Supply chain attacks on NPM

Rodrigo Peña

Senior Software Engineer II @ ZeroFox

My stats

- Desde los 14 años en la escena.
- **20+** Sitios del estado vulnerados.
- **100+** Targets vulnerados (RCE, SQLi, LPE, Cred. Stuffing).
- Fundador de GamerLive (AS263797) (2010-2019 🪦).
  - Empresa de IaaS para juegos multiplayer.
  - Víctima de muchos ataques DDoS y script kiddies.
- Bug Bounty: Fintual (US $150), Buda.com (**US $1000** 🤑).
  - Mi bounty de Buda.com fue por un supply chain attack 😳

Agenda

- ¿Qué es un supply chain attack?
- Cómo funciona NPM y por qué es atacable
- Tipos de ataques en el ecosistema NPM
- Casos reales y patrones comunes
- Cómo defendernos: Prácticas, tooling y cultura

¿Qué es un supply chain attack?

- **Definición**: Comprometer un sistema no directamente, sino atacando alguno de sus componentes externos:
  - Dependencias de terceros (librerías NPM)
  - Registries, pipelines de CI/CD
  - Herramientas de build, scripts, imágenes base
- **Objetivo**:
  - Insertar código malicioso
  - Exfiltrar datos / credenciales / tokens
  - Desplegar puertas traseras sin ser detectado

¿Por qué NPM es un objetivo tan atractivo?

- **Ecosistema masivo**:
  - Millones de paquetes, muchos sin mantenimiento activo
  - Cadenas de dependencias muy profundas (decenas o cientos)
- **Modelo de confianza implícita**:
  - `npm install` confía en el registry y en los maintainers
  - Mucho código se ejecuta en `postinstall` / scripts de lifecycle
- **Impacto amplificado**:
  - Un paquete comprometido puede afectar a miles de proyectos
  - Ideal para ataques de ransomware, cryptomining o robo de datos

```
{
        "name": "service-diagnostic-report",
        "private": true,
        "version": "0.0.1",
        "type": "module",
        "scripts": {
                "dev": "vite dev",
                "build": "vite build",
                "preview": "npm run build && wrangler dev",
                "prepare": "svelte-kit sync || echo ''",
                "check": "svelte-kit sync && svelte-check --tsconfig ./tsconfig.json",
                "check:watch": "svelte-kit sync && svelte-check --tsconfig ./tsconfig.json --watch",
                "format": "prettier --write .",
                "lint": "prettier --check . && eslint .",
                "deploy": "npm run build && wrangler deploy",
                "cf-typegen": "wrangler types && mv worker-configuration.d.ts src/"
        },
        "devDependencies": {
                "@cloudflare/workers-types": "^4.20250402.0",
                "@eslint/compat": "^1.2.5",
                "@eslint/js": "^9.18.0",
                "@sveltejs/adapter-cloudflare": "^5.1.0",
                "@sveltejs/kit": "^2.16.0",
                "@sveltejs/vite-plugin-svelte": "^5.0.0",
                "@tailwindcss/forms": "^0.5.9",
                "@tailwindcss/typography": "^0.5.15",
                "@tailwindcss/vite": "^4.0.0",
                "eslint": "^9.18.0",
                "eslint-config-prettier": "^10.0.1",
                "eslint-plugin-svelte": "^3.0.0",
                "globals": "^16.0.0",
                "prettier": "^3.4.2",
                "prettier-plugin-svelte": "^3.3.3",
                "prettier-plugin-tailwindcss": "^0.6.11",
                "svelte": "^5.0.0",
                "svelte-check": "^4.0.0",
                "tailwindcss": "^4.0.0",
                "typescript": "^5.0.0",
                "typescript-eslint": "^8.20.0",
```

Anatomía típica de un ataque en NPM

1. **Compromiso del paquete**:
    - Toma de control de la cuenta del maintainer
    - Ingeniería social para obtener acceso de publicación
    - Cuenta abandonada o paquete huérfano
2. **Inyección de payload**:
    - Código malicioso en `index.js` o dependencias nuevas
    - Scripts en `postinstall` que ejecutan binarios remotos
3. **Distribución**:
    - Publicación de nueva versión en NPM
    - Proyectos que usan `^` o `~` actualizan sin darse cuenta

Patrones de ataque en NPM

- **Typosquatting**:
  - Publicar `react-domm` o `expresss` esperando errores tipográficos
- **Dependency confusion**:
  - Subir a NPM un paquete con el mismo nombre que uno interno
  - Si el resolver prioriza NPM, el atacante gana
- **Maintainer compromise**:
  - Robo de credenciales o tokens de publicación
  - Colaborador malicioso con acceso legítimo
- **Package takeover**:
  - Adoptar paquetes abandonados y cambiar su comportamiento

Vectores técnicos más comunes

- **Scripts de lifecycle** en `package.json`:
  - `preinstall`, `install`, `postinstall`, `prepare`, etc.
  - Ejecución automática cuando alguien instala el paquete
- **Dependencias transitivas**:
  - El ataque se esconde en una dependencia "oscura"
  - Cuesta auditar toda la cadena
- **Exfiltración silenciosa**:
  - Recolección de variables de entorno (`process.env`)
  - Envío a un C2 (command & control) vía HTTP/HTTPS

Ejemplos reales (1/2)

- **event-stream (2018)**:
  - Maintainer cede el control a un tercero
  - Se introduce dependencia maliciosa que roba wallets de Bitcoin
  - Demuestra el riesgo de paquetes muy usados pero poco mantenidos
- **ua-parser-js (2021)**:
  - Cuenta comprometida y versiones maliciosas publicadas
  - Payload instalaba malware para minado de criptomonedas

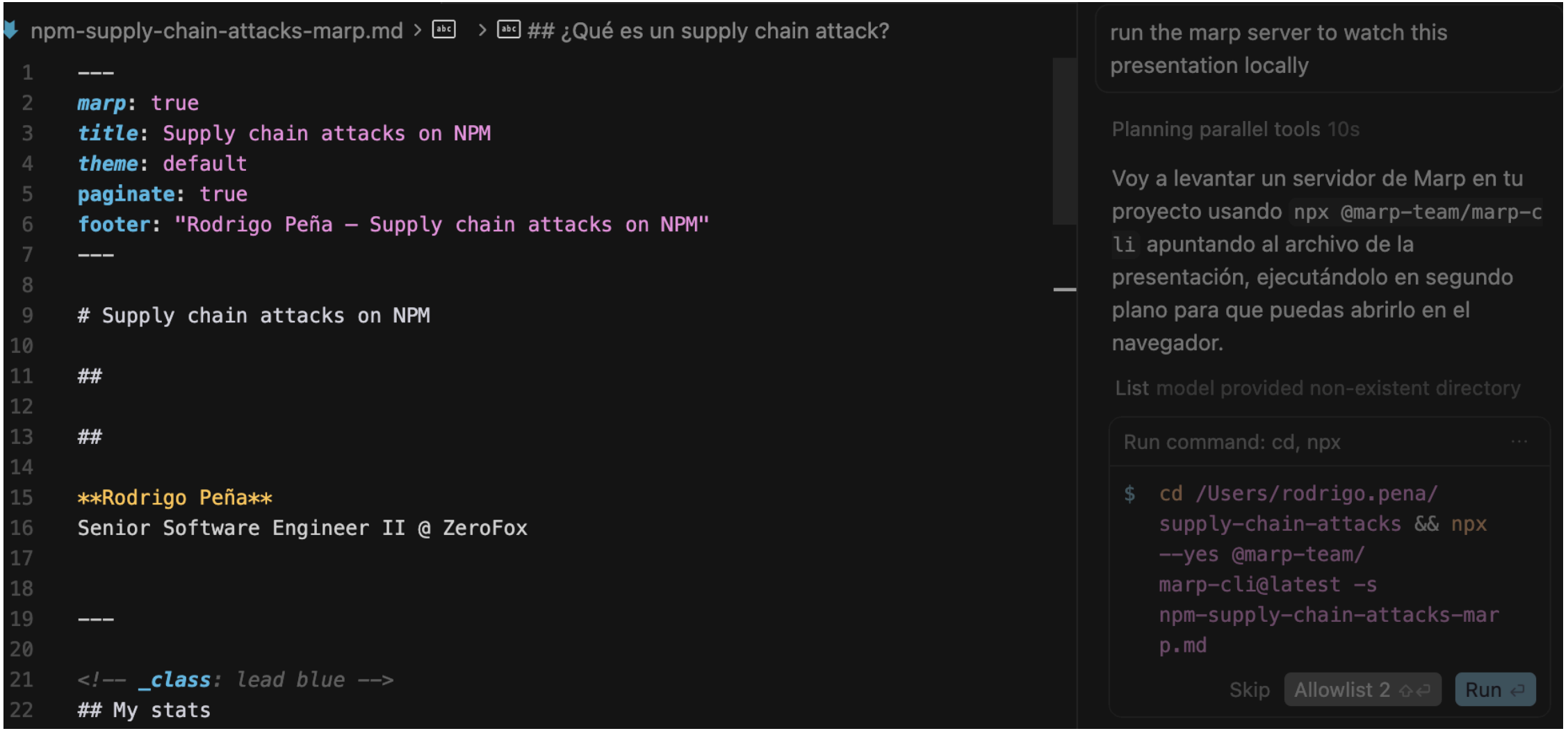
Ejemplos reales (2/2)

- **colors & faker (2022)**:
  - Maintainer introduce código malicioso como protesta
  - Rompe aplicaciones que confiaban ciegamente en las últimas versiones
  - Enseñanza: el riesgo no es solo “hackers externos”
- **singular1ty (agosto 2025)**:
  - Mantenedor de librería NX (herramienta para mantener monorepos) es hackeado
  - Liberan actualización maliciosa de distintos paquetes
  - Ejecuta código arbitrario y exfiltra todos los repositorios de github

Factores de riesgo en equipos y organizaciones

- **Uso de rangos de versión laxos**:
  - `^1.2.3` o `~1.2.3` permitirá actualización automática de paquetes
- **Ausencia de lockfiles**:
  - No se fija la versión exacta (`package-lock.json` , `yarn.lock` , `pnpm-lock.yaml`)
- **Falta de revisión de dependencias**:
  - " `npm install` y listo"
  - Confianza ciega en repos con pocos stars
- **Pipelines de CI/CD sin restricciones**:
  - Instalación desde internet sin proxies ni validaciones

IA Generativa y supply chain attacks



Estrategias de defensa (técnicas)

- **Usar lockfiles y versiones fijas**:
  - Evitar updates automáticos no auditados
- **Revisar scripts de instalación**:
  - Desconfiar de paquetes con scripts innecesarios
- **Configurar registries de confianza**:
  - Usar registries privados / proxies con políticas
  - Opensource: Verdaccio
  - Vendors: Sonatype / JFrog

Hice mi propio malware de npm

supplychain-firewall-benchmark-hello

A benign benchmark package to evaluate how SCA tools and repository firewalls behave with a clean baseline.

Usage

```
npx supplychain-firewall-benchmark-hello
# or, after installing globally
scfbh
```

Output

```
Hello world from supplychain-firewall-benchmark-hello v1.0.0. First versior
```

Versions

v1.0.0 is safe and side-effect-free. Later versions may include controlled test artifacts (EICAR test file, obfuscated install hook) for research purposes.

Install

```
npm i supplychain-firewall-benchmark-hello
```

Repository

github.com/rodrigopv/supplychain-firewall-benchmark-hello

Homepage

github.com/rodrigopv/supplychain-firewall-benchmark-hello#readme

Weekly Downloads

257



Version	License
1.10.2	MIT

Unpacked Size	Total Files

😭





Vue3-ShortKey - a port of `vue-shortkey` for VueJS 3.x accepts shortcuts globaly and in a single listener. I'm not vue3 expert, but I edited this plugin until it started working on Vue-cli with Vue 3.

Install

```
npm install vue-shortkey --save
```

Usage

```
Vue.use(require('vue-shortkey'))
```

Install

```
> npm i vue3-shortkey
```

Repository

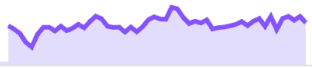
github.com/rodrigopv/vue3-shortkey

Homepage

github.com/rodrigopv/vue3-shortkey/blob/master/README.md

Weekly Downloads

3,723



Version	License
4.0.0	MIT

Final

Rodrigo Peña

Senior Software Engineer II @ ZeroFox