

FakeGit: LuaJIT malware distributed via GitHub at scale



derp.ca - we like the internet.

stealcgithubpolygonluajitinformation-stealeretherhiding

A Vietnamese-speaking operator has been distributing LuaJIT-based malware through GitHub since March 2025. The repos impersonate cracked browser extensions for SaaS tools, gaming cheats, developer utilities, and adult content. Each contains a ZIP archive with a LuaJIT loader chain. BitDefender tracks the archives as Gen:Heur.FakeGit.1 (as of March 2026). ESET tracks the Lua payloads as Lua/Agent.Z through Lua/Agent.BT (as of March 2026) -- 16 distinct obfuscator generations across the campaign.

The loader resolves its C2 by calling a getter function on a Polygon Mainnet smart contract. The operator updates the stored IP through on-chain transactions. Changing infrastructure does not require updating any deployed binary. The final payload, fetched from a GitHub dead drop and decrypted through a four-layer chain (hex, XOR, base64url, AES-ECB), is a **StealIC** information stealer.

600+ unique malicious ZIP archives are confirmed across 47+ GitHub accounts. As of March 4, 2026, at least 25 accounts remain active. URLhaus first detected the campaign in January 2026, roughly 10 months after it started.

Sample overview

Component	SHA256	VT	Notes
V1 ZIP	ebb0c76a03b8bb0ba246b8b31143f4462b4c3b0b3b5d581c499b4c3a484fd792	20/76	cracked-todoist-for-chrome.zip
V1 Lua payload	8b5d6ff49034626532fbf5d2b9f4ffeee53facde63252e3c8aaa89f05029b8d4	16/76	tcp.log
V1 executable	5343326fb0b4f79c32276f08ffcc36bd88cde23aa19962bd1e8d8b80f5d33953	4/76	luajit.exe (100KB, trojanized)

Component	SHA256	VT	Notes
V1 DLL	c7a657af5455812fb215a8888b7e3fd8fa1ba27672a3ed9021eb6004eff271ac	0/76	lua51.dll (3.5MB)
V2a ZIP	2c5d4e26385d968570ffe8c602e431bd7fc88c3f637d0b1736ee7326c754b9ab	16/76	cracked- chrome-for- todoist- v3.0.zip
V2a Lua payload	8cedec35b80b1deaf732c2b178d908f91b3e7a0c114d06dcae9075b8a9bf78b8f	3/76	uix.txt
V2a executable	f3e34c9e36f3be065d80d456281d3dd1cc85eb4980db7fa8c1b0eb6f29c25d8	31/76	luajit.exe (878KB)
V2c Lua payload	772ce19206d35699b4d2693c59f8c0bd1a927f287f8bf98bd14d65f1ff248828	0/76	icon.txt
V2e Lua payload	a563a7df740bce2bda1231cebb4ed136813df43361de17c224b97af9941ee0c4	0/76	ptd.txt
Dead drop Lua	57cc3f7b14c761fcc4a3713a58c1161356fed27fb0cbc04d56bbe6a3ee43f605	0/76	1.json / 7.txt (decrypted)
PE crypter (Feb 27)	e33f444edb92a0d5f01b040dd06520092c0d52a431bbb4e6c2f9e6e71265098c	15/76	2.json (decrypted)
PE crypter (Mar 2)	e2e3768af9e40610b030644486f7434c892bf6fff273732c96f43c9e9401d3aca	14/76	8.txt (decrypted)
StealC build0	ca42898e885979196647b6e5c469461a25870c1f15ef5910d531f37f8f3f147f	42/76	compiled 2026-02-24
StealC build6	63c8cedc49339dd788dbc25f851dfce5219045aef80dd9bd17343948fb73f954	41/76	compiled 2026-03-01
Browser module	61ab1d22949eac0582e989ae065ec4caee9ac99998276317edda96735cd311fb	33/76	embedded in stealer

Lure infrastructure

GitHub repos are created under a mix of purpose-built and compromised accounts. Each repo contains one ZIP archive with a fabricated name and an AI-generated README. Every link in the README -- including Documentation and Support links -- points to the same ZIP download.

The operator instructs victims to enable Chrome Developer Mode and load the extracted folder as an unpacked extension. Some repos also link to a legitimate YouTube video ("How to install unpacked extensions in chrome" by Easy Access Tech, [youtube.com/watch?v=yVvvA8kaIuk](https://www.youtube.com/watch?v=yVvvA8kaIuk)) that walks through the sideloading process.

Lure categories:

- Cracked browser extensions for SaaS tools: Jira, Asana, Todoist, Trello, Monday.com, ClickUp, Figma, Notion, Dropbox, Slack, Zoom, and others

- Gaming cheats: Roblox executors and injectors, Fortnite XP bots and crosshair tools, CS2 radar, Valorant scripts
- Developer tools: n8n workflows, CCNA labs, 7-Zip CVE-2025-0411 PoC, Cursor.so, Sora AI
- Adult content and trading/crypto lures

The ZIP payload sits inside a single directory with an auto-generated name -- a rare English word or technical term. The full list of 47 confirmed names is distinctive enough to use as a hunt signal for unreported repos:

altisonous / amidofluorid / archfire / arthragra / cadetcy / circumcone / clackety / conferee / coronate / cruck / decrepitation / doddart / elegit / exclusivist / febricula / fieldworker / gastrodidymus / Guarrau / gudewife / hangworthy / harmonically / Helenus / heteropod / histometabasis / Iliadic / innkeeper / invenient / isocheim / jogglety / josh / larker / mesomorphy / nonnucleated / opisthophagic / pastoralness / Pelodytes / Phlegethontal / pteroclomorphic / Rajput / rectifiable / sketchiness / squashy / syrtic / tertiarian / textbookless / unjuvenile / ustorious

A Node.js automation toolkit found in the campaign repos (written in Vietnamese) shows the distribution workflow:

1. Read repo name and description from a text file
2. Rename a template directory to the repo name
3. Compress to ZIP
4. Create a GitHub repository
5. Upload the ZIP and tutorial.txt

Inner payload files are byte-for-byte identical across lure categories. Extracting and hashing the contents of cracked extension ZIPs, gaming lure ZIPs, adult content ZIPs, and trading scam ZIPs targeting completely different audiences produces the same SHA256s for the executable, DLL, Lua payload, and launcher. The ZIP hashes differ because internal filenames vary per repo, but the malware binaries inside do not. This confirms single-operator attribution across all lure categories.

Accounts created specifically for this campaign typically have one public repo and a creation date anywhere from 2019 to 2025. Five accounts show signs of compromise -- existing developer accounts where a single malware repo was added alongside otherwise legitimate projects: MaybeDesxie7 (web developer, 19 legitimate repos), Shifalshfaqe (student developer, 14 legitimate repos), Mahmudul-Riad (web developer, 10 legitimate repos), sherinshamr (React project -- malware ZIP injected into `src/assets/`), and SYS123232 (Bannerlord modding account). Many of these compromised accounts have their legitimate repos showing a `pushed_at` date of 2025-10-11, suggesting a batch modification during account takeover.

Payload evolution

The first confirmed FakeGit ZIPs appear on VirusTotal on March 16, 2025, six days after the second C2 domain (`aproxy.app`) was registered. These earliest samples use a legitimate, unmodified 24KB LuaJIT binary from 2017 (0/76) paired with a 242KB obfuscated Lua script (`conf.txt`, `Lua/Agent.Z`, 27/76).

By May 2025, the operator replaced the clean runtime with a trojanized 100KB binary (`5343326f...`). The same SHA256 is present in all V1 Wave 1 ZIPs and in samples dated to May 2025, so the binary was not changed between those two points. The October 2025 jump in volume (16 to 51 ZIPs/month) was a distribution scale-up. The blockchain rotation cadence had already settled at roughly 15 days by June 2025 and did not shift in October.

In February 2026, the operator deployed a new 878KB trojanized binary alongside a second Polygon contract. Multiple EXE names appear across V2 variants: `luajit.exe` (878KB), `vm_s390x.exe` (292KB), `init.exe` (771KB), `luad.exe` (651KB), `gcc.exe` (651KB).

Period	Binary	Size	EXE VT	Lua payload	Lua VT	Contract
Mar 2025	luajit.exe (clean)	24KB	0/76	conf.txt	27/76	V1 (0xd689...)

Period	Binary	Size	EXE VT	Lua payload	Lua VT	Contract
May 2025	lua.exe (trojanized)	100KB	4/76	cli.txt	~16/76	V1
Oct 2025 (Wave 1)	luajit.exe (same binary)	100KB	4/76	tcp.log	16/76	V1
Feb 2026 (Wave 2)	multiple names	292-878KB	25-33/76	uix.txt / lib_bit.txt / icon.txt / rsp.json / ptd.txt	0-3/76	V2 (0x1823...)

Lua payload detection has dropped from 27/76 in March 2025 to 0/76 in three of the six current variants. The operator has iterated through 16 distinct obfuscator generations.

The 600+ figure comes from querying the communicating files endpoint on all 50 C2 IPs across both contracts. URLhaus had catalogued roughly 80 of these ZIPs at time of writing -- the MISP feed captured under 15% of total distribution volume.

Lua payload internals

All six Lua payload variants use the same VM-based obfuscation scheme:

1. Custom base64 alphabet: `L07ker/2Jn4Z+b6y8hfXYVpTiDCvQKc3qRjUEBdo9MzHxtGNgwuA1FSmlaP05wIs`
2. P table: 1,078 entries of encrypted binary constants (~114KB of encoded data per payload)
3. Three-pass table shuffle before decoding: ranges {1,1064}, {1,658}, {659,1064}
4. State machine dispatcher operating on encrypted bytecode from the P table
5. All strings (URLs, API names, keys) assembled at runtime from P table entries

The FFI CDEF block is 8,514 bytes and is identical across all six variants. Declared Win32 API categories:

Screenshot capture: `GetDC`, `CreateCompatibleDC`, `CreateDIBSection`, `SelectObject`, `BitBlt`. Full `BITMAP`, `BITMAPINFO`, `BITMAPFILEHEADER` structure definitions.

Process injection / PE hollowing: `IMAGE_DOS_HEADER`, `IMAGE_NT_HEADERS32/64`, `IMAGE_EXPORT_DIRECTORY`, `VirtualAlloc` with `PAGE_EXECUTE_READWRITE`. `PEB`, `LDR`, and `LDR_DATA_TABLE_ENTRY` structures for manual DLL resolution without IAT entries.

System fingerprinting: `GetComputerNameW`, `IsWow64Process`, `VerifyVersionInfoW`, `GetSystemMetrics`, `TOKEN_ELEVATION`, `OSVERSIONINFOEXW`.

Evasion: `GetConsoleWindow` + `ShowWindow(SW_HIDE)` to suppress the console window.

The C2 backend is nginx/1.24.0 on Ubuntu with a Python FastAPI backend (Pydantic validation errors visible in HTTP responses). POST to `/api/{base64_bot_id}` for check-in; POST to `/task/{base64_bot_id}` for commands.

Blockchain C2 resolver

The loader calls a getter function on a Polygon Mainnet smart contract to retrieve the current C2 URL. This is the same **EtherHiding** technique documented in the [OCRFix campaign](#), applied here to a standalone loader rather than a multi-stage botnet.

V1 contract: `0xd68910ED4D4A5A9bAdF9ec95604CAE0f3378479B`

- Function: `getDomain()`, selector `0xb68d1809`
- Deployed: 2025-02-21

- Deployer: [0x6135f15449b65a962cf250258cf15c3aa16205a2](#)
- Transactions: 50 (48 `updateDomain()` + 1 `destroyContract()` + 1 encrypted blob test)
- Current value: <http://89.169.12.173>

V2 contract: [0x1823A9a0Ec8e0C25dD957D0841e3D41a4474bAdc](#)

- Function: `getData()`, selector [0x3bc5de30](#)
- Deployed: 2025-11-15
- Deployer: [0xdE275aD38C3352A7cb6b0d3efcBF45900c9716f2](#)
- Funded by: [0xF9Bd8BAD6DEB8706f63c16485F7209B8362529eA](#) (27.82 POL from a wallet with 133,000+ transactions)
- Transactions: 9 (all `updateData()`)
- Current value: <http://89.169.12.241>

V1 alternate contract: [0x2cbd2464dd749f5c0034fc9cddc6db2d53dea400](#) -- deployed 2026-01-26 by the same V1 deployer, single transaction, returns <http://158.78.56.52>. Never updated.

The V1 contract started with two C2 domains: [layer1.icu](#) on February 21, 2025, then [aproxy.app](#) on March 10, 2025. On March 17, 2025, the operator switched permanently to raw IPs. The [aproxy.app](#) DNS A record was [77.105.164.45](#), which was the next value set directly in the contract -- the domain-to-IP transition is confirmed in the blockchain record.

C2 rotation cadence averaged 5.5 days from March to May 2025 while the operator iterated on infrastructure. On May 23, 2025, six contract updates fired within 30 minutes, cycling between three IPs -- an operator manually testing C2 switching logic. From June 2025 onward the cadence settled at roughly 15 days per rotation and held that rate through the end of the V1 contract's active period.

The V2 contract was preceded by a test deployment. On November 15, 2025 the operator deployed a [DataManager](#) contract at [0xC224C02Cc0C5B5bf83538B3Db0b6f5b10a2bf6D2](#), stored [127.0.0.1](#), then destroyed it. The production V2 contract went live the same day.

On January 26, 2026, the operator called `destroyContract()` on the V1 contract. The call succeeded but the contract survived due to EIP-6780 -- self-destruct only removes code when called in the same transaction as contract creation. The same day, the operator tested a 128-byte encrypted blob as contract storage before reverting to plaintext IPs. Both contracts received new IP values on the same day, suggesting a planned infrastructure migration.

GitHub dead drops and decryption chain

The Lua payload fetches two encrypted blobs from GitHub repos that serve as a secondary delivery mechanism:

- Small blob (~594KB hex-encoded): delivers an updated Lua payload for hot-patching without modifying the lure repos
- Large blob (~2.3MB hex-encoded): delivers the PE crypter stub that decrypts and injects the final StealC payload

Current dead drop repos:

- github.com/rd898/package (json/1.json, json/2.json) -- purpose-built account, created 2022-09-14
- github.com/Mahmudu1-Riad/www (index/7.txt, index/8.txt) -- compromised legitimate developer account

Both repos serve the same small blob (1.json and 7.txt share the same SHA256). The large blob differs between repos and across rotations.

Decryption chain for both blobs:

```

hex-encoded blob
--> hex decode
--> XOR decrypt (32-byte repeating key: Ece6VGLRJum2qYt1790i0U7aHot7Zhbn)
|
+--> obfuscated Lua payload (small blob, 297KB, 0/76)
|
+--> PE crypter stub (large blob, ~1.2MB, 15/76 VT)

```

The XOR key is not stored as a literal in any payload file. The Lua VM assembles it at runtime from encrypted P table entries.

PE crypter:

Both observed builds are x64 GUI executables importing only KERNEL32.dll. All other APIs are resolved via PEB walking. The RCDATA section (resource ID 101) contains approximately 1MB of base64url-encoded, AES-256-ECB-encrypted content.

At RVA 0x1000, the crypter XORs two hardcoded 32-byte constants using AVX2 `vpxor ymm1, ymm0, [rbp+0x20]` to produce the AES-256 key at runtime. The key is stored in a global `std::string`, used, then zeroed out.

PE crypter build	AES-256 key	Compiled
Dead drop 2 (build0)	Y0Yd5_dZpfW_nmA-rTxg61gpdbu_26-u	2026-02-27 12:44 UTC
Dead drop 8 (build6)	51Aabz1G90_c_K9JZ8YXRGTOFralTd8k	2026-03-02 21:10 UTC

Each build has a unique AES key (different XOR constants compiled in). Both decrypt to a 779,808-byte information stealer.

Injection: `CreateProcessA` (suspended) -> `NtUnmapViewOfSection` -> `VirtualAllocEx` -> `WriteProcessMemory` -> `SetThreadContext` -> `ResumeThread`.

StealC payload

Both decrypted inner PEs are 779,808-byte x64 GUI executables. They share 99.3% of their code and differ only in the encrypted config strings in .rdata and the build tag.

Build	Compiled	SHA256	VT
build0	2026-02-24 11:27 UTC	ca42898e885979196647b6e5c469461a25870c1f15ef5910d531f37f8f3f147f	42/76
build6	2026-03-01 16:06 UTC	63c8cedc49339dd788dbc25f851dfce5219045aef80dd9bd17343948fb73f954	41/76

An embedded browser module (163,840 bytes) sits at .data offset 0x8DF20 and is shared across builds. SHA256: [61ab1d22949eac0582e989ae065ec4caee9ac99998276317edda96735cd311fb](#). VT: 33/76. Detection names include StealC, Luca Stealer, Win64/Spy.Agent.NZ. This module handles Chrome, Brave, and Edge credential extraction including App-Bound Encryption bypass. It has a `.fptable` PE section not commonly seen in other families.

Config encryption: RC4. The first string in the .rdata config block is the RC4 key itself. The remaining 266 strings are base64(RC4(plaintext)). All 120+ Win32 API names are encrypted and resolved at runtime via PEB walking.

Build	RC4 key	Stealer C2
build0	qAw6EF2z3ycMeACsJA	http://217.119.129.110/3d9c1a1d0dc9436eb7b7.php
build6	dVh1lnj7MBHRDVTFDt	http://213.176.72.200/3d9c1a1d0dc9436eb7b7.php

Both stealer C2 IPs are on ASN 207957 (Serv.host Group Ltd), the same ASN as 37 of the 48 loader C2 IPs. [213.176.72.200](#) is in the same /23 as loader C2s [213.176.72.204](#) and [213.176.72.209](#).

Two additional StealC builds appeared on [217.119.129.110](#) in late February 2026 that are not from the dead drops analysed here:

SHA256	VT	Compiled
b0ca436fef7c39e0f6ebd93989d38b9a58eb4d1cf83ab45d7b07d70794a59e6d	52/76	~2026-02-27
a3bc1a47f6bd9782dc594ef10feef0ee8c422a1cd06a5ddc429c61fdfa81b567	43/76	~2026-02-28

Both detected as Stealer.StealC!1.133F7 / TrojanPSW:Win64/StealC. The VT filenames carry a community clustering suffix ([cobalt-strike_icedid_luca-stealer_njrat_stealc](#)) that is a tagging convention, not a malware name.

Stealer targets (from decrypted config strings):

- Browsers: Chrome, Brave, Edge. Login Data, Cookies, Web Data, History, Extension Settings, IndexedDB. App-Bound Encryption via [os_crypt/encrypted_key](#) from Local State. Chrome v10 and v20 cookie encryption formats.
- Firefox: NSS3 direct decryption via [NSS_Init](#), [PK11_GetInternalKeySlot](#), [PK11SDR_Decrypt](#). Targets [logins.json](#), [cookies.sqlite](#), [formhistory.sqlite](#), [places.sqlite](#).
- Email: Outlook (registry paths for Office 13.0 through 16.0 and Windows Messaging Subsystem), FoxMail.
- SSH: WinSCP sessions ([Software\Martin Prikryl\WinSCP 2\Sessions](#)).
- Gaming: Steam tokens and config files ([ssf*](#), [config.vdf](#), [DialogConfig.vdf](#), [loginusers.vdf](#)).
- Loader: PowerShell [iex\(New-Object Net.WebClient\).DownloadString\('...'\)](#) download-and-execute and MSI installer via [msiexec /passive](#).

Exfiltration via JSON POST with [opcode](#), [data](#), and [upload_file](#) fields to the PHP endpoint. Screenshots via GDI+ ([GdiplusCreateBitmapFromHBITMAP](#) -> [GdiplusSaveImageToStream](#) -> [screenshot.jpg](#)). System fingerprint written to [system_info.txt](#). Output directory structure: [soft\Outlook\](#), [soft\Steam\](#), [soft\WinSCP\](#), [soft\FoxMail\](#).

Full infection chain

1. Victim extracts ZIP, runs Launcher.cmd
2. Launcher.cmd: start [lua_jit.exe](#) <payload.txt>
3. Lua VM hides console window ([GetConsoleWindow](#) + [ShowWindow\(SW_HIDE\)](#))
4. [eth_call](#) to Polygon contract -> current C2 IP
5. Fetch two hex blobs from GitHub dead drop repos
6. Small blob: hex -> XOR -> updated Lua payload (0/76, hot-patch)
7. Large blob: hex -> XOR -> PE crypter stub (15/76 VT)
8. PE crypter: XOR two embedded 32-byte constants -> AES-256 key
9. PE crypter: [base64url](#) decode RCDATA -> AES-256-ECB decrypt -> 779KB StealC
10. PE crypter: [CreateProcessA](#) (suspended) -> hollow -> inject -> resume
11. StealC: RC4-decrypt 267 config strings (API names, C2 URL, target paths)
12. StealC: collect credentials, cookies, tokens
13. StealC: JSON POST to [/3d9c1a1d0dc9436eb7b7.php](#)
14. StealC: optionally download additional payloads via PowerShell [iex](#) or [msiexec](#)

C2 infrastructure

All 52 endpoints are on bulletproof hosting providers. ASN 207957 (Serv.host Group Ltd) accounts for 37 of the 48 loader IPs and both stealer IPs. The remaining 11 loader IPs are split across four ASNs, with QWINS (AS213702) being the only other recurring provider.

ASN	Provider	Loader IPs	Period
207957	Serv.host Group Ltd	37	2025-03-18 to present
213702	QWINS	6	2025-05-23 to 2025-12-22
213877	u1host	3	2025-03-31 to 2025-04-11
207043	Dedik	2	2025-12-02 to 2025-12-11
209207	Digital Hosting	1	2025-05-06
216300	AbkhazMedia	1	2025-03-17 (domain phase)

Loader C2 IPs (48 total):

IP	ASN	Country	Contract	First seen in contract
58.64.137.69	17444 (HKBN)	HK	layer1.icu DNS	2025-02-21
77.105.164.45	216300 (AbkhazMedia)	AM	V1 + aproxy.app DNS	2025-03-17
89.169.12.115	207957	DE	V1	2025-03-18
89.169.13.30	207957	RU	V1	2025-03-21
91.196.33.33	207957	RU	V1	2025-03-23
89.169.12.119	207957	DE	V1	2025-03-25
89.169.12.78	207957	DE	V1	2025-03-28
91.196.34.17	207957	DE	V1	2025-03-29
185.184.123.138	213877 (u1host)	NL	V1	2025-03-31
185.184.122.38	213877 (u1host)	US	V1	2025-04-02
89.169.12.42	207957	DE	V1	2025-04-06
150.241.108.62	213877 (u1host)	US	V1	2025-04-11

IP	ASN	Country	Contract	First seen in contract
89.169.12.179	207957	DE	V1	2025-04-26
193.233.126.91	209207 (Digital Hosting)	NL	V1	2025-05-06
91.196.32.114	207957	RU	V1	2025-05-15
95.164.53.26	213702 (QWINS)	DE	V1	2025-05-23
95.164.53.153	213702 (QWINS)	DE	V1	2025-05-23
185.170.153.239	207957	US	V1	2025-06-03
185.170.153.243	207957	US	V1	2025-06-05
185.170.153.173	207957	US	V1	2025-06-08
91.196.33.27	207957	RU	V1	2025-06-13
91.196.33.38	207957	RU	V1	2025-06-22
89.169.13.215	207957	RU	V1	2025-06-27
80.66.85.195	207957	DE	V1	2025-07-07
91.196.32.121	207957	RU	V1	2025-07-22
91.196.34.40	207957	DE	V1	2025-08-01
94.156.155.34	207957	BG	V1	2025-08-18
193.23.200.78	207957	SE	V1	2025-09-06
45.149.235.146	207957	DE	V1	2025-10-04
94.156.154.194	207957	BG	V1	2025-10-20
185.170.154.101	207957	US	V1	2025-11-09
93.123.39.74	213702 (QWINS)	EE	V2	2025-11-16
84.21.189.135	213702 (QWINS)	DE	V2	2025-12-11
151.243.113.70	207043 (Dedik)	DE	V1 + V2	2025-12-02
151.243.113.15	207043 (Dedik)	DE	V1	2025-12-11
78.40.209.225	213702 (QWINS)	FI	V2	2025-12-22

IP	ASN	Country	Contract	First seen in contract
93.123.39.246	213702 (QWINS)	EE	V1	2025-12-22
144.31.219.13	207957	DE	V1	2026-01-01
144.31.219.15	207957	DE	V2	2026-01-01
213.176.72.209	207957	GB	V1	2026-01-26
213.176.72.204	207957	GB	V2	2026-01-26
89.169.12.173	207957	DE	V1 (current)	2026-02-08
89.169.12.160	207957	DE	V2	2026-02-08
89.169.12.241	207957	DE	V2 (current)	2026-02-27
158.78.56.52	--	US	V1 alt	2026-01-26
213.176.73.151	207957	FR	V2 sandbox	--
213.176.73.163	207957	FR	V2 sandbox	--
64.188.98.20	207957	US	V1 sandbox	--

151.243.113.70 (Dedik, AS207043) appears in both V1 and V2 contracts and hosts concurrent, unrelated malware operations: ANDROMEDA/Gamarue (47-50/76), Wallstealer/luau.exe builds (37-45/76), LummaC2 (32/76), Aurora Stealer (43/76). This is shared bulletproof hosting -- the other families are not attributed to this operator.

Stealer C2 IPs:

IP	ASN	Country	Build
217.119.129.110	207957	LV	build0
213.176.72.200	207957	GB	build6

C2 domains:

Domain	VT	Created	A record
layer1.icu	11/94	2025-02-21	58.64.137.69 (HK, AS17444)
aproxy.app	15/94	2025-03-10	77.105.164.45 (AM, AS216300)

IOC summary

ZIP archives

SHA256	Variant	Representative filename
ebb0c76a03b8bb0ba246b8b31143f4462b4c3b0b3b5d581c499b4c3a484fd792	V1	cracked-todoist-for-chrome.zip
2c5d4e26385d968570ffe8c602e431bd7fc88c3f637d0b1736ee7326c754b9ab	V2a	cracked-chrome-for-todoist-v3.0.zip
89730887d1ac7aab9e37d590081a5c9c56fa3f79b24960bff3f9cdabb0486424	V2b	Buyhatke...1.9.zip
4508b938d16e16585d4065d10ba5890f380a9d7e82d1ae0246bae41937d17add	V2c	cracked_autocomplete...2.5.zip
365d1b01cd5e605f6e7d39b192aaf56040176f2bce6eaec0fc11584850793a83	V2d	cracked-annotator-webpage-..._2.1.zip
61f5e68f1dc91eee925af875bf511de0de1483a8e663567c22ca4bc568808341	V2e	to-extension-save-cracked-milanote-revalidate.zip

Zip hashes differ across repos because internal filenames vary per lure. The executable and Lua payload hashes inside are identical for all ZIPs sharing the same variant.

Executables

SHA256	Filename	VT	Notes
5343326fb0b4f79c32276f08ffcc36bd88cde23aa19962bd1e8d8b80f5d33953	luajit.exe / lua.exe	4/76	Trojanized, 100KB, in use May 2025 through Wave 1
f3e34c9e36f3be065d80d456281d31dd1cc85eb4980db7fa8c1b0eb6f29c25d8	luajit.exe	31/76	V2a, 878KB
1f546f1b4b2cd404fdd2ba9b52223ecf935ad26588993f703bef69db4c94034e	vm_s390x.exe	25/76	V2b, 292KB
ac5885b78810a7bf987ff6674f6717059e227df9c969b9fb46d00b2c0de1ba74	init.exe	33/76	V2c, 771KB
88ec32a311b56441cfe6126b7780f073f36dfb8808de0dab9219d1a0be9c01ac	luad.exe	25/76	V2d, 651KB
2ea6200c846af534a07338a803acf7f49520abf59b2ae82a701a24e7fada0b97	gcc.exe	27/76	V2e, 651KB

Lua payloads

SHA256	Filename	VT
8b5d6ff49034626532fbf5d2b9f4ffeee53facde63252e3c8aaa89f05029b8d4	tcp.log	16/76
8cedce35b80b1deaf732c2b178d908f91b3e7a0c114d06dfae9075b8a9bf78b8f	uix.txt	3/76
ead242a036f033f0829afa33768814be168ae29431d9de479e63a80c4cf9f431	lib_bit.txt	0/76
772ce19206d35699b4d2693c59f8c0bd1a927f287f8bf98bd14d65f1ff248828	icon.txt	0/76
acdd9ac3d55c4b1e12a239331145665d9eddf84a7a34fccac164f8afd006f875	rsp.json	1/76
a563a7df740bce2bda1231cebb4ed136813df43361de17c224b97af9941ee0c4	ptd.txt	0/76
57cc3f7b14c761fcc4a3713a58c1161356fed27fb0cbc04d56bbe6a3ee43f605	dead drop Lua (decrypted)	0/76

Launchers

SHA256	Filename	Command
2ad929f97ae428864a4e74c8f44c9aa392f8e044ae960b6906ffac9bfcbb2a43c	Launcher.cmd	start luajit.exe tcp.log
ce1e33483d353200a266b3bc383ccf500e5a760c6dcd8218747260f5bbe39509	Launcher.cmd	start luajit.exe uix.txt
0476281e2ed8cca25b881092334c2aa7bca82c4be9819fcb9cefe8027e532f1e	LaunchTool.cmd	start vm_s390x.exe lib_bit.txt
a3e6236fa517f863ea551e17fcedcfc22df23e044dcd4863e6ddf3ef00966596	LaunchApp.bat	start init.exe icon.txt
dce5fc61e153377bb42b53b7dfca841399dd5c94d5fdf2631617a6df8c7cded1	Launch.bat	start luad.exe rsp.json
37f60b7923dae8ab38bb72ccfe94bf06c3266202d4881e9d2e0ac28abcd85b25	Launch.cmd	start gcc.exe ptd.txt

DLLs

SHA256	Filename	VT
c7a657af5455812fb215a8888b7e3fd8fa1ba27672a3ed9021eb6004eff271ac	lua51.dll (3.5MB)	0/76
ef224f48255771d28b15169d60c8b0a925875dba15b3bd245c0b34d77290120d	lua51.dll (488KB)	0/76

Dead drop files (pre-decryption)

SHA256	File	Size
2d92df6beff6ccdb68272adedc1673ea12c1a19efecb77a16f9161ecda56ae8b	1.json / 7.txt	594KB
b4f46436037bf0c12eb047d87999c956b09c5cf58d03221190f21e5ce7f97a81	2.json	2.37MB
7625f9ae947f8cb3d7ed463694db72bd3eb07d1792e1a85232b1933370b9562b	8.txt	2.37MB

Stealer payloads

SHA256	Component	Compiled	VT
ca42898e885979196647b6e5c469461a25870c1f15ef5910d531f37f8f3f147f	StealC build0	2026-02-24	42/76
63c8cedc49339dd788dbc25f851dfce5219045aef80dd9bd17343948fb73f954	StealC build6	2026-03-01	41/76
b0ca436fef7c39e0f6ebd93989d38b9a58eb4d1cf83ab45d7b07d70794a59e6d	StealC (additional build)	~2026-02-27	52/76
a3bc1a47f6bd9782dc594ef10feef0ee8c422a1cd06a5ddc429c61fdfa81b567	StealC (additional build)	~2026-02-28	43/76
e33f444edb92a0d5f01b040dd06520092c0d52a431bbb4e6c2f9e6e71265098c	PE crypter (DD2)	2026-02-27	15/76
e2e3768af9e40610b030644486f7434c892bf6ff273732c96f43c9e9401d3aca	PE crypter (DD8)	2026-03-02	14/76
61ab1d22949eac0582e989ae065ec4caee9ac99998276317edda96735cd311fb	browser module	2026-02-15	33/76

Kirk's note: If anyone knows an efficient way to bulk-report malicious GitHub repositories or accounts, please reach out to . Reporting 25+ accounts one at a time through the web form isn't practical, and emailing support@github.com bounced.

GitHub accounts (still live as of 2026-03-03)

Cracked extension repos: teskkkkk, bibabiboreal, class1k, hamzaabiadi, jsm2raj, kayraizm3131, kukil-saikia, lazydave, TeeeeeeeeellKall, DARKPHATOM, MaybeDesxie7, sameeronwheels, j yng2002, Shifalshfaqe, clieez

Gaming and utility lure repos: hydrangeapaniculatabokchoi171, jorgeZdark, Masbe2117, kenshigenesis-jpg, SYS123232, blackwall0220, sherinshamr, Vitor-gf

Dead drop repos: rd898, Mahmudul-Riad

Blockchain contracts (Polygon Mainnet)

0xd68910ED4D4A5A9bAdF9ec95604CAE0f3378479B	V1 loader contract (active)
0x2cbd2464dd749f5c0034fc9cddc6db2d53dea400	V1 alternate contract (active)
0x1823A9a0Ec8e0C25dD957D0841e3D41a4474bAdc	V2 loader contract (active)

MITRE ATT&CK

ID	Technique
T1566.002	Phishing: Spearphishing Link
T1204.002	User Execution: Malicious File
T1059.007	Command and Scripting Interpreter: Lua
T1036.005	Masquerading: Match Legitimate Name
T1036.007	Masquerading: Double File Extension
T1218	Signed Binary Proxy Execution
T1027	Obfuscated Files or Information
T1027.002	Software Packing
T1140	Deobfuscate/Decode Files
T1102.001	Web Service: Dead Drop Resolver
T1053.005	Scheduled Task/Job
T1055.012	Process Hollowing
T1113	Screen Capture
T1106	Native API
T1555.003	Credentials from Web Browsers
T1555	Credentials from Password Stores
T1539	Steal Web Session Cookie
T1586.001	Compromise Accounts: Social Media
T1583.005	Acquire Infrastructure: Botnet

What this analysis cannot determine: victim count, whether additional distribution platforms exist beyond GitHub, or the full extent of the operator's infrastructure beyond what the blockchain records and VT communicating files data expose. The 600+ ZIP figure is a lower bound -- repos removed before analysis would not appear in VT telemetry.

The campaign has processed 600+ distinct ZIP archives across 47+ GitHub accounts over 13 months without a sustained platform disruption. The V1 Polygon contract has rotated through 50 C2 values. Both V1 and V2 contracts are currently active.

See also: [OCRFix EtherHiding botnet](#).

If you operate a threat intelligence platform with API access and can provide a researcher account, please reach out to . Additional data sources directly increase the quality and coverage of the threat intel published here.

K

Kirk

I like the internet. Want to get in touch? · [GitHub](#) ·