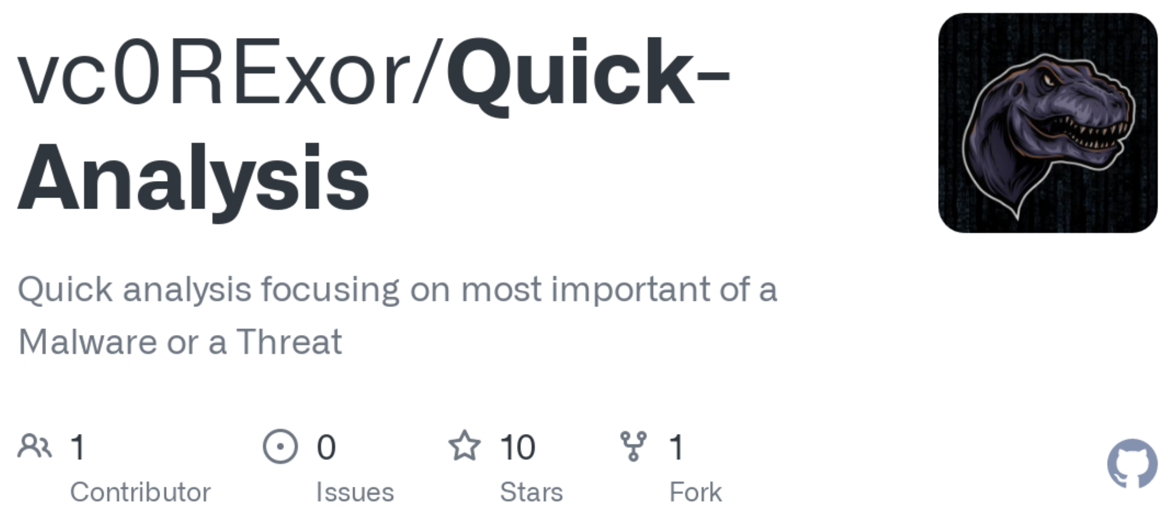
Quick-Analysis/SmokeLoader.md at main · vc0REXor/Quick-Analysis · GitHub

github.com/vc0REXor/Quick-Analysis/blob/main/SmokeLoader/SmokeLoader.md

vc0REXor


vc0REXor/**Quick-Analysis**

Quick analysis focusing on most important of a Malware or a Threat

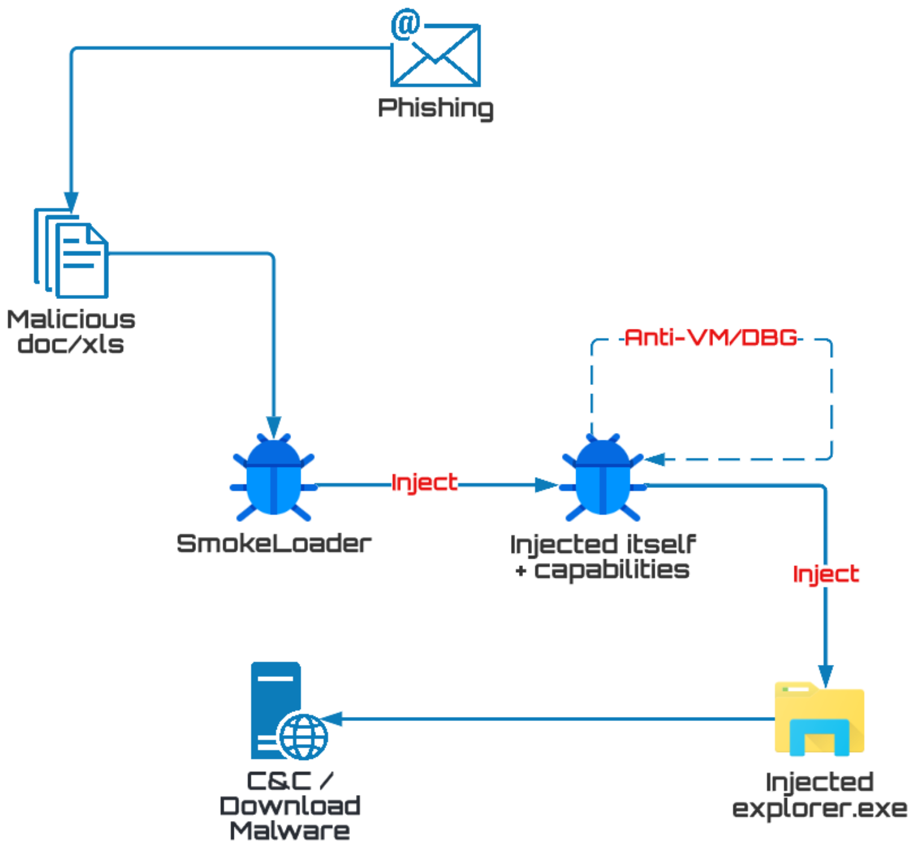
1 Contributor · 0 Issues · 10 Stars · 1 Fork

_Overview

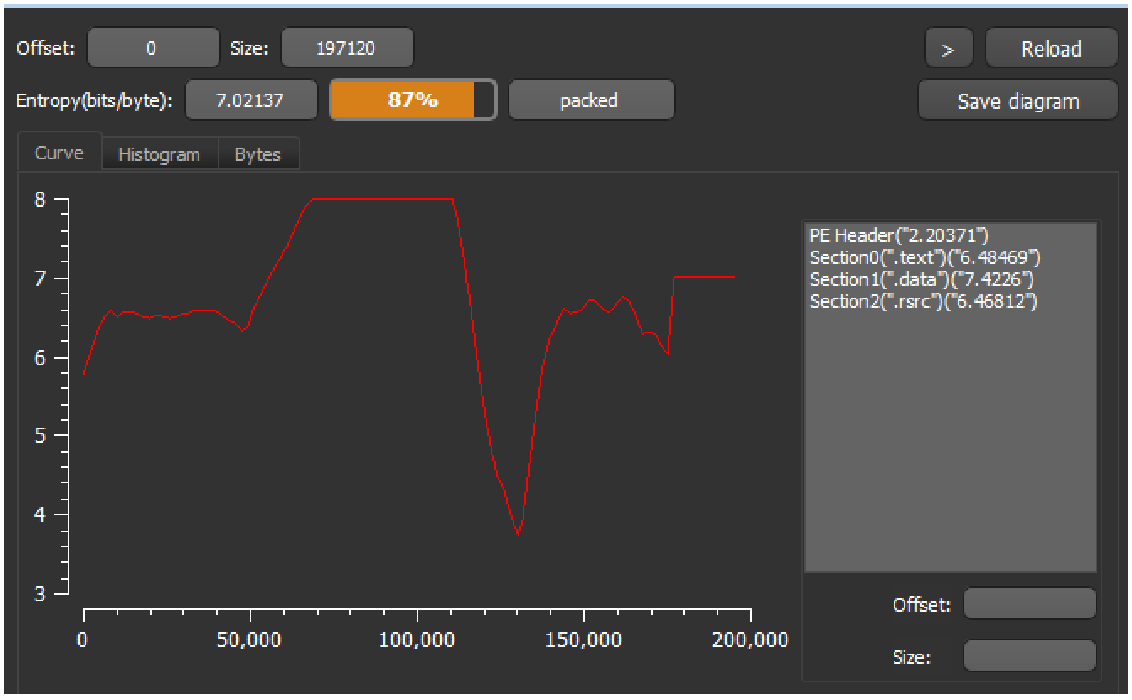
SmokeLoader is a malware that generally acts as a backdoor and is commonly used as a loader for other malware. Attributed to the criminal group Smoky Spider, a group that uses SmokeLoader and Sasfis, loader and downloader respectively. SmokeLoader has been used as a bot in infrastructures and contains strong evasion capabilities as well as Anti-Analysis, Anti-VM and Anti-DBG techniques.

_Technical Analysis

SmokeLoader appears on systems usually through phishing, although it can be loaded by other PUP/PUA or malware. The main execution will revolve around a document that will spawn the SmokeLoader which will run, in most of its versions, a version of itself in a suspended state to inject code, after which it will execute an *explorer.exe* that it will inject again in order to perform the malicious C&C actions or download other files using legitimate software.



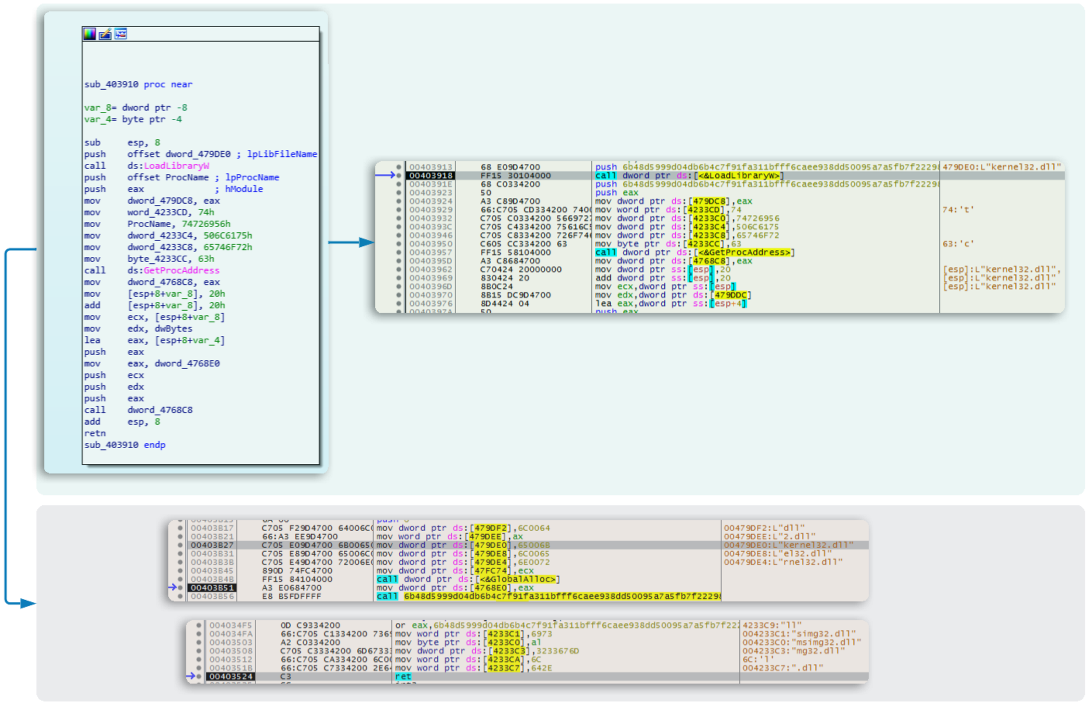
The samples that have been found have in most cases been detected as packed, due to the high level of entropy contained in their sections.



```
PE Header("2.20371")
Section0(".text")("6.48469")
Section1(".data")("7.4226")
Section2(".rsrc")("6.46812")
```

At the initial point, we see how it tries to load libraries in RunTime, something really useful since it prevents us from being able to discern its intentions if we perform a basic static analysis, so it will obtain new functionalities during its execution.

Dll runtime load



```
sub_403910 proc near

var_8= dword ptr -8
var_4= byte ptr -4

sub     esp, 8
push    offset dword_479DE0 ; lpLibFileName
call    ds:LoadLibraryW
push    offset ProcName ; lpProcName
push    eax             ; hModule
mov     dword_479DC8, eax
mov     word_4233CD, 74h
mov     ProcName, 74726956h
mov     dword_4233C4, 506C6175h
mov     dword_4233C8, 65746F72h
mov     byte_4233CC, 63h
call    ds:GetProcAddress
mov     dword_4768C8, eax
mov     [esp+8+var_8], 20h
add     [esp+8+var_8], 20h
mov     ecx, [esp+8+var_8]
mov     edx, dwBytes
lea     eax, [esp+8+var_4]
push    eax
mov     eax, dword_4768E0
push    ecx
push    edx
push    eax
call    dword_4768C8
add     esp, 8
retn
sub_403910 endp
```

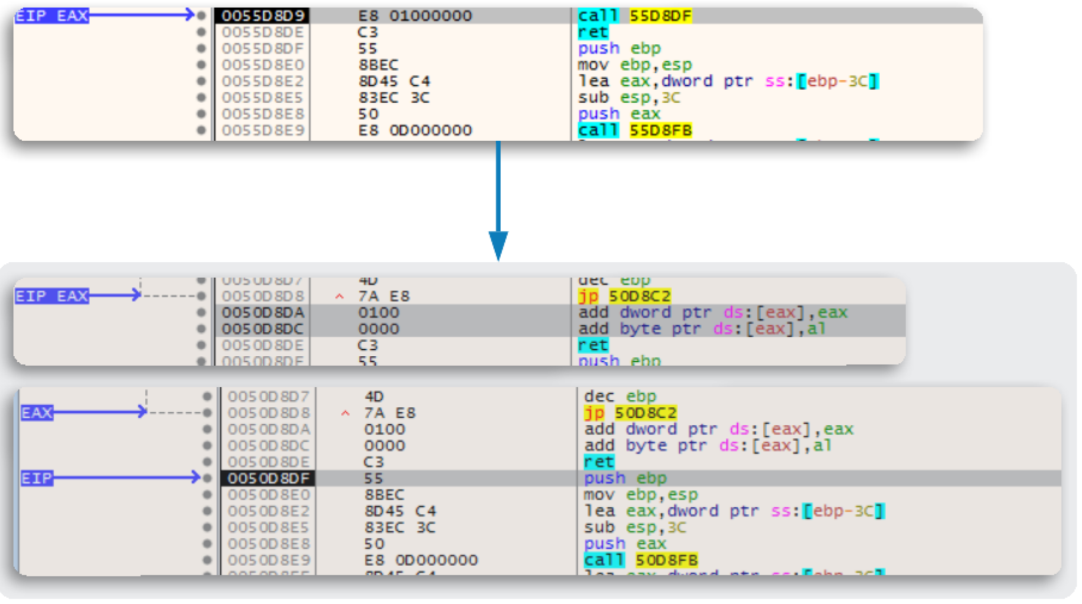
```
00403913  68 E09D4700            push 6b48d5999d04db6b4c7f91fa311bfff6caee938dd50095a7a5fb7f22298  479DE0:L"kernel32.dll"
00403918  FF15 30104000          call dword ptr ds:[<&LoadLibraryW>]
0040391E  68 C0334200            push 6b48d5999d04db6b4c7f91fa311bfff6caee938dd50095a7a5fb7f22298
00403923  50                     push eax
00403924  A3 C89D4700            mov dword ptr ds:[479DC8],eax
00403929  66:C705 CD334200 7400  mov word ptr ds:[4233CD],74                                         74:'t'
00403932  C705 C0334200 5669727  mov dword ptr ds:[4233C0],74726956
0040393C  C705 C4334200 75616C5  mov dword ptr ds:[4233C4],506C6175
00403946  C705 C8334200 726F740  mov dword ptr ds:[4233C8],65746F72
00403950  C605 CC334200 63       mov byte ptr ds:[4233CC],63                                         63:'c'
00403957  FF15 58104000          call dword ptr ds:[<&GetProcAddress>]
0040395D  A3 C8684700            mov dword ptr ds:[4768C8],eax
00403962  C70424 20000000        mov dword ptr ss:[esp],20                                           [esp]:L"kernel32.dll",
00403969  830424 20              add dword ptr ss:[esp],20                                           [esp]:L"kernel32.dll"
0040396D  8B0C24                 mov ecx,dword ptr ss:[esp]                                          [esp]:L"kernel32.dll"
00403970  8B15 DC9D4700          mov edx,dword ptr ds:[479DDC]
00403976  8D4424 04              lea eax,dword ptr ss:[esp+4]
```

```
00403B17  C705 F29D4700 64006C0  mov dword ptr ds:[479DF2],6C0064     00479DF2:L"dll"
00403B21  66:A3 EE9D4700         mov word ptr ds:[479DEE],ax          00479DEE:L"2.dll"
00403B27  C705 E09D4700 6B00650  mov dword ptr ds:[479DE0],65006B     00479DE0:L"kernel32.dll"
00403B31  C705 E89D4700 65006C0  mov dword ptr ds:[479DE8],6C0065     00479DE8:L"el32.dll"
00403B3B  C705 E49D4700 72006E0  mov dword ptr ds:[479DE4],6E0072     00479DE4:L"rnel32.dll"
00403B45  890D 74FC4700          mov dword ptr ds:[47FC74],ecx
00403B4B  FF15 84104000          call dword ptr ds:[<&GlobalAlloc>]
00403B51  A3 E0684700            mov dword ptr ds:[4768E0],eax
00403B56  E8 B5FDFFFF            call 6b48d5999d04db6b4c7f91fa311bfff6caee938dd50095a7a5fb7f22298
```

```
004034F5  0D C9334200             or eax,6b48d5999d04db6b4c7f91fa311bfff6caee938dd50095a7a5fb7f222  4233C9:"ll"
004034FA  66:C705 C1334200 736    mov word ptr ds:[4233C1],6973       004233C1:"simg32.dll"
00403503  A2 C0334200             mov byte ptr ds:[4233C0],al         004233C0:"msimg32.dll"
00403508  C705 C3334200 6D6733    mov dword ptr ds:[4233C3],3233676D  004233C3:"mg32.dll"
00403512  66:C705 CA334200 6C0    mov word ptr ds:[4233CA],6C         6C:'l'
0040351B  66:C705 C7334200 2E6    mov word ptr ds:[4233C7],642E       004233C7:".dll"
00403524  C3                      ret
```

In some of the techniques used to hinder the analysis, such as code obfuscation, we find different hidden calls, as well as abuses of RET to reach calls that we will not see statically.

Hidden calls



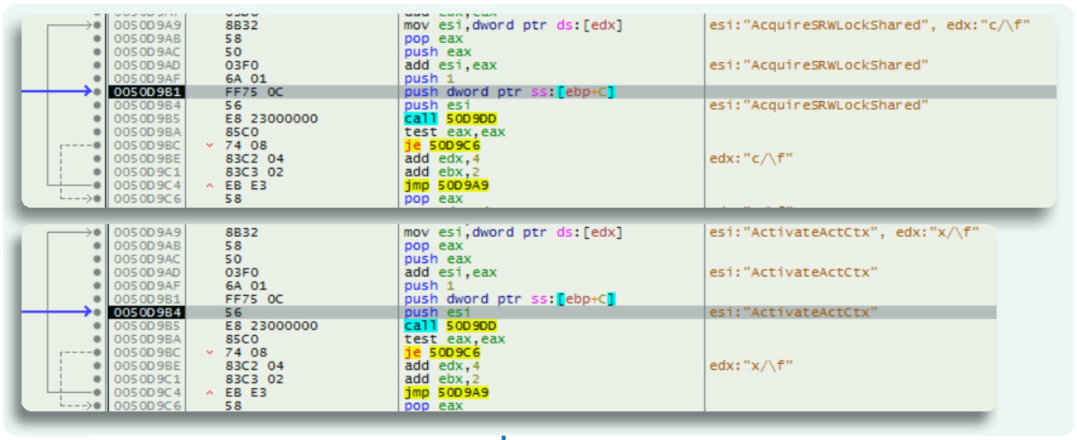
```
EIP EAX ──> 0055D8D9   E8 01000000   call 55D8DF
            0055D8DE   C3            ret
            0055D8DF   55            push ebp
            0055D8E0   8BEC          mov ebp,esp
            0055D8E2   8D45 C4       lea eax,dword ptr ss:[ebp-3C]
            0055D8E5   83EC 3C       sub esp,3C
            0055D8E8   50            push eax
            0055D8E9   E8 0D000000   call 55D8FB
```

```
            0050D8D7   4D            dec ebp
EIP EAX ──> 0050D8D8 ^ 7A E8         jp 50D8C2
            0050D8DA   0100          add dword ptr ds:[eax],eax
            0050D8DC   0000          add byte ptr ds:[eax],al
            0050D8DE   C3            ret
            0050D8DF   55            push ebp
```

```
            0050D8D7   4D            dec ebp
EAX ──────> 0050D8D8 ^ 7A E8         jp 50D8C2
            0050D8DA   0100          add dword ptr ds:[eax],eax
            0050D8DC   0000          add byte ptr ds:[eax],al
            0050D8DE   C3            ret
EIP ──────> 0050D8DF   55            push ebp
            0050D8E0   8BEC          mov ebp,esp
            0050D8E2   8D45 C4       lea eax,dword ptr ss:[ebp-3C]
            0050D8E5   83EC 3C       sub esp,3C
            0050D8E8   50            push eax
            0050D8E9   E8 0D000000   call 50D8FB
```

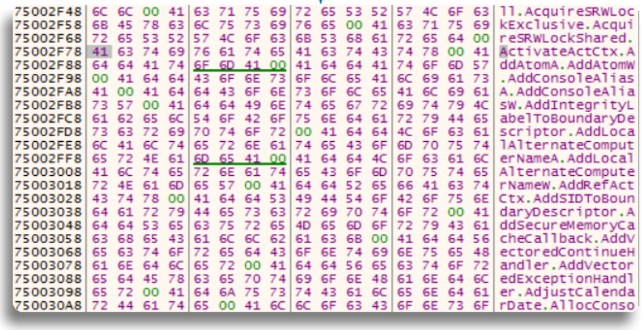
As mentioned above, it fetches libraries during runtime and is dedicated to resolving APIs that it could use later on

Resolving APIs



```
0050D9A9   8B32          mov esi,dword ptr ds:[edx]       esi:"AcquireSRWLockShared", edx:"c/\f"
0050D9AB   58            pop eax
0050D9AC   50            push eax
0050D9AD   03F0          add esi,eax                      esi:"AcquireSRWLockShared"
0050D9AF   6A 01         push 1
0050D9B1   FF75 0C       push dword ptr ss:[ebp+C]
0050D9B4   56            push esi                         esi:"AcquireSRWLockShared"
0050D9B5   E8 23000000   call 50D9DD
0050D9BA   85C0          test eax,eax
0050D9BC   74 08         je 50D9C6
0050D9BE   83C2 04       add edx,4                        edx:"c/\f"
0050D9C1   83C3 02       add ebx,2
0050D9C4   EB E3         jmp 50D9A9
0050D9C6   58            pop eax
```

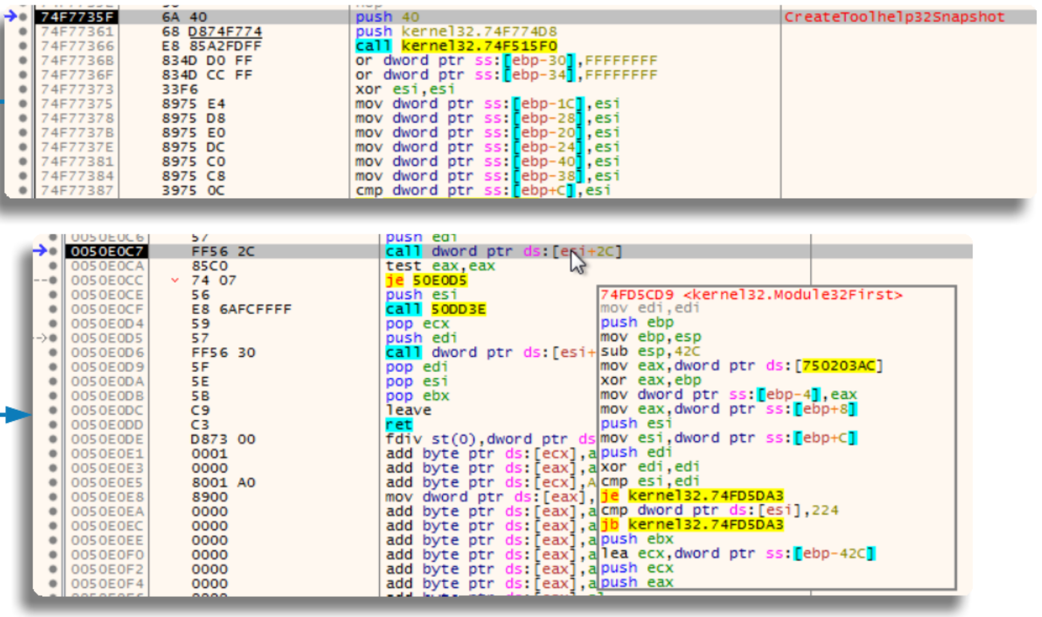
```
0050D9A9   8B32          mov esi,dword ptr ds:[edx]       esi:"ActivateActCtx", edx:"x/\f"
0050D9AB   58            pop eax
0050D9AC   50            push eax
0050D9AD   03F0          add esi,eax                      esi:"ActivateActCtx"
0050D9AF   6A 01         push 1
0050D9B1   FF75 0C       push dword ptr ss:[ebp+C]
0050D9B4   56            push esi                         esi:"ActivateActCtx"
0050D9B5   E8 23000000   call 50D9DD
0050D9BA   85C0          test eax,eax
0050D9BC   74 08         je 50D9C6
0050D9BE   83C2 04       add edx,4                        edx:"x/\f"
0050D9C1   83C3 02       add ebx,2
0050D9C4   EB E3         jmp 50D9A9
0050D9C6   58            pop eax
```



```
75002F48  6C 6C 00 41 63 71 75 69 72 65 53 52 57 4C 6F 63  ll.AcquireSRWLoc
75002F58  6B 45 78 63 6C 75 73 69 76 65 00 41 63 71 75 69  kExclusive.Acqui
75002F68  72 65 53 52 57 4C 6F 63 6B 53 68 61 72 65 64 00  reSRWLockShared.
75002F78  41 63 74 69 76 61 74 65 41 63 74 43 74 78 00 41  ActivateActCtx.A
75002F88  64 64 41 74 6F 6D 41 00 41 64 64 41 74 6F 6D 57  ddAtomA.AddAtomW
75002F98  00 41 64 64 43 6F 6E 73 6F 6C 65 41 6C 69 61 73  .AddConsoleAlias
75002FA8  41 00 41 64 64 43 6F 6E 73 6F 6C 65 41 6C 69 61  A.AddConsoleAlia
75002FB8  73 57 00 41 64 64 49 6E 74 65 67 72 69 74 79 4C  sW.AddIntegrityL
75002FC8  61 62 65 6C 54 6F 42 6F 75 6E 64 61 72 79 44 65  abelToBoundaryDe
75002FD8  73 63 72 69 70 74 6F 72 00 41 64 64 4C 6F 63 61  scriptor.AddLoca
75002FE8  6C 41 6C 74 65 72 6E 61 74 65 43 6F 6D 70 75 74  lAlternateComput
75002FF8  65 72 4E 61 6D 65 41 00 41 64 64 4C 6F 63 61 6C  erNameA.AddLocal
75003008  41 6C 74 65 72 6E 61 74 65 43 6F 6D 70 75 74 65  AlternateCompute
75003018  72 4E 61 6D 65 57 00 41 64 64 52 65 66 41 63 74  rNameW.AddRefAct
75003028  43 74 78 00 41 64 64 53 49 44 54 6F 42 6F 75 6E  Ctx.AddSIDToBoun
75003038  64 61 72 79 44 65 73 63 72 69 70 74 6F 72 00 41  daryDescriptor.A
75003048  64 64 53 65 63 75 72 65 4D 65 6D 6F 72 79 43 61  ddSecureMemoryCa
75003058  63 68 65 43 61 6C 6C 62 61 63 6B 00 41 64 64 56  cheCallback.AddV
75003068  65 63 74 6F 72 65 64 43 6F 6E 74 69 6E 75 65 48  ectoredContinueH
75003078  61 6E 64 6C 65 72 00 41 64 64 56 65 63 74 6F 72  andler.AddVector
75003088  65 64 45 78 63 65 70 74 69 6F 6E 48 61 6E 64 6C  edExceptionHandl
75003098  65 72 00 41 64 6A 75 73 74 43 61 6C 65 6E 64 61  er.AdjustCalenda
750030A8  72 44 61 74 65 00 41 6C 6C 6F 63 43 6F 6E 73 6F  rDate.AllocConso
```

At all times, it has control over what is running on the machine, as it subsequently performs various Anti-Vm and Anti-dbg techniques, so having all running processes mapped is always a good technique.

Control of running processes



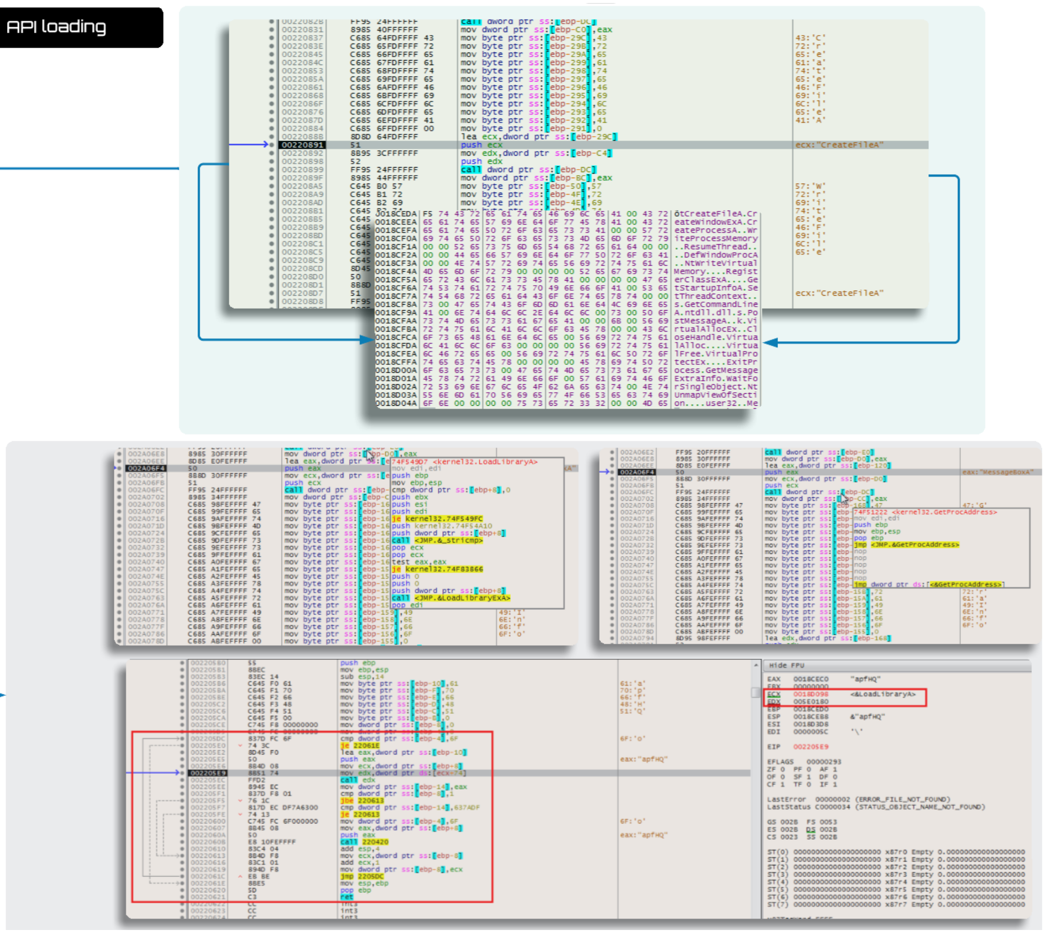
```
74F7735F   6A 40          push 40                              CreateToolhelp32Snapshot
74F77361   68 D874F774    push kernel32.74F774D8
74F77366   E8 85A2FDFF    call kernel32.74F515F0
74F7736B   834D D0 FF     or dword ptr ss:[ebp-30],FFFFFFFF
74F7736F   834D CC FF     or dword ptr ss:[ebp-34],FFFFFFFF
74F77373   33F6           xor esi,esi
74F77375   8975 E4        mov dword ptr ss:[ebp-1C],esi
74F77378   8975 D8        mov dword ptr ss:[ebp-28],esi
74F7737B   8975 E0        mov dword ptr ss:[ebp-20],esi
74F7737E   8975 DC        mov dword ptr ss:[ebp-24],esi
74F77381   8975 C0        mov dword ptr ss:[ebp-40],esi
74F77384   8975 C8        mov dword ptr ss:[ebp-38],esi
74F77387   3975 0C        cmp dword ptr ss:[ebp+C],esi
```

```
0050E0C7   FF56 2C        call dword ptr ds:[esi+2C]
0050E0CA   85C0           test eax,eax
0050E0CC   74 07          je 50E0D5
0050E0CE   56             push esi
0050E0CF   E8 6AFCFFFF    call 50DD3E
0050E0D4   59             pop ecx
0050E0D5   57             push edi
0050E0D6   FF56 30        call dword ptr ds:[esi+
0050E0D9   5F             pop edi
0050E0DA   5E             pop esi
0050E0DB   5B             pop ebx
0050E0DC   C9             leave
0050E0DD   C3             ret
0050E0DE   D873 00        fdiv st(0),dword ptr ds
0050E0E1   0001           add byte ptr ds:[ecx],a
0050E0E3   0000           add byte ptr ds:[eax],a
0050E0E5   8001 A0        add byte ptr ds:[ecx],A
0050E0E8   8900           mov dword ptr ds:[eax],
0050E0EA   0000           add byte ptr ds:[eax],a
0050E0EC   0000           add byte ptr ds:[eax],a
0050E0EE   0000           add byte ptr ds:[eax],a
0050E0F0   0000           add byte ptr ds:[eax],a
0050E0F2   0000           add byte ptr ds:[eax],a
0050E0F4   0000           add byte ptr ds:[eax],a
```

```
74FD5CD9 <kernel32.Module32First>
mov edi,edi
push ebp
mov ebp,esp
sub esp,42C
mov eax,dword ptr ds:[750203AC]
xor eax,ebp
mov dword ptr ss:[ebp-4],eax
mov eax,dword ptr ss:[ebp+8]
push esi
mov esi,dword ptr ss:[ebp+C]
push edi
xor edi,edi
cmp esi,edi
je kernel32.74FD5DA3
cmp dword ptr ds:[esi],224
jb kernel32.74FD5DA3
push ebx
lea ecx,dword ptr ss:[ebp-42C]
push ecx
push eax
```

After this, it starts loading APIs that will serve it moments later, in which we will see a routine that will be loading from memory and using LoadLibrary + GetProcAddress

API loading



```
00220831   8985 40FFFFFF       mov dword ptr ss:[ebp-C0],eax
00220837   C685 64FDFFFF 43    mov byte ptr ss:[ebp-29C],43      43:'C'
0022083E   C685 65FDFFFF 72    mov byte ptr ss:[ebp-29B],72      72:'r'
00220845   C685 66FDFFFF 65    mov byte ptr ss:[ebp-29A],65      65:'e'
0022084C   C685 67FDFFFF 61    mov byte ptr ss:[ebp-299],61      61:'a'
00220853   C685 68FDFFFF 74    mov byte ptr ss:[ebp-298],74      74:'t'
0022085A   C685 69FDFFFF 65    mov byte ptr ss:[ebp-297],65      65:'e'
00220861   C685 6AFDFFFF 46    mov byte ptr ss:[ebp-296],46      46:'F'
00220868   C685 6BFDFFFF 69    mov byte ptr ss:[ebp-295],69      69:'i'
0022086F   C685 6CFDFFFF 6C    mov byte ptr ss:[ebp-294],6C      6C:'l'
00220876   C685 6DFDFFFF 65    mov byte ptr ss:[ebp-293],65      65:'e'
0022087D   C685 6EFDFFFF 41    mov byte ptr ss:[ebp-292],41      41:'A'
00220884   C685 6FFDFFFF 00    mov byte ptr ss:[ebp-291],0
0022088B   8D8D 64FDFFFF       lea ecx,dword ptr ss:[ebp-29C]
00220891   51                  push ecx                          ecx:"CreateFileA"
00220892   8B95 3CFFFFFF       mov edx,dword ptr ss:[ebp-C4]
00220898   52                  push edx
00220899   FF95 24FFFFFF       call dword ptr ss:[ebp-DC]
0022089F   8985 44FFFFFF       mov dword ptr ss:[ebp-BC],eax
002208A5   C645 B0 57          mov byte ptr ss:[ebp-50],57       57:'W'
002208A9   C645 B1 72          mov byte ptr ss:[ebp-4F],72       72:'r'
002208AD   C645 B2 69          mov byte ptr ss:[ebp-4E],69       69:'i'
```

```
0018CEDA  F5 74 43 72 65 61 74 65 46 69 6C 65 41 00 43 72  õtCreateFileA.Cr
0018CEEA  65 61 74 65 57 69 6E 64 6F 77 45 78 41 00 43 72  eateWindowExA.Cr
0018CEFA  65 61 74 65 50 72 6F 63 65 73 73 41 00 00 57 72  eateProcessA..Wr
0018CF0A  69 74 65 50 72 6F 63 65 73 73 4D 65 6D 6F 72 79  iteProcessMemory
0018CF1A  00 00 52 65 73 75 6D 65 54 68 72 65 61 64 00 00  ..ResumeThread..
0018CF2A  00 00 44 65 66 57 69 6E 64 6F 77 50 72 6F 63 41  ..DefWindowProcA
0018CF3A  00 00 4E 74 57 72 69 74 65 56 69 72 74 75 61 6C  ..NtWriteVirtual
0018CF4A  4D 65 6D 6F 72 79 00 00 00 00 52 65 67 69 73 74  Memory....Regist
0018CF5A  65 72 43 6C 61 73 73 45 78 41 00 00 00 00 47 65  erClassExA....Ge
0018CF6A  74 53 74 61 72 74 75 70 49 6E 66 6F 41 00 53 65  tStartupInfoA.Se
0018CF7A  74 54 68 72 65 61 64 43 6F 6E 74 65 78 74 00 00  tThreadContext..
0018CF8A  73 00 47 65 74 43 6F 6D 6D 61 6E 64 4C 69 6E 65  s.GetCommandLine
0018CF9A  41 00 6E 74 64 6C 6C 2E 64 6C 6C 00 73 00 50 6F  A.ntdll.dll.s.Po
0018CFAA  73 74 4D 65 73 73 61 67 65 41 00 00 6B 00 56 69  stMessageA..k.Vi
0018CFBA  72 74 75 61 6C 41 6C 6C 6F 63 45 78 00 00 43 6C  rtualAllocEx..Cl
0018CFCA  6F 73 65 48 61 6E 64 6C 65 00 56 69 72 74 75 61  oseHandle.Virtua
0018CFDA  6C 41 6C 6C 6F 63 00 00 00 00 56 69 72 74 75 61  lAlloc....Virtua
0018CFEA  6C 46 72 65 65 00 56 69 72 74 75 61 6C 50 72 6F  lFree.VirtualPro
0018CFFA  74 65 63 74 45 78 00 00 00 00 45 78 69 74 50 72  tectEx....ExitPr
0018D00A  6F 63 65 73 73 00 47 65 74 4D 65 73 73 61 67 65  ocess.GetMessage
0018D01A  45 78 74 72 61 49 6E 66 6F 00 57 61 69 74 46 6F  ExtraInfo.WaitFo
0018D02A  72 53 69 6E 67 6C 65 4F 62 6A 65 63 74 00 4E 74  rSingleObject.Nt
0018D03A  55 6E 6D 61 70 56 69 65 77 4F 66 53 65 63 74 69  UnmapViewOfSecti
0018D04A  6F 6E 00 00 00 00 75 73 65 72 33 32 00 00 4D 65  on....user32..Me
```

```
002A06E2   FF95 20FFFFFF       call dword ptr ss:[ebp-E0]
002A06E8   8985 30FFFFFF       mov dword ptr ss:[ebp-D0],eax
002A06EE   8D85 E0FEFFFF       lea eax,dword ptr ss:[ebp-120]
002A06F4   50                  push eax                          eax:"MessageBoxA"
002A06F5   8B8D 30FFFFFF       mov ecx,dword ptr ss:[ebp-D0]
002A06FB   51                  push ecx
002A06FC   FF95 24FFFFFF       call dword ptr ss:[ebp-DC]
002A0702   8985 34FFFFFF       mov dword ptr ss:[ebp-CC],eax
002A0708   C685 98FEFFFF 47    mov byte ptr ss:[ebp-168],47      47:'G'
002A070F   C685 99FEFFFF 65    mov byte ptr ss:[ebp-16
002A0716   C685 9AFEFFFF 74    mov byte ptr ss:[ebp-16
002A071D   C685 9BFEFFFF 4D    mov byte ptr ss:[ebp-16
002A0724   C685 9CFEFFFF 65    mov byte ptr ss:[ebp-16
002A072B   C685 9DFEFFFF 73    mov byte ptr ss:[ebp-16
002A0732   C685 9EFEFFFF 73    mov byte ptr ss:[ebp-16
002A0739   C685 9FFEFFFF 61    mov byte ptr ss:[ebp-16
002A0740   C685 A0FEFFFF 67    mov byte ptr ss:[ebp-16
002A0747   C685 A1FEFFFF 65    mov byte ptr ss:[ebp-15
002A074E   C685 A2FEFFFF 45    mov byte ptr ss:[ebp-15
002A0755   C685 A3FEFFFF 78    mov byte ptr ss:[ebp-15
002A075C   C685 A4FEFFFF 74    mov byte ptr ss:[ebp-15
002A0763   C685 A5FEFFFF 72    mov byte ptr ss:[ebp-15B],72      72:'r'
002A076A   C685 A6FEFFFF 61    mov byte ptr ss:[ebp-15A],61      61:'a'
002A0771   C685 A7FEFFFF 49    mov byte ptr ss:[ebp-159],49      49:'I'
002A0778   C685 A8FEFFFF 6E    mov byte ptr ss:[ebp-158],6E      6E:'n'
002A077F   C685 A9FEFFFF 66    mov byte ptr ss:[ebp-157],66      66:'f'
002A0786   C685 AAFEFFFF 6F    mov byte ptr ss:[ebp-156],6F      6F:'o'
002A078D   C685 ABFEFFFF 00    mov byte ptr ss:[ebp-155],0
002A0794   8D95 98FEFFFF       lea edx,dword ptr ss:[ebp-168]
```

```
74F549D7 <kernel32.LoadLibraryA>
mov edi,edi
push ebp
mov ebp,esp
cmp dword ptr ss:[ebp+8],0
push ebx
push esi
push edi
je kernel32.74F549FC
push kernel32.74F54A10
push dword ptr ss:[ebp+8]
call <JMP.&_stricmp>
pop ecx
pop ecx
test eax,eax
je kernel32.74F83866
push 0
push 0
push dword ptr ss:[ebp+8]
call <JMP.&LoadLibraryExA>
pop edi
```

```
74F51222 <kernel32.GetProcAddress>
mov edi,edi
push ebp
mov ebp,esp
pop ebp
jmp <JMP.&GetProcAddress>
nop
nop
nop
nop
nop
jmp dword ptr ds:[<&GetProcAddress>]
```

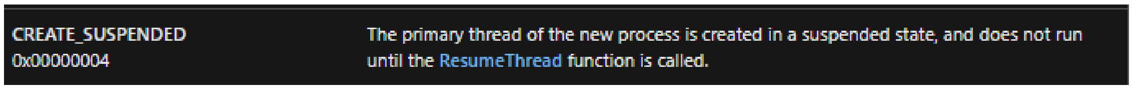
```
002205B0   55                  push ebp
002205B1   8BEC                mov ebp,esp
002205B3   83EC 14             sub esp,14
002205B6   C645 F0 61          mov byte ptr ss:[ebp-10],61       61:'a'
002205BA   C645 F1 70          mov byte ptr ss:[ebp-F],70        70:'p'
002205BE   C645 F2 66          mov byte ptr ss:[ebp-E],66        66:'f'
002205C2   C645 F3 48          mov byte ptr ss:[ebp-D],48        48:'H'
002205C6   C645 F4 51          mov byte ptr ss:[ebp-C],51        51:'Q'
002205CA   C645 F5 00          mov byte ptr ss:[ebp-B],0
002205CE   C745 F8 00000000    mov dword ptr ss:[ebp-8],0
002205D5   C745 FC 00000000    mov dword ptr ss:[ebp-4],0
002205DC   837D FC 6F          cmp dword ptr ss:[ebp-4],6F       6F:'o'
002205E0   74 3C               je 22061E
002205E2   8D45 F0             lea eax,dword ptr ss:[ebp-10]
002205E5   50                  push eax                          eax:"apfHQ"
002205E6   8B4D 08             mov ecx,dword ptr ss:[ebp+8]
002205E9   8B51 74             mov edx,dword ptr ds:[ecx+74]
002205EC   FFD2                call edx
002205EE   8945 EC             mov dword ptr ss:[ebp-14],eax
002205F1   837D F8 01          cmp dword ptr ss:[ebp-8],1
002205F5   76 1C               jbe 220613
002205F7   817D EC DF7A6300    cmp dword ptr ss:[ebp-14],637ADF
002205FE   74 13               je 220613
00220600   C745 FC 6F000000    mov dword ptr ss:[ebp-4],6F       6F:'o'
00220607   8B45 08             mov eax,dword ptr ss:[ebp+8]
0022060A   50                  push eax                          eax:"apfHQ"
0022060B   E8 10FEFFFF         call 220420
00220610   83C4 04             add esp,4
00220613   8B4D F8             mov ecx,dword ptr ss:[ebp-8]
00220616   83C1 01             add ecx,1
00220619   894D F8             mov dword ptr ss:[ebp-8],ecx
0022061C   EB BE               jmp 2205DC
0022061E   8BE5                mov esp,ebp
00220620   5D                  pop ebp
00220621   C3                  ret
00220622   CC                  int3
00220623   CC                  int3
```

```
Hide FPU

EAX    0018CEC0    "apfHQ"
EBX    00000000
ECX    0018D098    <&LoadLibraryA>
EDX    005E0180
EBP    0018CED0
ESP    0018CEB8    &"apfHQ"
ESI    0018D3D8
EDI    0000005C    '\'

EIP    002205E9

EFLAGS   00000293
ZF 0  PF 0  AF 1
OF 0  SF 1  DF 0
CF 1  TF 0  IF 1

LastError  00000002 (ERROR_FILE_NOT_FOUND)
LastStatus C0000034 (STATUS_OBJECT_NAME_NOT_FOUND)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 0000000000000000000 x87r0 Empty 0.000000000000000000
ST(1) 0000000000000000000 x87r1 Empty 0.000000000000000000
ST(2) 0000000000000000000 x87r2 Empty 0.000000000000000000
ST(3) 0000000000000000000 x87r3 Empty 0.000000000000000000
ST(4) 0000000000000000000 x87r4 Empty 0.000000000000000000
ST(5) 0000000000000000000 x87r5 Empty 0.000000000000000000
ST(6) 0000000000000000000 x87r6 Empty 0.000000000000000000
ST(7) 0000000000000000000 x87r7 Empty 0.000000000000000000
```

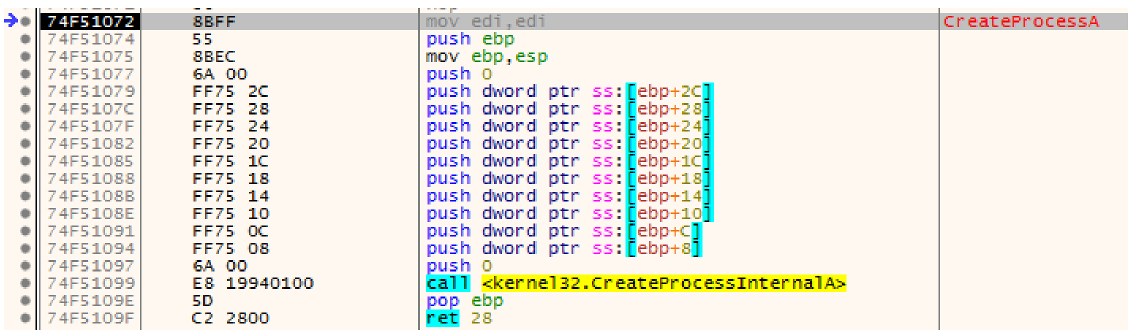
APIs:

```
CreateFileA
CreateWindowExA
CreateProcessA
WriteProcessMemory
ResumeThread
DefWindowProcA
NtWriteVirtualMemory
RegisterClassExA
GetStartupInfoA
SetThreadContext
GetCommandLineA
PostMessageA
VirtualAllocEx
CloseHandle
VirtualAlloc
VirtualFree
VirtualProtectEx
ExitProcess
GetMessageExtraInfo
WaitForSingleObject
NtUnmapViewOfSection
MessageBoxA
ReadProcessMemory
GetThreadContext
WriteFile
GetModuleFileNameA
GetFileAttributesA
WinExec
GetMessageA
```

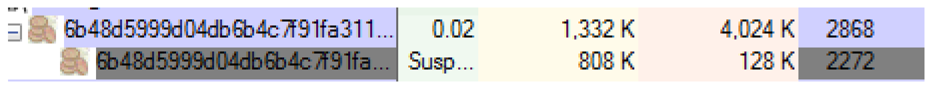
Once it has the libraries, APIs and processes controlled, it creates a process in suspended state, for this it uses CreateProcessInternalA that will call CreateProcessInternalW entering 0x04 in dwCreationflags to create the process in suspended state.



CREATE_SUSPENDED 0x00000004	The primary thread of the new process is created in a suspended state, and does not run until the ResumeThread function is called.

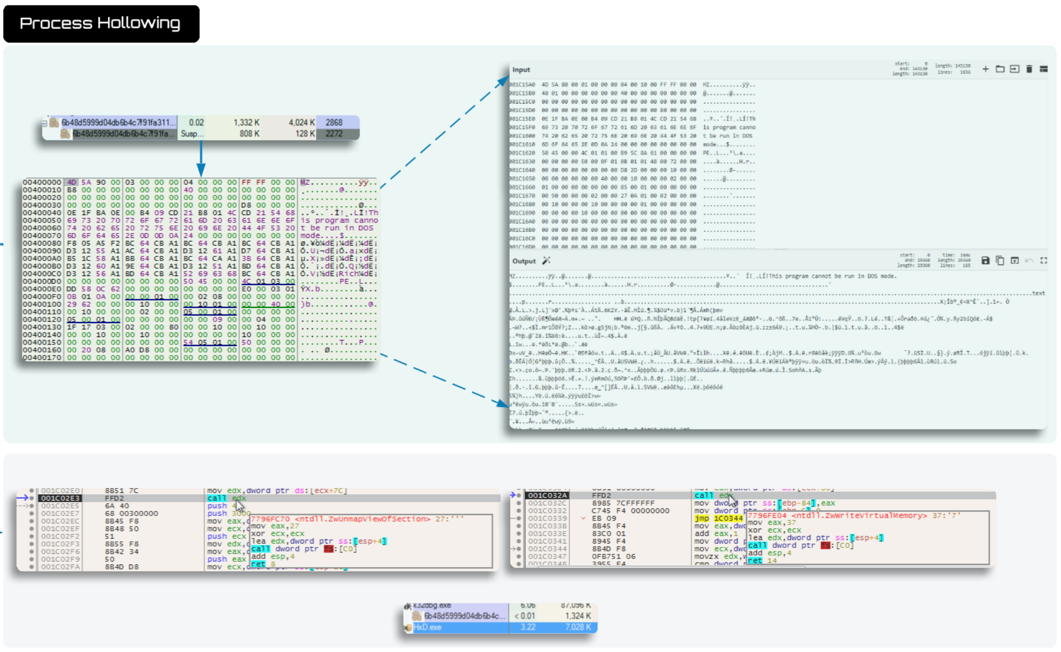


```
74F51072   8BFF            mov edi,edi                              CreateProcessA
74F51074   55              push ebp
74F51075   8BEC            mov ebp,esp
74F51077   6A 00           push 0
74F51079   FF75 2C         push dword ptr ss:[ebp+2C]
74F5107C   FF75 28         push dword ptr ss:[ebp+28]
74F5107F   FF75 24         push dword ptr ss:[ebp+24]
74F51082   FF75 20         push dword ptr ss:[ebp+20]
74F51085   FF75 1C         push dword ptr ss:[ebp+1C]
74F51088   FF75 18         push dword ptr ss:[ebp+18]
74F5108B   FF75 14         push dword ptr ss:[ebp+14]
74F5108E   FF75 10         push dword ptr ss:[ebp+10]
74F51091   FF75 0C         push dword ptr ss:[ebp+C]
74F51094   FF75 08         push dword ptr ss:[ebp+8]
74F51097   6A 00           push 0
74F51099   E8 19940100     call <kernel32.CreateProcessInternalA>
74F5109E   5D              pop ebp
74F5109F   C2 2800         ret 28
```



6b48d5999d04db6b4c7f91fa311...	0.02	1,332 K	4,024 K	2868
6b48d5999d04db6b4c7f91fa...	Susp...	808 K	128 K	2272

Once the process is created in a suspended state, it proceeds to introduce the binary inside the previously spawned process, which, through ProcessHollowing, will unmap data from itself, to write the binary inside, this is usually done through ZwUnmapViewOfSection + VirtualAlloc + ZwWriteVirtualMemory, once introduced into the memory of the process in suspension, it will stop being suspended and will execute it, so the memory file will be detonated.



[The binary extracted from memory, which will inject explorer.exe, is very interesting, we will follow soon :) 🕵️]

_IOC

_SHA256

```
Ebdebba349aba676e9739df18c503ab8c16c7fa1b853fd183f0a005c0e4f68ae
D618d086cdfc61b69e6d93a13cea06e98ac2ad7d846f044990f2ce8305fe8d1b
Ee8f0ff6b0ee6072a30d45c135228108d4c032807810006ec77f2bf72856e04a
6b48d5999d04db6b4c7f91fa311bfff6caee938dd50095a7a5fb7f222987efa3
B961d6795d7ceb3ea3cd00e037460958776a39747c8f03783d458b38daec8025
02083f46860f1ad11e62b2b5f601a86406f7ee3c456e6699ee2912c5d1d89cb9
059d615ce6dee655959d7feae7b70f3b7c806f3986deb1826d01a07aec5a39cf
5318751b75d8c6152d90bbbf2864558626783f497443d4be1a003b64bc2acbc2
79ae89733257378139cf3bdce3a30802818ca1a12bb2343e0b9d0f51f8af1f10
F92523fa104575e0605f90ce4a75a95204bc8af656c27a04aa26782cb64d938d
```

_IP

```
216.128.137.31
8.209.71.53
```

_Domains

```
host-file-host6[.]com
host-host-file8[.]com
fiskahlilian16[.]top
paishancho17[.]top
ydiannetter18[.]top
azarehanelle19[.]top
quericeriant20[.]top
xpowebs[.]ga
venis[.]ml
tootoo[.]ga
eyecosl[.]ga
bullions[.]tk
mizangs[.]tw
mbologwuholing[.]co[.]ug
quadoil[.]ru
```

> 🦖 vc0=Rexor 🕵️