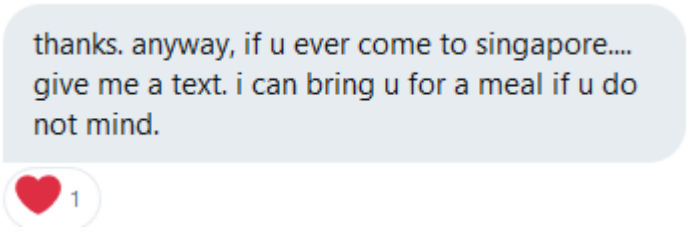
Quick analysis note about GuLoader (or CloudEyE)

Published: 2020-06-27 · Archived: 2026-04-05 20:25:55 UTC

Recently, I've supported a foreign friend on Twitter during the analysis one of GuLoader (or CloudEyE) variant sample. Although, he have read these articles (1, 2) but still stuck and I know that feeling.

The discussion between us was quite long, finally I sent him my quick analysis so that he can read and follow. Now, I put the analysis that we discussed on this blog hoping it will help others like him.

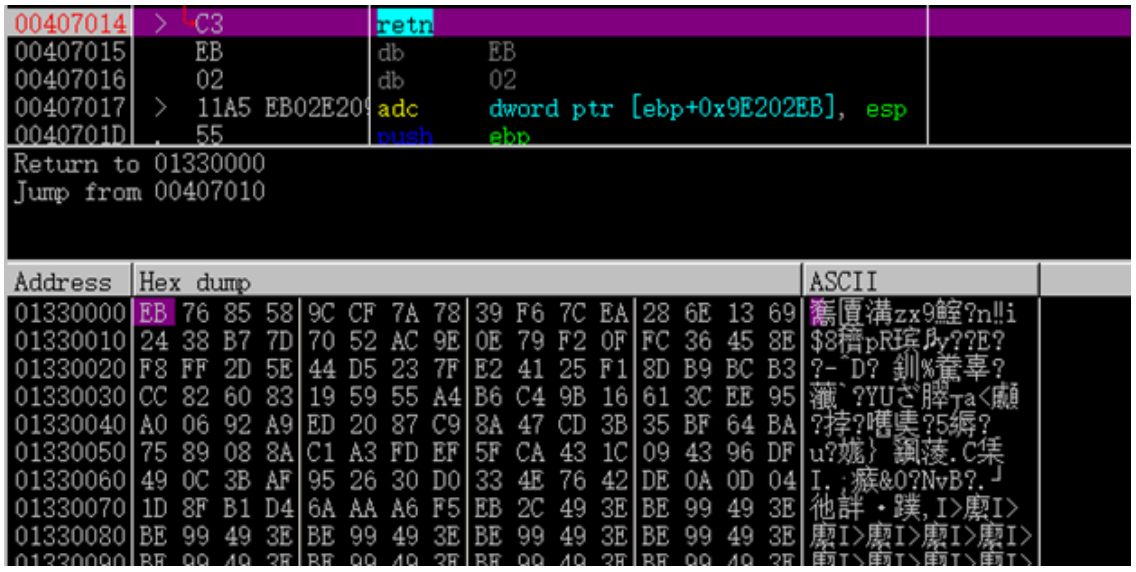


1. Get the GuLoader's shellcode

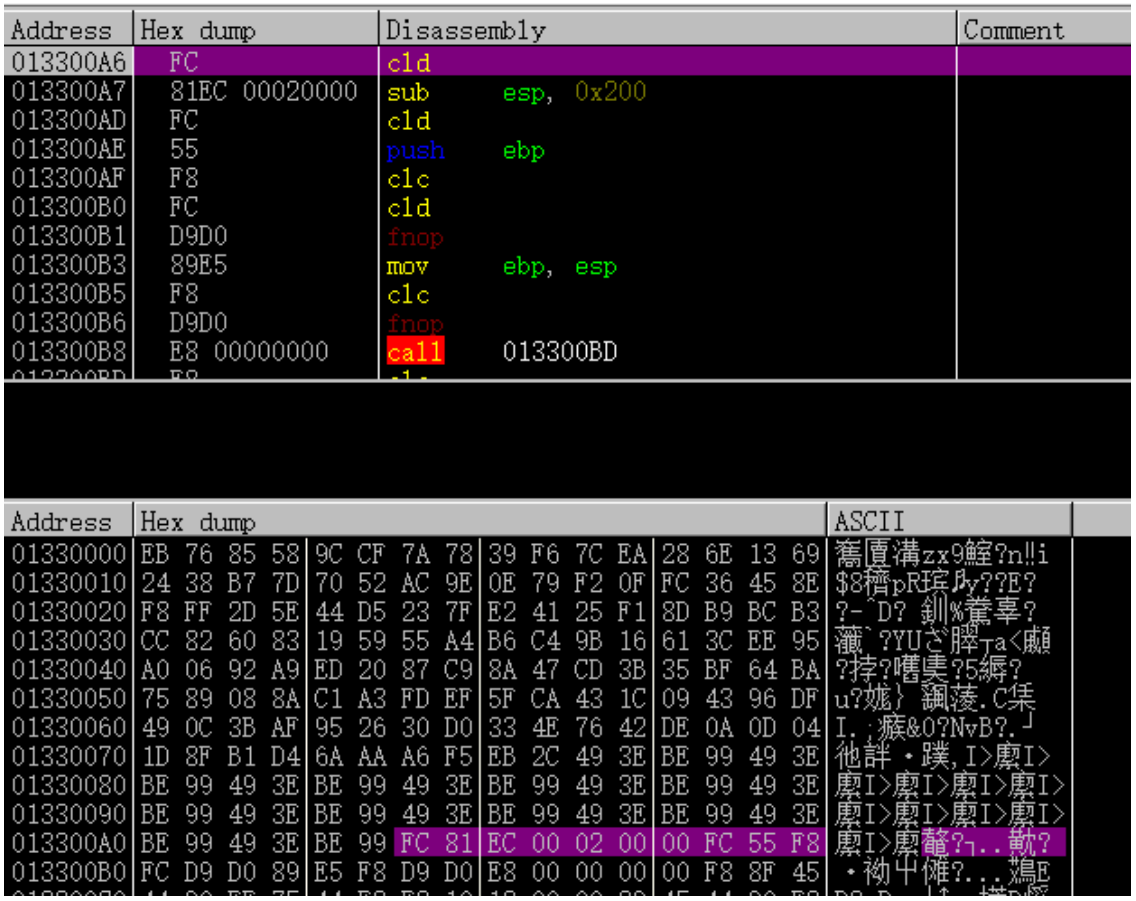
GuLoader uses **VirtualAlloc** api for allocating a new memory section and drop shellcode to the allocated memory.

- Call to **VirtualAlloc**:

- Fill shellcode to the allocated buffer:



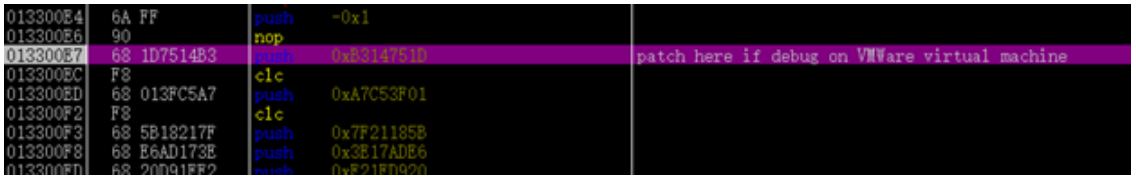
- Continue trace, will jump to the shellcode. This shellcode may vary with each sample:



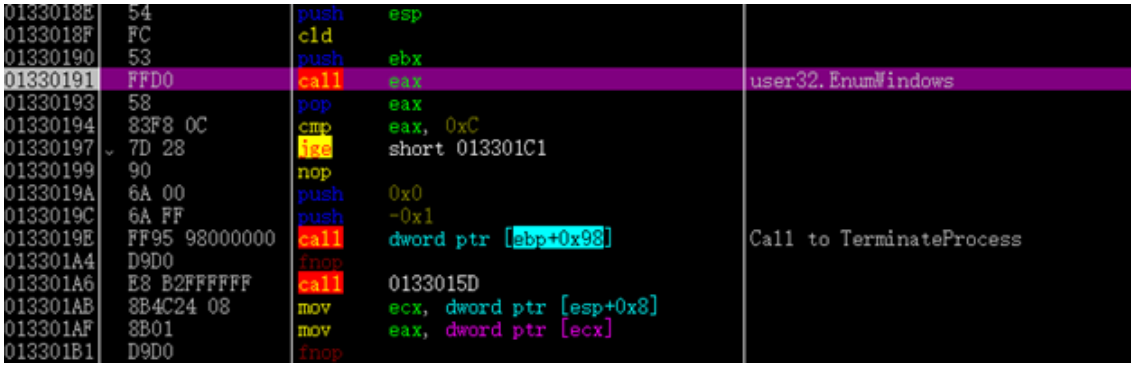
2. Debug shellcode for finding the next payload URL

This shellcode uses **Heaven's Gate technique** to execute on x64 environment. You can read more about this technique that I wrote here. Preferably, you should debug GuLoader on *32bit Windows VM*.

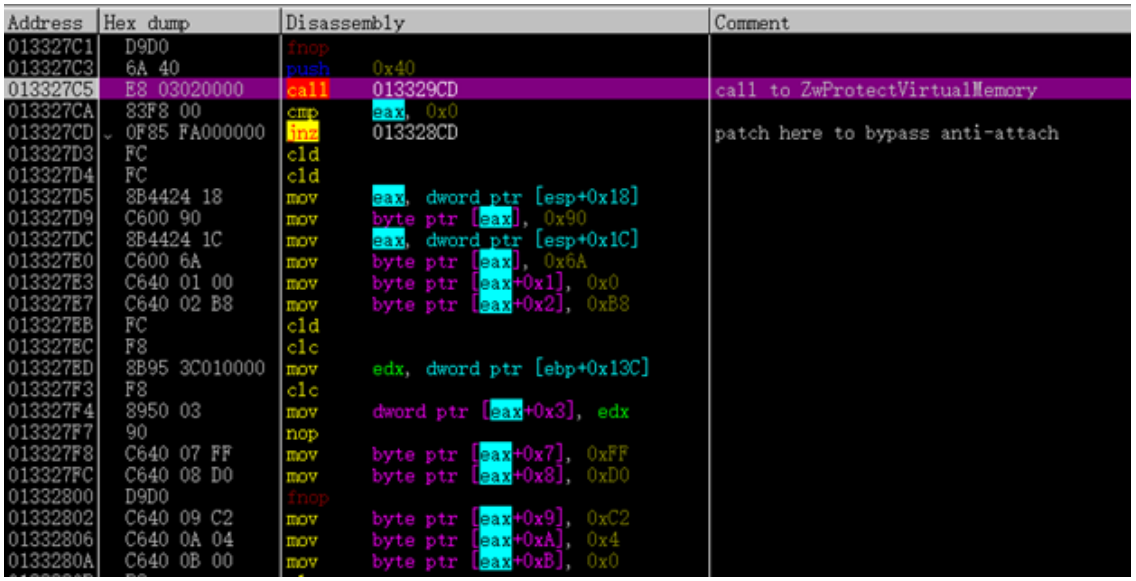
- Patch to bypass anti-VM:



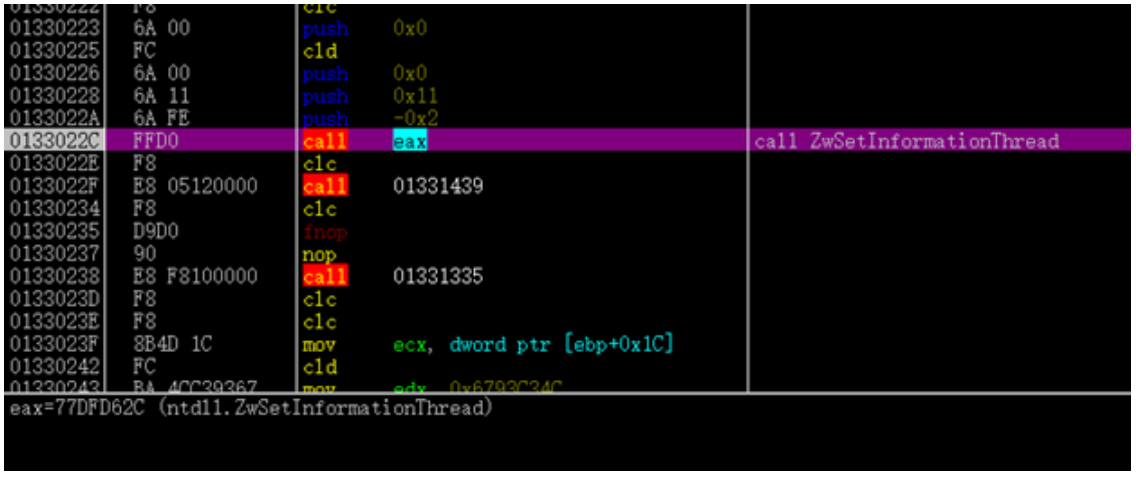
- Break on call to **EnumWindows** (*patch if need to bypass call to* **TerminateProcess**):



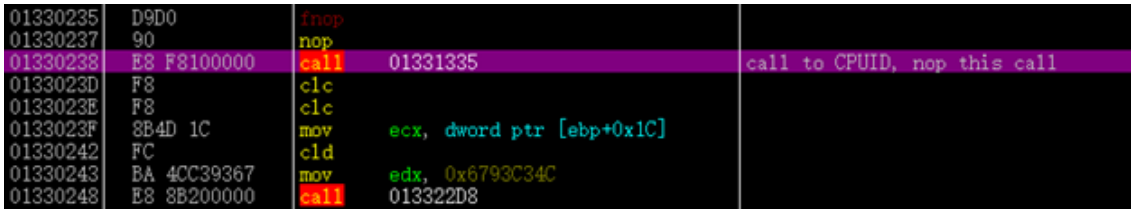
- Break on call to **ZwProtectVirtualMemory** (*need to patch to bypass anti-attach*):



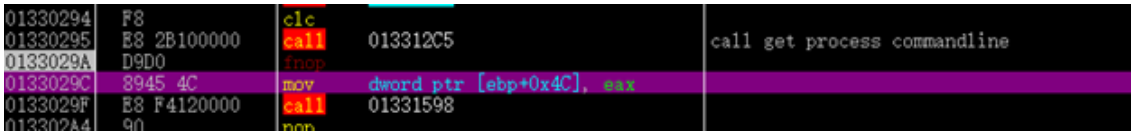
- Break on call to **ZwSetInformationThread** for hidding thread (*need to patch **0xC3** when trace into this call or nop this call*):

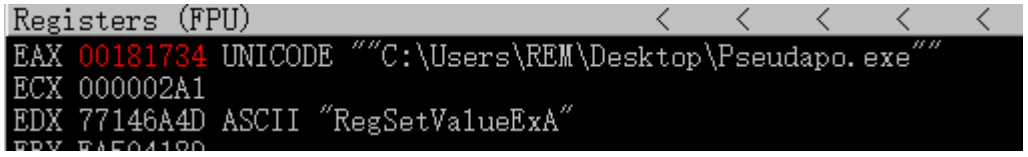


- Directly, below will usually be the sub function that call to the **CPUID** command, nop this call:

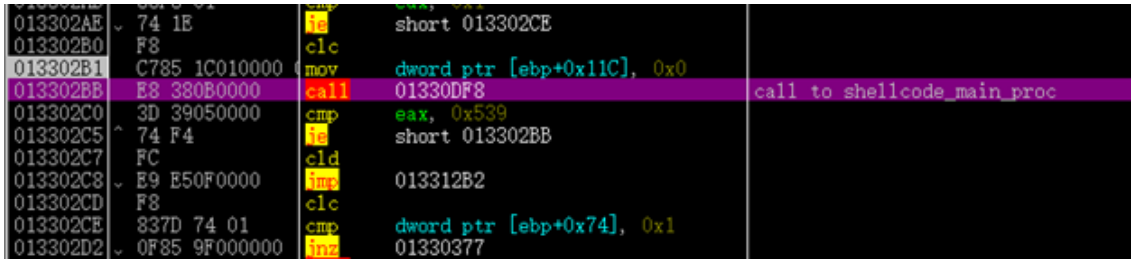


- Call to get process command line:





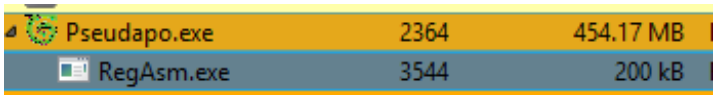
- Call to shellcode main proc, need to trace into this func:



- This shellcode main proc will do:

_ Get RegAsm's path (ex: *C:\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe*)

_ Call to **kernel32.CreateProcessInternalW** to create **RegAsm.exe** in ***suspended state***:

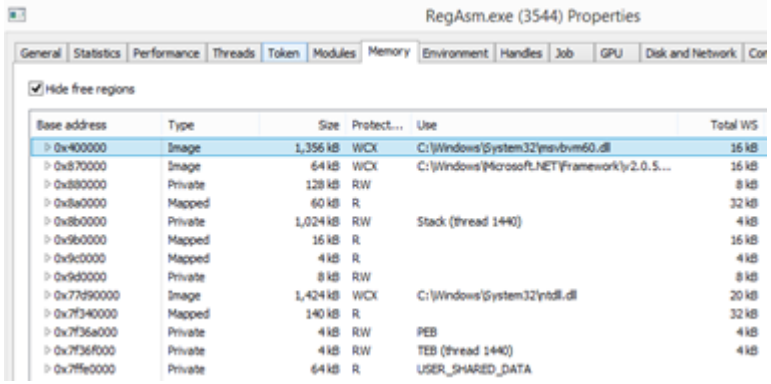


_ Get **msvbvm60.dll**'s path (ex: *C:\Windows\system32\msvbvm60.dll*) and then replace to *\??\C:\Windows\system32\msvbvm60.dll*

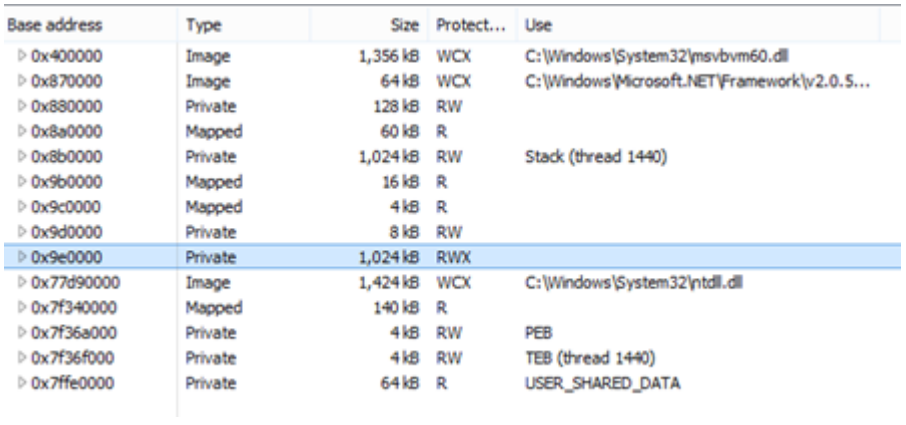
_ Call to **ntdll.ZwOpenFile**

_ Call to **ntdll.ZwCreateSection** with *FileHandle* of **msvbvm60.dll** (ex: File, C:\Windows\System32\msvbvm60.dll, 0x190)

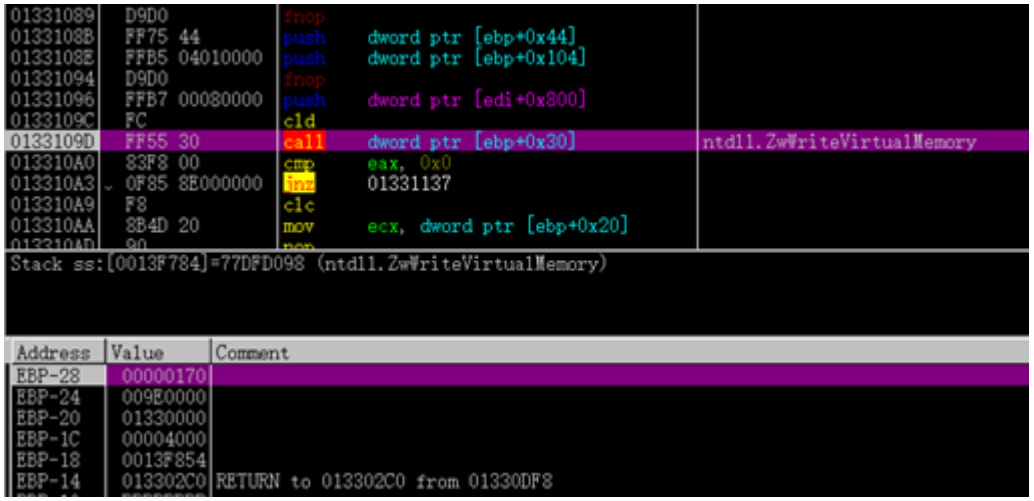
_ Call to **ntdll.ZwMapViewOfSection** with *SectionHandle* of **msvbvm60.dll** and *ProcessHandle* of **RegAsm.exe** suspended process. For mapping msvbvm60.dll:

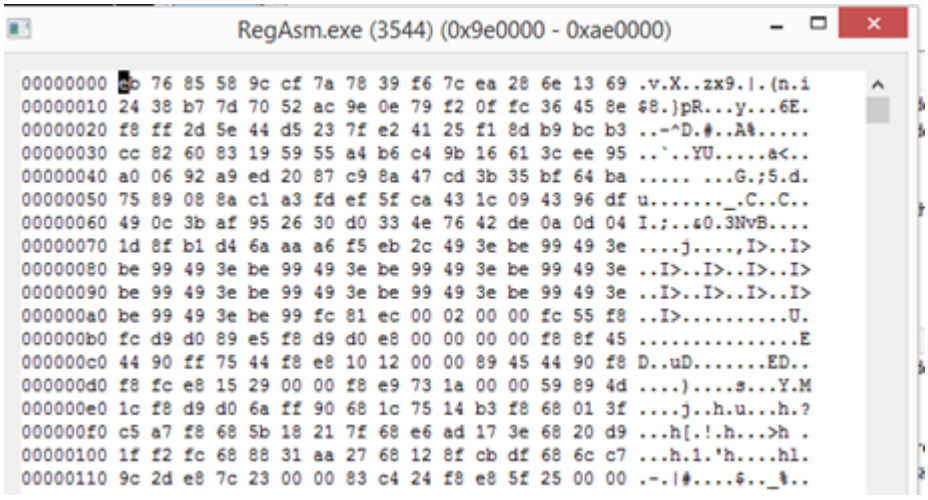


_ Allocate **RWX** memory section on **RegAsm.exe** suspended process:



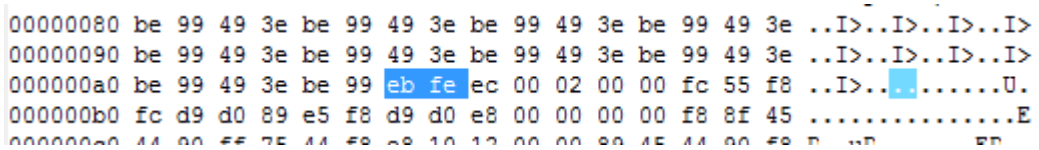
_ Then call **ZwWriteVirtualMemory** for writing the 2nd shellcode to the allocated buffer at **RegAsm** process. The 2nd shellcode same as the 1st shellcode, but its main task is to decode the URL and download the final payload.



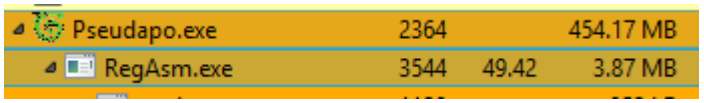


_ After that it calls **ZwGetContextThread**, **ZwSetContextThread** and then **ZwResumeThread**. So **RegAsm** process will return to the normal state and execute the 2nd shellcode to download the final payload.

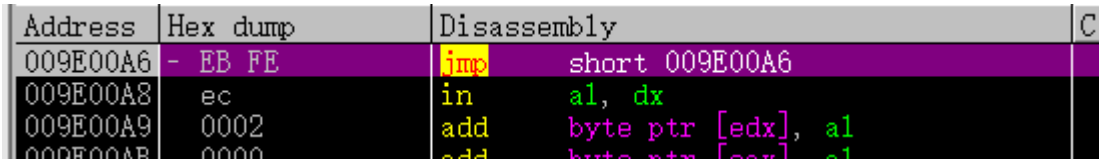
- For debugging the 2nd shellcode, use **ProcessHacker** to change bytes of 2nd shellcode to **0xEB 0xFE** (*must restore to orginal bytes later. The original bytes is **0xFC 0x81***):



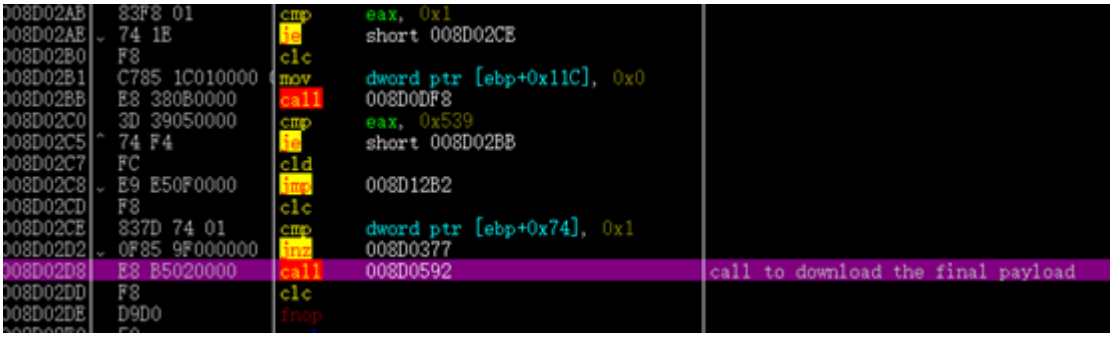
- Let's trace over **ZwResumeThread**:

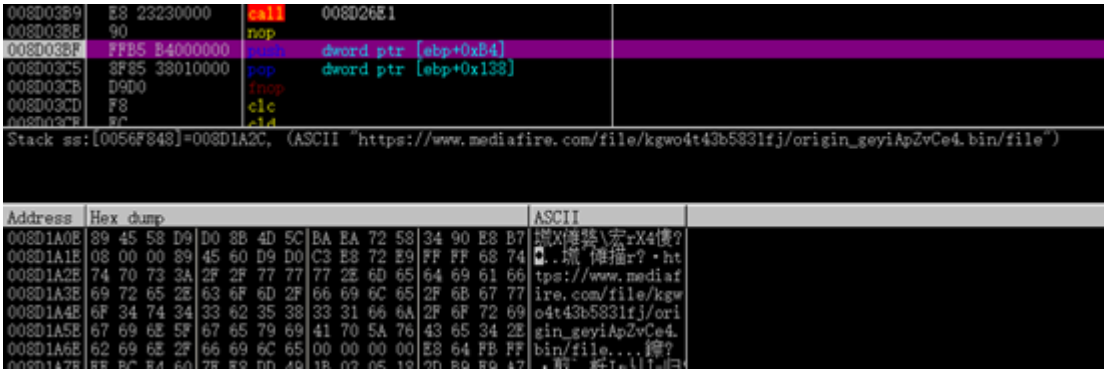


- Open new debugger and attach **RegAsm**. F9 then F12, stop at the **EB FE**. Change back to the original bytes:



- Debug the 2nd shellcode will locate the code decode the URL. For example: *Stack ss: [0056F848]=008D1A2C, (ASCII "hxxps://www.mediafire.com/file/kgwo4t43b5831fj/origin_geyiApZvCe4.bin/file")*





- Sometimes, the mediafire / google drive link was blocked by CloudFlare, so need to manually download and save it. Then let's the shellcode resolve the **wininet_api funcs**, use these apis for downloading the CloudFlare's content. It will check the size of downloaded content is equal to **0x4B600** (in this case). Must patch to let's it think you have downloaded the right binary. Then you trace into the func that will decrypt payload. My trick is replace the CloudFlare content with the content of encrypted payload. Here is the loop it try to find 2 bytes that decrypt 2 bytes of payload to **MZ** signature

```
008D269F   8B55 20        mov     edx, dword ptr [ebp+0x20]
008D26A2   F8             clc
008D26A3   31C9           xor     ecx, ecx                    i = 0x0
008D26A5   D9D0           fnop
008D26A7   90             nop
008D26A8   8B45 64        mov     eax, dword ptr [ebp+0x64]
008D26AB   66:8B9A 400001 mov     bx, word ptr [edx+0x10040]  bx = 6C 61 (content of cloudflare)
008D26B2   FC             cld
008D26B3   66:8B00        mov     ax, word ptr [eax]          ax = 80 D4
008D26B6   90             nop
008D26B7   66:31C8        xor     ax, cx
008D26BA   66:31C3        xor     bx, ax
008D26BD   66:81FB 4D5A   cmp     bx, 0x5A4D                  equal "MZ" header?
008D26C2   74 05          je      short 008D26C9
008D26C4   66:41          inc     cx                          i++
008D26C6   EB DF          jmp     short 008D26A7
```

- Then build the **xor_key_buffer**, buffer length is **0x270** bytes:

```
008D26C9   8B45 64        mov     eax, dword ptr [ebp+0x64]   eax -> xor_key_buf
008D26CC   31DB           xor     ebx, ebx                    j=0
008D26CE   FC             cld
008D26CF   66:310C18      xor     word ptr [eax+ebx], cx      xor_key_buf[j]=xor_key_buf[j] ^ i
008D26D3   81FB 70020000  cmp     ebx, 0x270                  while j < 0x270, continue loop
008D26D9   7D 06          jge     short 008D26E1
008D26DB   83C3 02        add     ebx, 0x2                    j+=2
008D26DE   EB EF          jmp     short 008D26CF
008D26E0   90             nop
```

```
008D1DAC  8A 89 4A 06 8C FB 38 7F 24 79 34 48 C8 8D 80 62  妖J-峻8 $y4H勁 €'
008D1DBC  BD AE 1E 8A 03 20 0D 46 56 9F 08 0F 3E 6E 55 E6  疆? .FV?✡>nU?
008D1DCC  33 8F F2 51 35 01 9D C9 CC C4 DC 93 70 94 29 69  3忿Q5  澤棠較p?i
008D1DDC  65 B4 C7 D4 AB 26 71 4D FE A5 B1 16 A2 B9 FD 30  e辞垣&qM ・??.?
008D1DEC  97 DA 9B 57 DD 4C 45 D0 75 CA 85 99 19 9A D1 B3  複沆輒E袄蕊?氈?
008D1DFC  0E FF 6F DB 54 2D 1A 53 A7 F0 59 1C 4B BF A6 37  ♬・o跿-→S§ðYK喀7
008D1E0C  40 E0 43 5E 86 52 EE 1A 1D 15 2D A0 C1 E5 7A BA  @邸^骂?⊥-牾鎩?
008D1E1C  B6 05 18 25 FC 77 C2 9E 4F F6 02 67 F3 0A 4E 3D  ?↑%黚瀍O?g?N=
008D1E2C  E8 2B A8 A8 2E 59 96 21 82 1B 92 EA 6A EB 22 04  ?è.Y??掙j?┘
008D1E3C  5F 0C 7C 2C 61 7E 6B A4 F8 41 66 6D 9C 10 F7 88  _.|,a~k・Afm?鲌
008D1E4C  91 31 50 AF D7 A3 3F 27 2A 22 3A F1 12 36 87 0B  ?P・?'*":?6?
008D1E5C  07 56 25 32 09 84 13 AB A0 47 0F 74 44 17 5B 8E  •V%2.?珼G✡tD┤[?
008D1E6C  39 38 F9 F9 7F AA E7 72 D3 6C E3 3B 77 3C 2F 11  98・  ・r鬊?w</◀
008D1E7C  6C 5D CD 7D B2 CF BC F5 49 4E B7 BE ED 61 04 95  l]蛚蚕减IN肪韆┘?
008D1E8C  E2 82 A1 00 28 F4 90 78 7B 73 8B 42 1F 43 D8 5C  鈆?(鮀x{s嫻C殺
008D1E9C  14 63 76 83 5A D5 64 FC AD 98 60 C5 95 68 AC DF  ¶cv横誨・榻艏h・
008D1EAC  8A 89 4A 06 8C FB 38 7F 24 79 34 48 C8 8D 80 62  妖J-峻8 $y4H勁 €'
008D1EBC  BD AE 1E 8A 03 20 0D 46 56 9F 08 0F 3E 6E 55 E6  疆? .FV?✡>nU?
008D1ECC  33 8F F2 51 35 01 9D C9 CC C4 DC 93 70 94 29 69  3忿Q5  澤棠較p?i
008D1EDC  65 B4 C7 D4 AB 26 71 4D FE A5 B1 16 A2 B9 FD 30  e辞垣&qM ・??.?
008D1EEC  97 DA 9B 57 DD 4C 45 D0 75 CA 85 99 19 9A D1 B3  複沆輒E袄蕊?氈?
008D1EFC  0E FF 6F DB 54 2D 1A 53 A7 F0 59 1C 4B BF A6 37  ♬・o跿-→S§ðYK喀7
008D1F0C  40 E0 43 5E 86 52 EE 1A 1D 15 2D A0 C1 E5 7A BA  @邸^骂?⊥-牾鎩?
008D1F1C  B6 05 18 25 FC 77 C2 9E 4F F6 02 67 F3 0A 4E 3D  ?↑%黚瀍O?g?N=
008D1F2C  E8 2B A8 A8 2E 59 96 21 82 1B 92 EA 6A EB 22 04  ?è.Y??掙j?┘
008D1F3C  5F 0C 7C 2C 61 7E 6B A4 F8 41 66 6D 9C 10 F7 88  _.|,a~k・Afm?鲌
008D1F4C  91 31 50 AF D7 A3 3F 27 2A 22 3A F1 12 36 87 0B  ?P・?'*":?6?
008D1F5C  07 56 25 32 09 84 13 AB A0 47 0F 74 44 17 5B 8E  •V%2.?珼G✡tD┤[?
008D1F6C  39 38 F9 F9 7F AA E7 72 D3 6C E3 3B 77 3C 2F 11  98・  ・r鬊?w</◀
008D1F7C  6C 5D CD 7D B2 CF BC F5 49 4E B7 BE ED 61 04 95  l]蛚蚕减IN肪韆┘?
008D1F8C  E2 82 A1 00 28 F4 90 78 7B 73 8B 42 1F 43 D8 5C  鈆?(鮀x{s嫻C殺
008D1F9C  14 63 76 83 5A D5 64 FC AD 98 60 C5 95 68 AC DF  ¶cv横誨・榻艏h・
008D1FAC  8A 89 4A 06 8C FB 38 7F 24 79 34 48 C8 8D 80 62  妖J-峻8 $y4H勁 €'
008D1FBC  BD AE 1E 8A 03 20 0D 46 56 9F 08 0F 3E 6E 55 E6  疆? .FV?✡>nU?
008D1FCC  33 8F F2 51 35 01 9D C9 CC C4 DC 93 70 94 29 69  3忿Q5  澤棠較p?i
008D1FDC  65 B4 C7 D4 AB 26 71 4D FE A5 B1 16 A2 B9 FD 30  e辞垣&qM ・??.?
008D1FEC  97 DA 9B 57 DD 4C 45 D0 75 CA 85 99 19 9A D1 B3  複沆輒E袄蕊?氈?
008D1FFC  0E FF 6F DB 54 2D 1A 53 A7 F0 59 1C 4B BF A6 37  ♬・o跿-→S§ðYK喀7
008D200C  40 E0 43 5E 86 52 EE 1A 1D 15 2D A0 C1 E5 7A BA  @邸^骂?⊥-牾鎩?
008D201C  B5 21 49 3F BF 99 49 3F BF 99 49 3F BF 99 49 3F  ?I>廄I>廄I>廄I>
```

- After decrypt loop, get the final payload. It can be a Trojans (RAT) or malware that steals information such as *Agent Tesla, FormBook, NanoCore RAT, Netwire RAT, Remcos RAT, …*

End!

m4n0w4r

Hopefully, in the future, if I have the opportunity to go to Singapore, I will meet him !!

Source: https://kienmanowar.wordpress.com/2020/06/27/quick-analysis-note-about-guloader-or-cloudeye/