – Network analysis to detect exfiltration attempts from malicious dependencies:  
   https://zeek.org/
* VirusTotal – Scan suspicious packages or URLs:  
   https://www.virustotal.com/