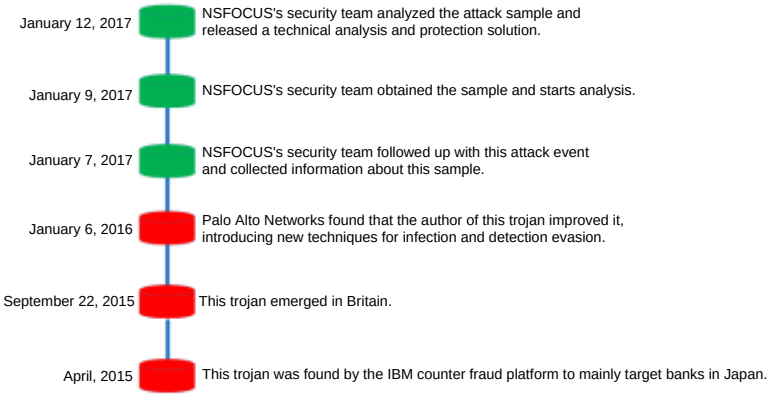
"Shifu" Banking Trojan Technical Analysis and Solution

Release Date: January 16, 2017

Overview

The banking trojan "Shifu" was discovered by the IBM counter fraud platform in April, 2015. Built based on the Shiz source code, this trojan employs techniques adopted by multiple notorious trojans such as Zeus, Gozi, and Dridex. This kind of trojan once mainly targeted 14 banks in Japan and has emerged in Britain since September 22, 2015, attacking at least more than 10 banks. On January 6, 2017, researchcenter.paloaltonetworks.com issued an article, indicating that the author of this trojan refined it in 2016. Specifically, this trojan, at its early stage, obtained system privileges of the attacked host by exploiting the vulnerability assigned CVE-2015-0003, but now achieves this purpose by leveraging the Windows privilege escalation vulnerability assigned CVE-2016-0167.
The sample analyzed in this document is a variant of the "Shifu" trojan. With privileges escalated to the system level by using the embedded system vulnerability

exploitation module, this trojan steals users' login credentials of the online banking business to cause damage.
The following figure shows the timeline of attacks launched via this trojan.



CVE-2016-0167

Microsoft Windows is a series of operating systems developed by Microsoft. win32k.sys is a kernel-mode device driver and is the kernel part of the Windows subsystem. It contains the window manager which controls window displays, as well as manages screen output.
The kernel-mode device driver contains a privilege escalation vulnerability because it does not properly handle objects in memory. An attacker could exploit this vulnerability to escalate his/her privileges and execute arbitrary code.
The following systems are affected:

- Microsoft Windows Vista SP2
- Windows Server 2008 SP2 and R2 SP1
- Windows 7 SP1
- Windows 8.1
- Windows Server 2012 Gold and R2
- Windows RT 8.1
- Windows 10 Gold and 1511

Propagation and Infection

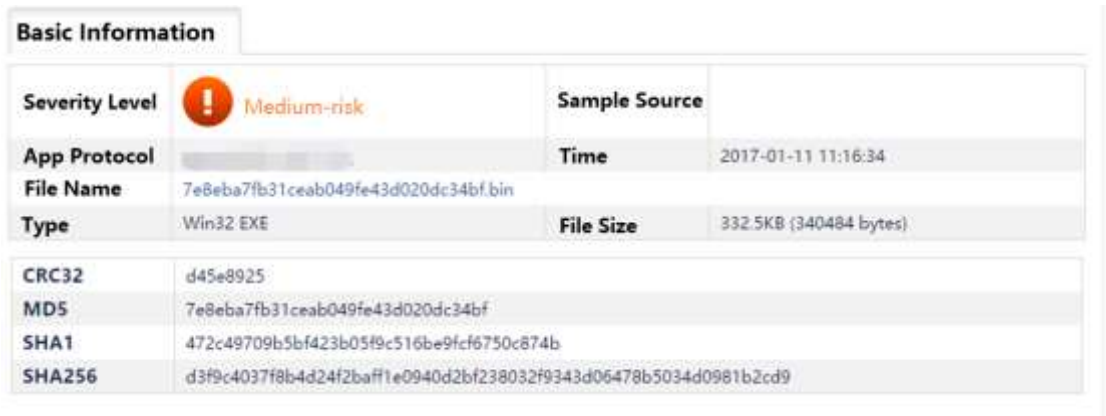
- File binding
- Email attachment

Sample Analysis

(1) Environment

System	Windows 7 (32-bit)
Tools	ProcessMonitor, Xuetr, Wireshark, OllyDbg, IDA

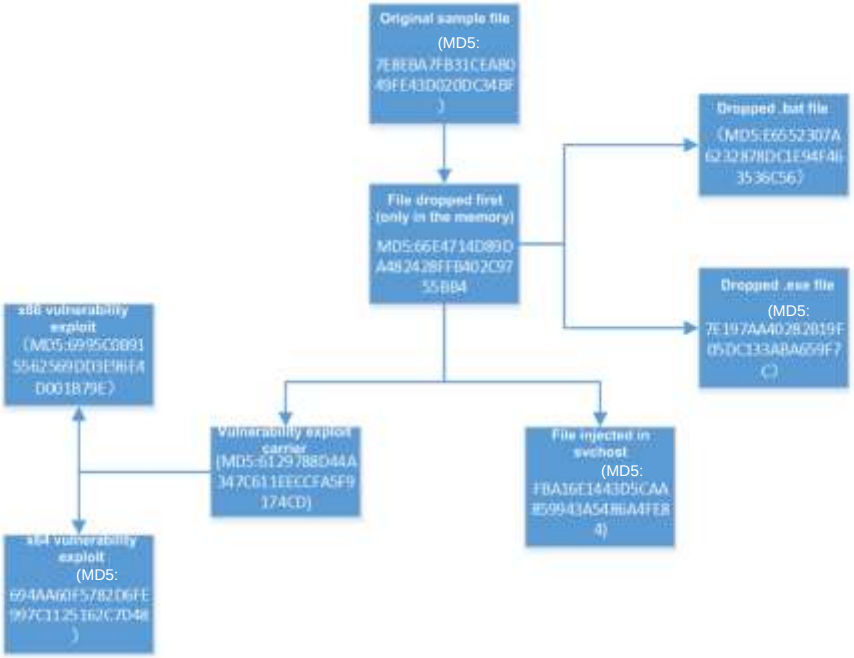
(2) Detection Results of TAC



The following table lists other same-origin samples:

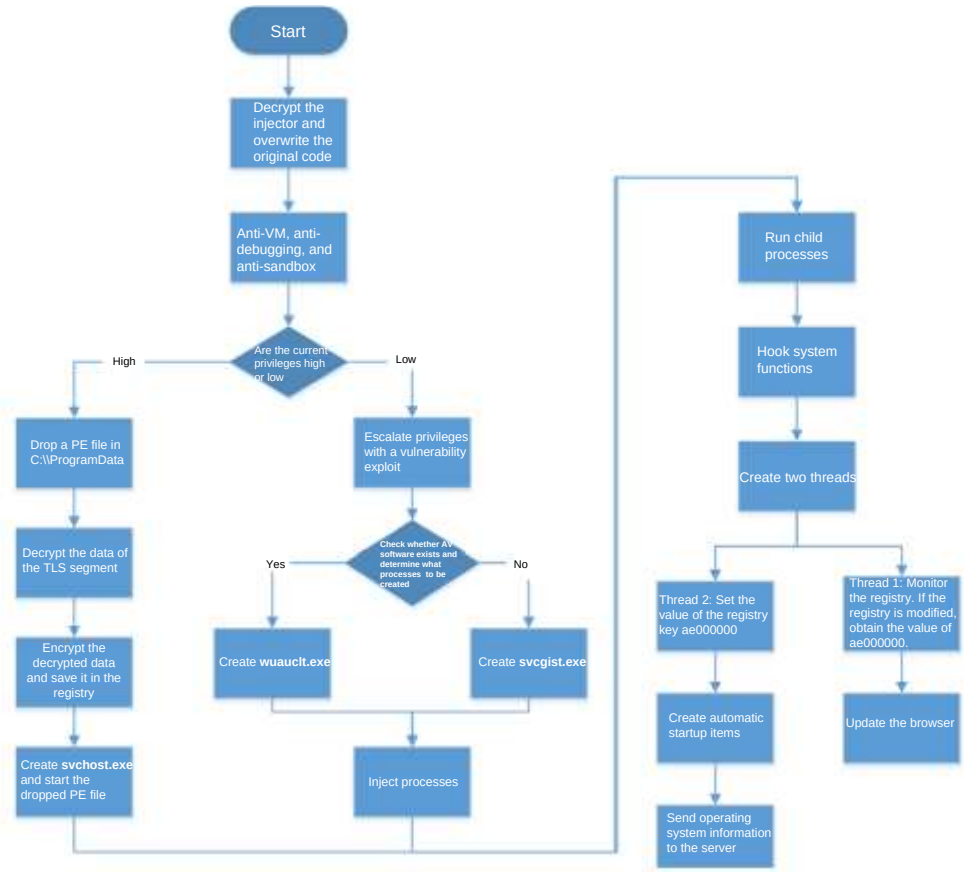
MD5	File Size	Risk Level
f25528baf3d68444fa7d7fda382e9835	338948	High
ebf3e72f8b698bbb0d026416d7a75a6a	338948	Medium
e98459c647a6e328c8b65945884ef29a	338948	Medium

(3) File Structure



(4) Main Functions

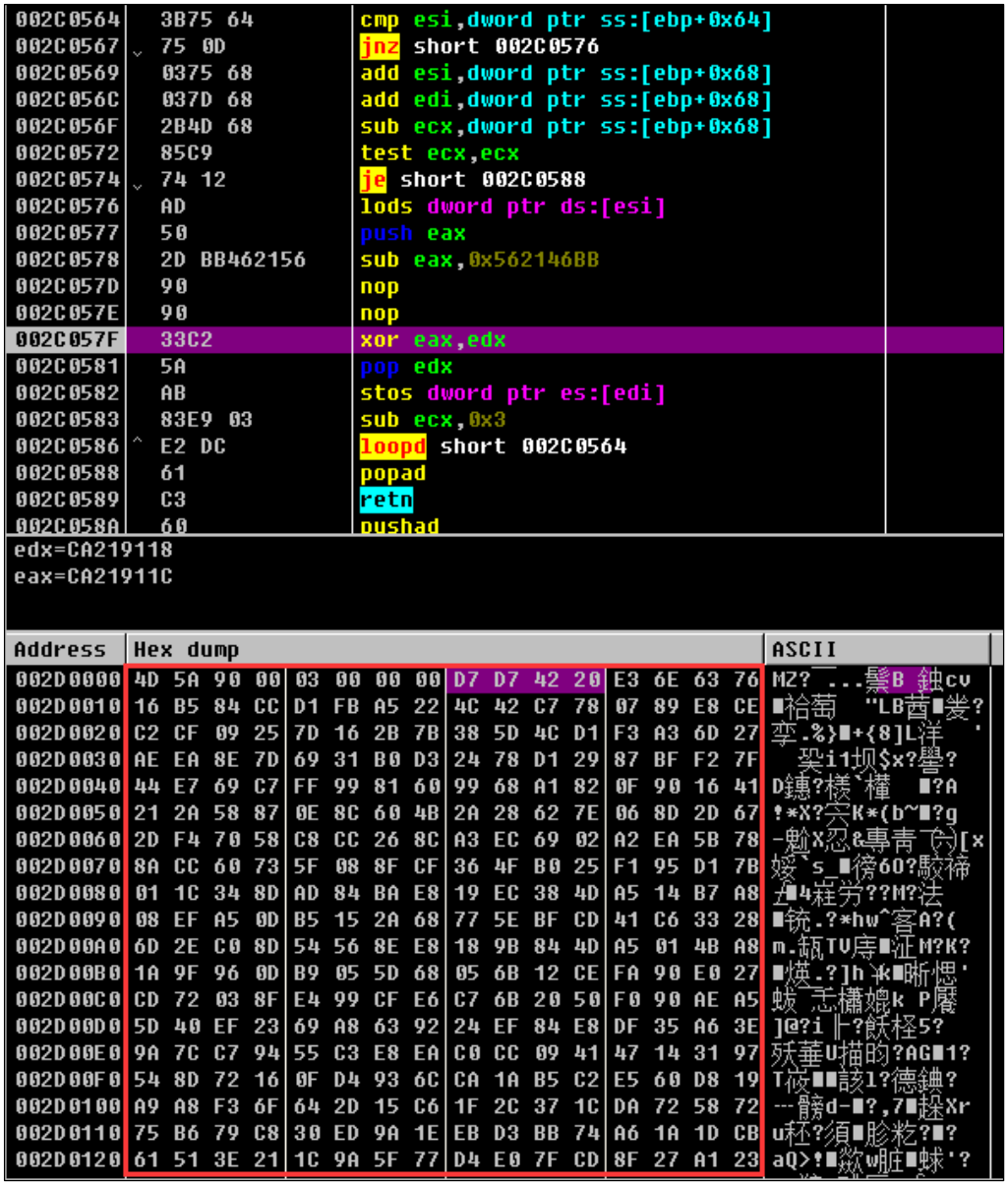
[1] Covert attack: Attacks are completed through multiple encryptions and process injections.
[2] Network behavior: The sample collects information (including but not limited to the local time zone, current time, operating system version, antivirus software version, and host name) about local hosts, uploads it to the remote server, and keeps communicating with the remote server to control the user and steal its information.
[3] Sandbox detection: The sample provides anti-debugging and anti-virtual machine (VM) functions. That is, the sample checks whether it is likely to be in the sandbox by comparing file names, process names, user names, and system signatures.
[4] Confrontation with antivirus tools: The sample can detect multiple analysis tools, antivirus software, and sandboxes. When antivirus software is found, this sample enters a sleep infinite loop, exhibiting no malicious behaviors. When a sandbox is detected, this sample terminates the script interpreter, traffic capture tool, binary analysis tool, and other processes, cutting off the interaction between the sandbox and the outside or preventing the sandbox's automated analysis of this sample.
[5] Persistent attack: This sample, via concealing and self-starting, implements persistent attacks against target hosts, by taking the following actions: injecting **svchost.exe** for concealing processes and creating JavaScript scripts in the Startup folder on the Start menu for completing self-starting.
The following figure shows the sample execution process.



Functions of this sample are as follows:

- Decrypting the injector to overwrite the original code.

Code for decrypting the PE file is as follows:

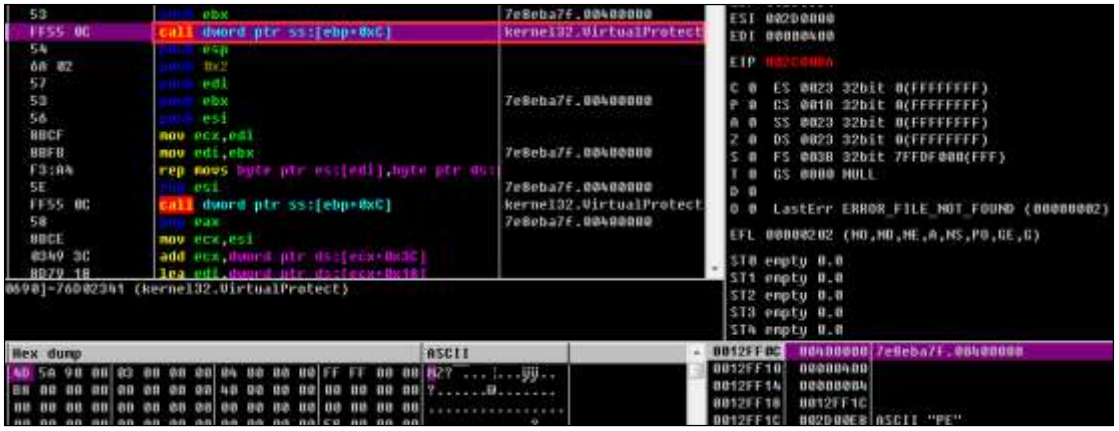


Code for the decryption algorithm is as follows:

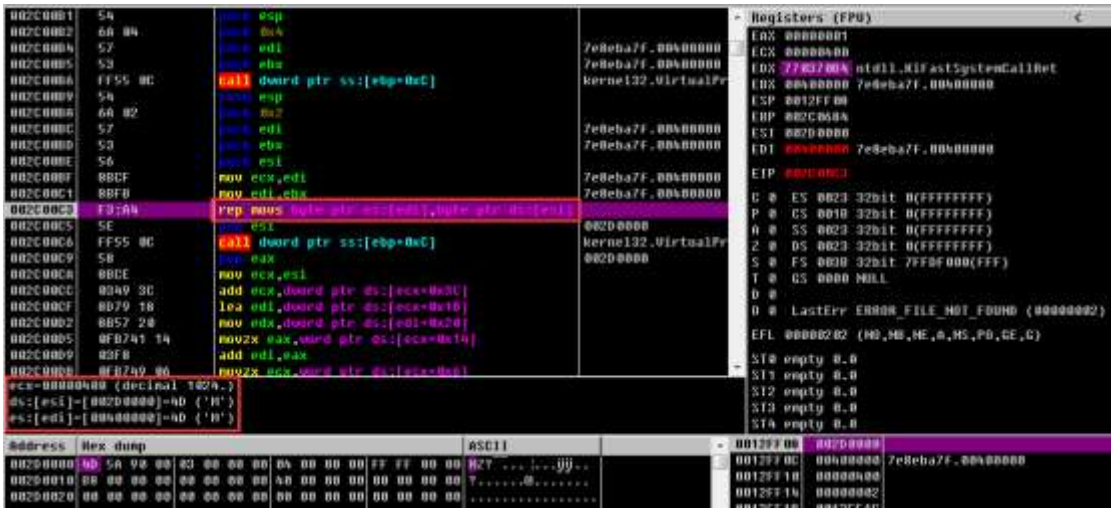
002C0564	3B75 64	cmp esi,dword ptr ss:[ebp+0x64]
002C0567	75 0D	jnz short 002C0576
002C0569	0375 68	add esi,dword ptr ss:[ebp+0x68]
002C056C	037D 68	add edi,dword ptr ss:[ebp+0x68]
002C056F	2B4D 68	sub ecx,dword ptr ss:[ebp+0x68]
002C0572	85C9	test ecx,ecx
002C0574	74 12	je short 002C0588
002C0576	AD	lods dword ptr ds:[esi]
002C0577	50	push eax
002C0578	2D BB462156	sub eax,0x562146BB
002C057D	90	nop
002C057E	90	nop
002C057F	33C2	xor eax,edx

002C0581	5A	pop edx
002C0582	AB	stos dword ptr es:[edi]
002C0583	83E9 03	sub ecx,0x3
002C0586	^ E2 DC	loopd short 002C0564

The memory attribute at 0x400000 is changed to writable.



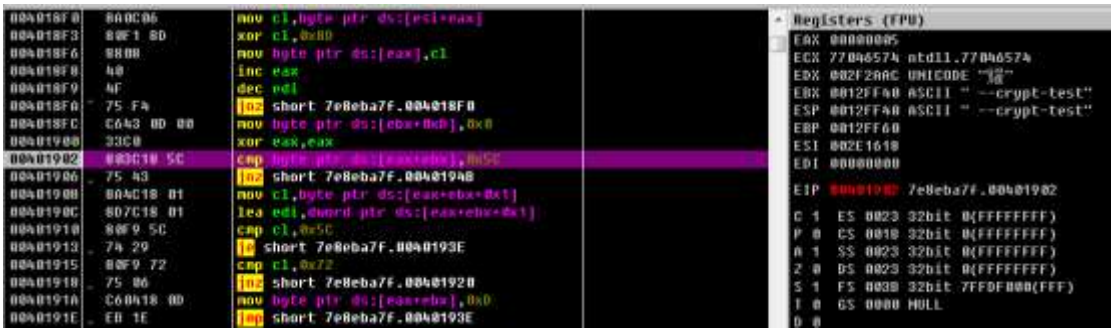
The content at 0x400000 is replaced (the address space of the original sample is overwritten and essentially replaced by the injection module).



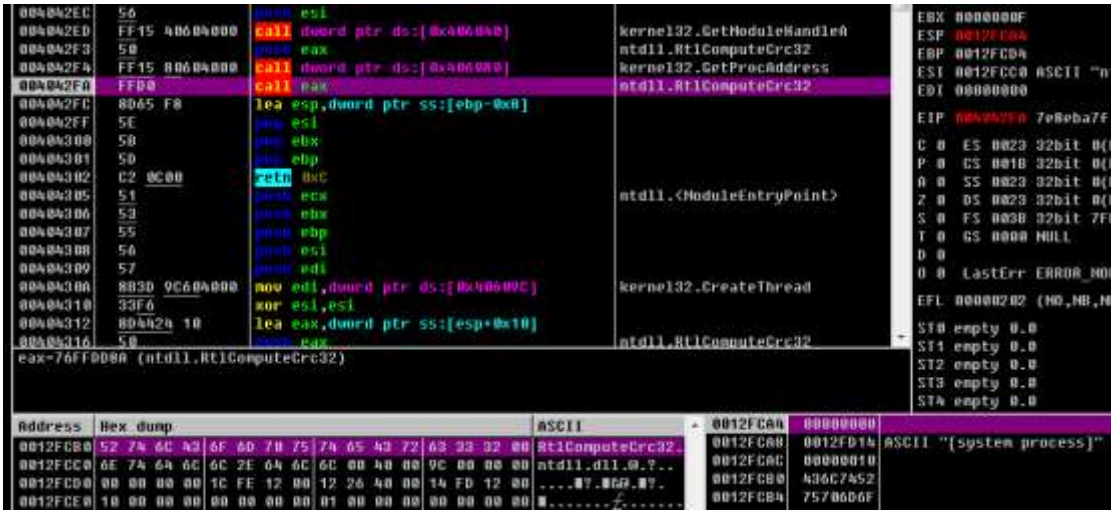
The function address is changed in IAT:

- Anti-VM, anti-debugging, and sandbox detection
  - Anti-VM

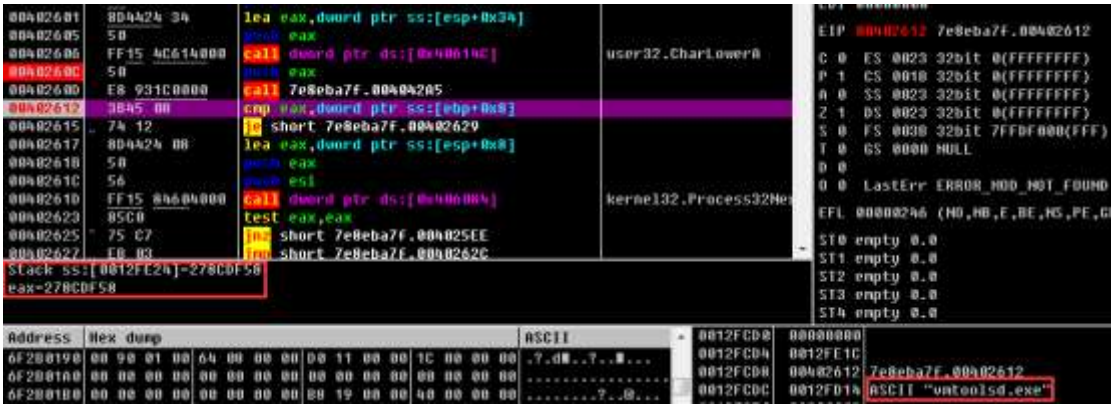
    The sample identifies parameters contained in the command line.



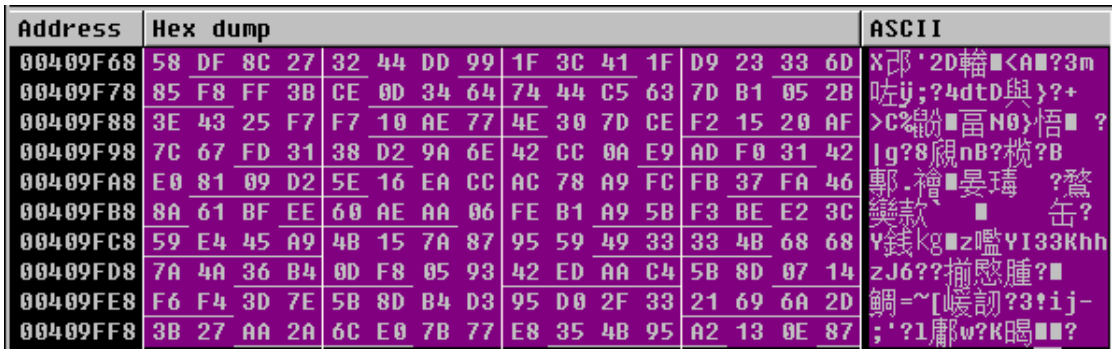
Then this sample searches the process list for target processes, encrypts the names of obtained processes with CRC32, and invokes the RtlComputeCrc32 function.



The sample checks whether such process names indicate specific processes such as **vmtoolsd.exe**, and then compares these names with hard code that is obtained after specific process names are encrypted with CRC32.



The sample searches for the following checksums in the system:



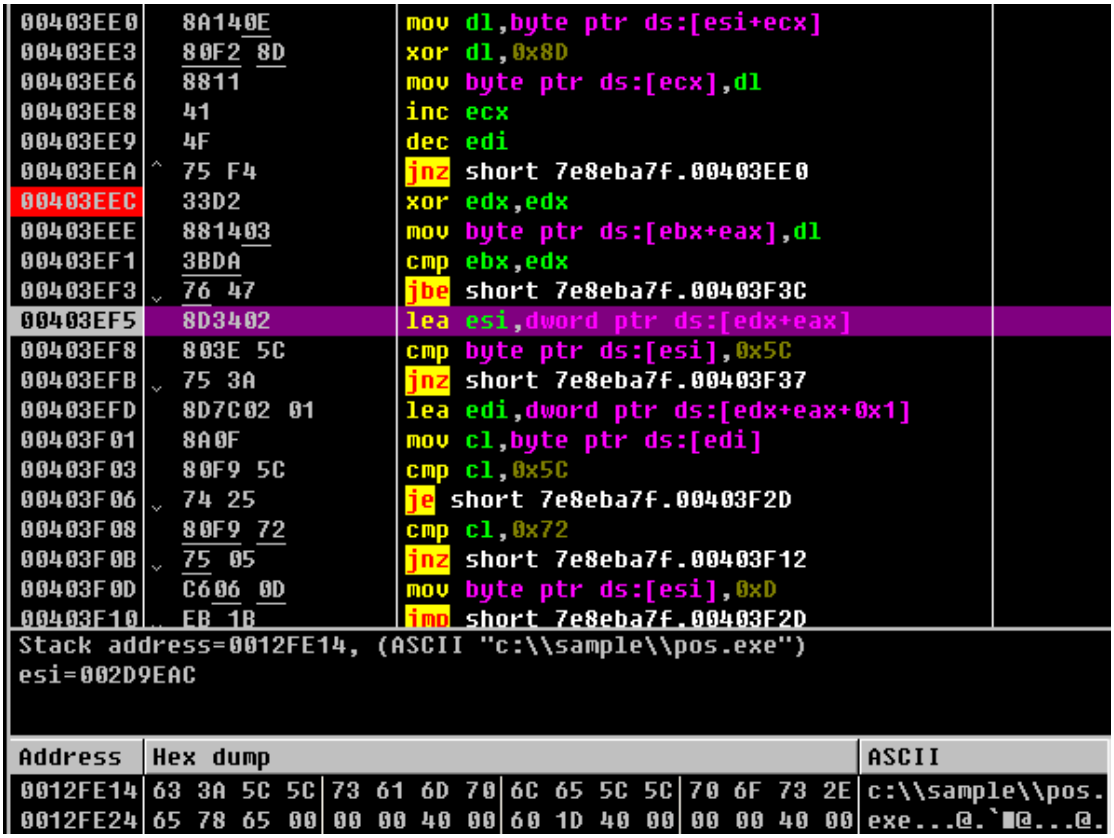
Address	Hex dump																ASCII
00409F68	58	DF	8C	27	32	44	DD	99	1F	3C	41	1F	D9	23	33	6D	X邵'2D輜■<A■?3m
00409F78	85	F8	FF	3B	CE	0D	34	64	74	44	C5	63	7D	B1	05	2B	咗ÿ;?4dtD與}?+
00409F88	3E	43	25	F7	F7	10	AE	77	4E	30	7D	CE	F2	15	20	AF	>C%鞠■冨N0}悟■ ?
00409F98	7C	67	FD	31	38	D2	9A	6E	42	CC	0A	E9	AD	F0	31	42	|g?8覛nB?榄?B
00409FA8	E0	81	09	D2	5E	16	EA	CC	AC	78	A9	FC	FB	37	FA	46	鄴.禬■曼琦 ?鶖
00409FB8	8A	61	BF	EE	60	AE	AA	06	FE	B1	A9	5B	F3	BE	E2	3C	孌款 ■ 缶?
00409FC8	59	E4	45	A9	4B	15	7A	87	95	59	49	33	33	4B	68	68	Y銭kg■z嚗YI33Khh
00409FD8	7A	4A	36	B4	0D	F8	05	93	42	ED	AA	C4	5B	8D	07	14	zJ6??揃愍膧?■
00409FE8	F6	F4	3D	7E	5B	8D	B4	D3	95	D0	2F	33	21	69	6A	2D	鯛=~[嵼訒?3!ij-
00409FF8	3B	27	AA	2A	6C	E0	7B	77	E8	35	4B	95	A2	13	0E	87	;'?1鄘w?K暍■■?

The process name code are listed as follows:

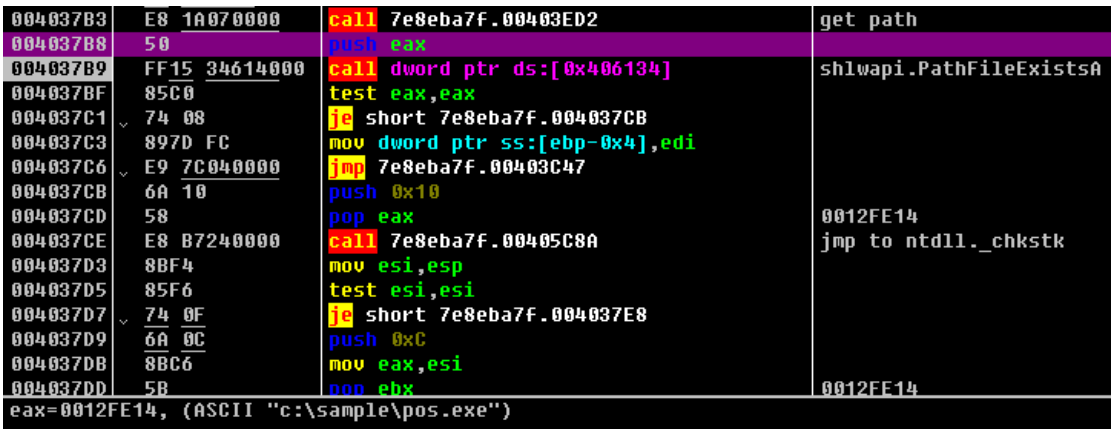
0x278CDF58 – vmtoolsd.exe
0x99DD4432 – ?
0x1F413C1F – vmwaretray.exe
0x6D3323D9 – vmusrvc.exe
0x3BFFF885 – vmsrvc.exe
0x64340DCE – ?
0x63C54474 – vboxtray.exe
0x2B05B17D – ?
0xF725433E – ?
0x77AE10F7 – ?
0xCE7D304E – dumpcap.exe
0xAF2015F2 – ollydbg.exe
0x31FD677C – importrec.exe
0x6E9AD238 – petools.exe
0xE90ACC42 – idag.exe
0x4231F0AD – sysanalyzer.exe
0xD20981E0 – sniff_hit.exe
0xCCEA165E – scktool.exe
0xFCA978AC – proc_analyzer.exe
0x46FA37FB – hookexplorer.exe
0xEEBF618A – multi_pot.exe
0x06AAAE60 – idaq.exe
0x5BA9B1FE – procmon.exe
0x3CE2BEF3 – regmon.exe
0xA945E459 – procexp.exe
0x877A154B – peid.exe
0x33495995 – autoruns.exe
0x68684B33 – autorunsc.exe
0xB4364A7A – ?
0x9305F80D – imul.exe
0xC4AAED42 – emul.exe
0x14078D5B – apispy.exe
0x7E3DF4F6 – ?

0xD3B48D5B – hookanaapp.exe
0x332FD095 – fortitracer.exe
0x2D6A6921 – ?
0x2AAA273B – joeboxserver.exe
0x777BE06C – joeboxcontrol.exe
0x954B35E8 – ?
0x870E13A2 – ?

The c:\sample\pos.exe path is obtained.



PathFileExistsA is invoked to check the existence of the file:



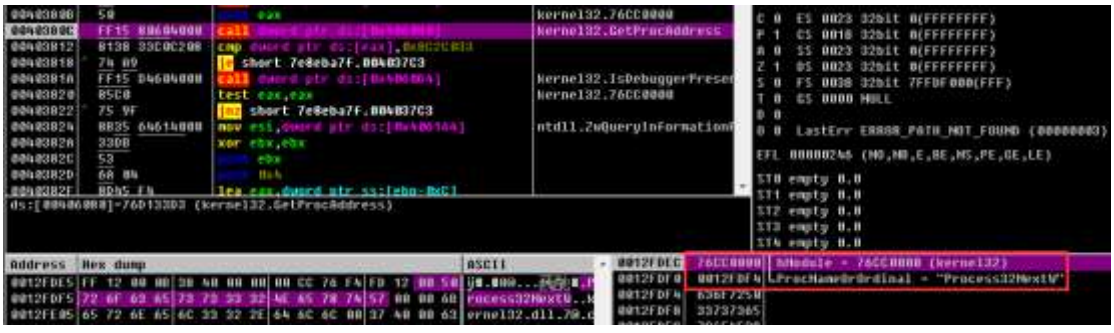
If this file does not exist, this sample obtains kernel32.dll.

```
00403EE0    8A140E          mov dl,byte ptr ds:[esi+ecx]
00403EE3    80F2 8D         xor dl,0x8D
00403EE6    8811            mov byte ptr ds:[ecx],dl
00403EE8    41              inc ecx
00403EE9    4F              dec edi
00403EEA  ^ 75 F4           jnz short 7e8eba7f.00403EE0
00403EEC    33D2            xor edx,edx                          7e8eba7f
00403EEE    881403          mov byte ptr ds:[ebx+eax],dl
00403EF1    3BDA            cmp ebx,edx                          7e8eba7f
00403EF3  v 76 47           jbe short 7e8eba7f.00403F3C
00403EF5    8D3402          lea esi,dword ptr ds:[edx+eax]
00403EF8    803E 5C         cmp byte ptr ds:[esi],0x5C
00403EFB  v 75 3A           jnz short 7e8eba7f.00403F37
00403EFD    8D7C02 01       lea edi,dword ptr ds:[edx+eax+0x1]
00403F01    8A0F            mov cl,byte ptr ds:[edi]
edx=0040976C (7e8eba7f.0040976C)

Address   Hex dump                                          ASCII
0012FE04  6B 65 72 6E 65 6C 33 32 2E 64 6C 6C D3 37 40 00  kernel32.dll?@.
0012FE14  63 3A 5C 73 61 6D 70 6C 65 5C 70 6F 73 2E 65 78  c:\sample\pos.ex
```

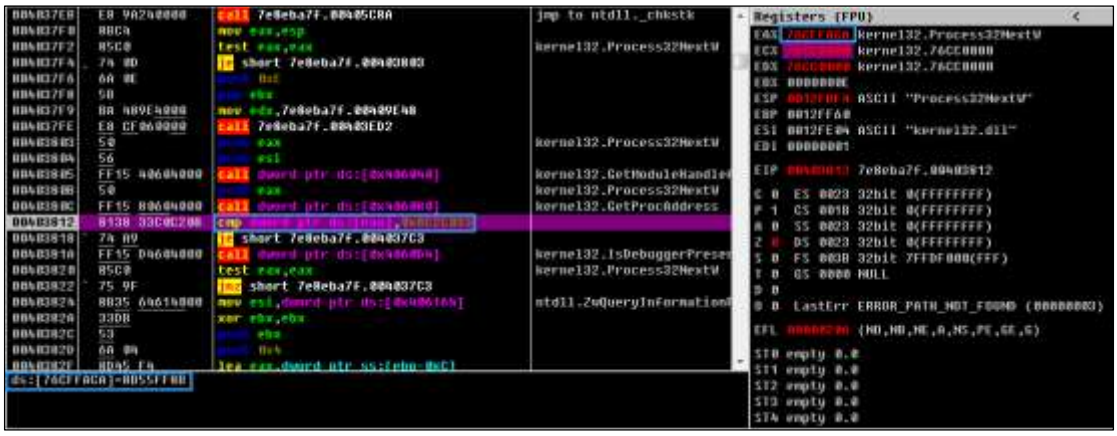
The function name Process32NextW is obtained:

```
00403EEA  ^ 75 F4           jnz short 7e8eba7f.00403EE0
00403EEC    33D2            xor edx,edx                          7e8eba7f
00403EEE    881403          mov byte ptr ds:[ebx+eax],dl
00403EF1    3BDA            cmp ebx,edx                          7e8eba7f
00403EF3  v 76 47           jbe short 7e8eba7f.00403F3C
00403EF5    8D3402          lea esi,dword ptr ds:[edx+eax]
00403EF8    803E 5C         cmp byte ptr ds:[esi],0x5C
00403EFB  v 75 3A           jnz short 7e8eba7f.00403F37
00403EFD    8D7C02 01       lea edi,dword ptr ds:[edx+eax+0x1]
00403F01    8A0F            mov cl,byte ptr ds:[edi]
edx=00409E57 (7e8eba7f.00409E57)

Address   Hex dump                                          ASCII
0012FDE5  00 00 00 01 00 00 00 04 FE 12 00 03 38 40 00 50  ...£...¦?.¯8@.P
0012FDF5  72 6F 63 65 73 73 33 32 4E 65 78 74 57 40 00 6B  rocess32NextW@.k
```

The function address is obtained:



The sample checks whether the first four bytes of the function address is 0x8C2C033:



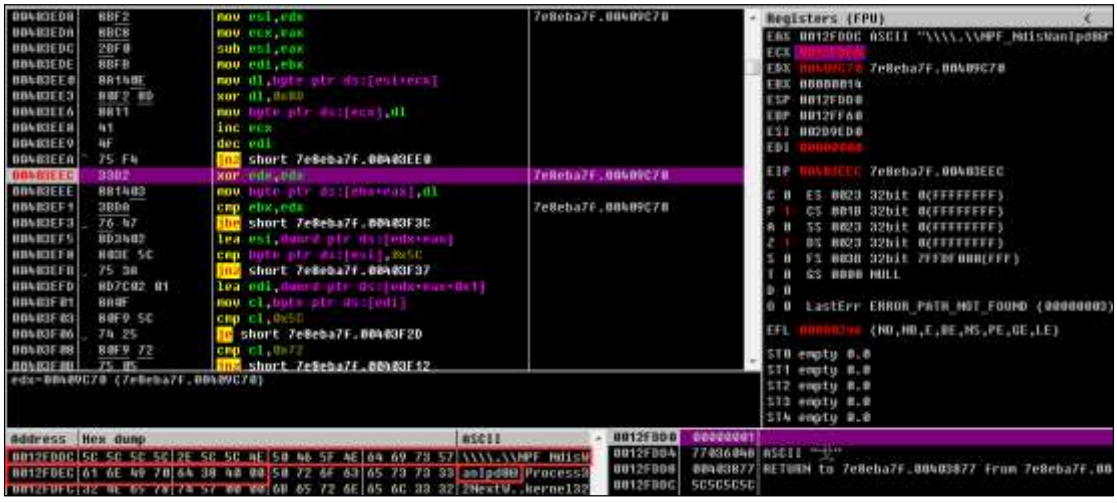
- Anti-debugging

kernel32.IsDebuggerPresent is invoked:

The sample checks whether DebugPort and ExceptionPort are occupied to determine whether the sample is in the remote debugging state.

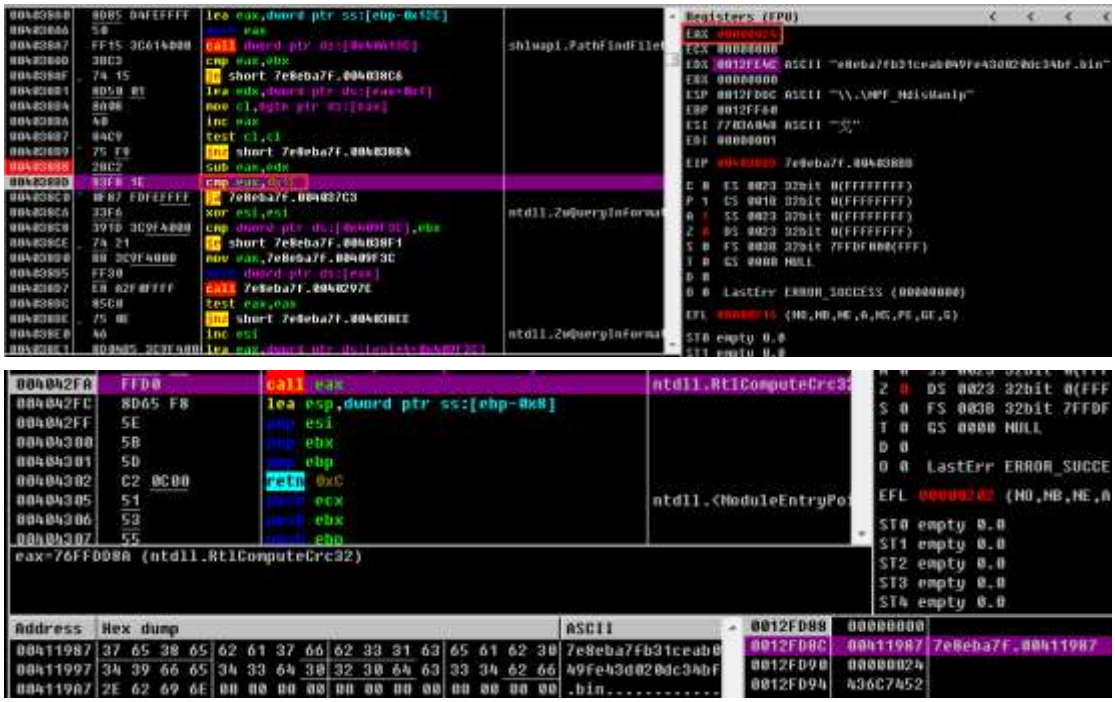
```
94  if ( *(_DWORD *)GetProcAddress(v8, v48) == 0x8C2C033// Process32NextW
95    || IsDebuggerPresent()
96    || ZwQueryInformationProcess((HANDLE)0xFFFFFFFF, ProcessDebugPort, &ProcessInformation, 4u, 0) >= 0
97    && ProcessInformation )
98  {
99    goto LABEL_96;
100 }
101 if ( ZwQueryInformationProcess(
102         (HANDLE)0xFFFFFFFF,
103         ProcessPriorityBoost|ProcessExceptionPort,
104         &ProcessInformation,
105         4u,
106         0) >= 0
107   && ProcessInformation )
108 {
109   goto LABEL_96;                              // error
110 }
```

Then the sample obtains \\.\NPF_NdisWanIp to determine the existence of Wireshark.

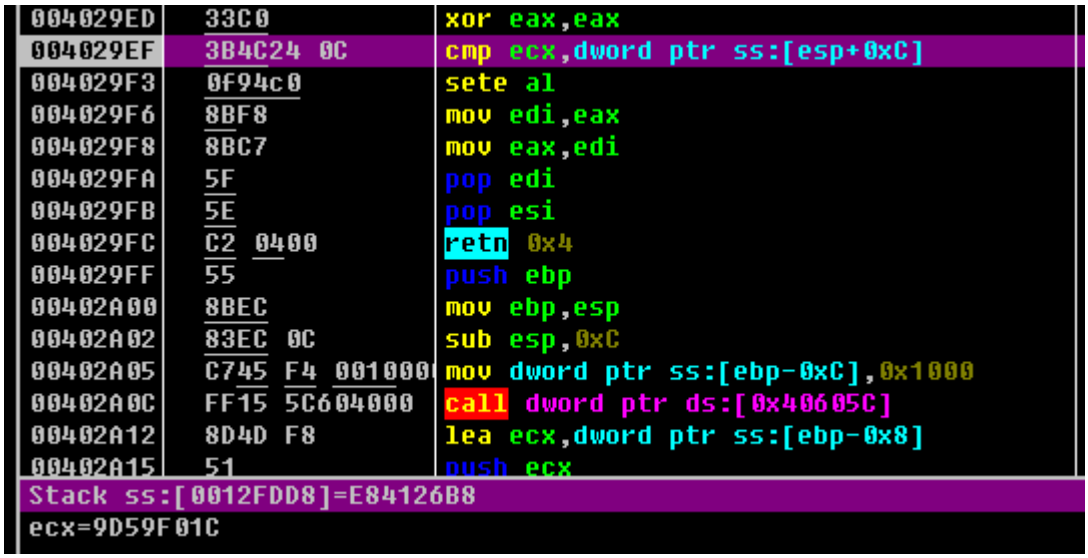


- Sandbox detection

◆ Check on the length of the file name (if the file name contains over 30 characters, it indicates that the sandbox runs.)



◆ Comparison with specific values:

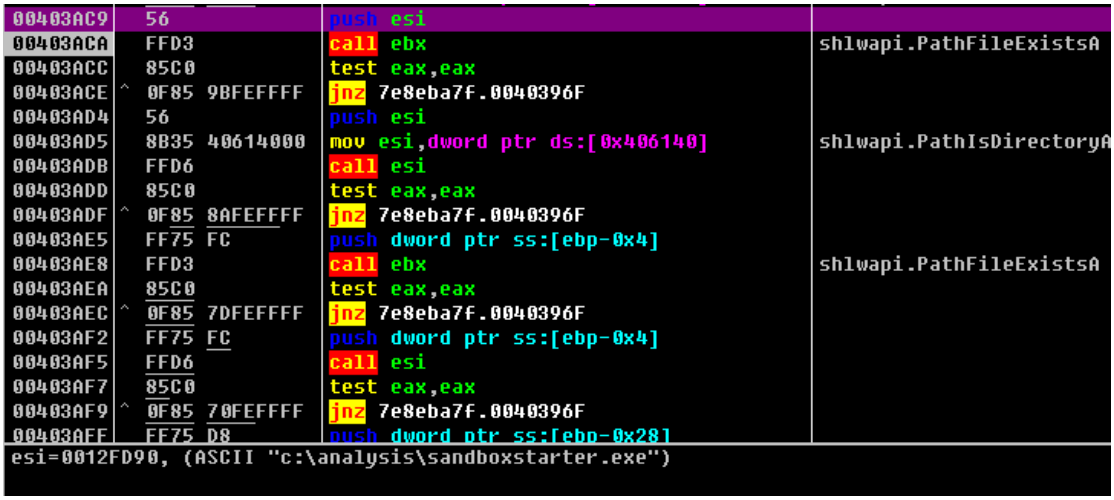


A comparison is made between the CRC32 checksum of the file name and the following values in the sample (that is, new names assigned by common sandboxes to the sample):

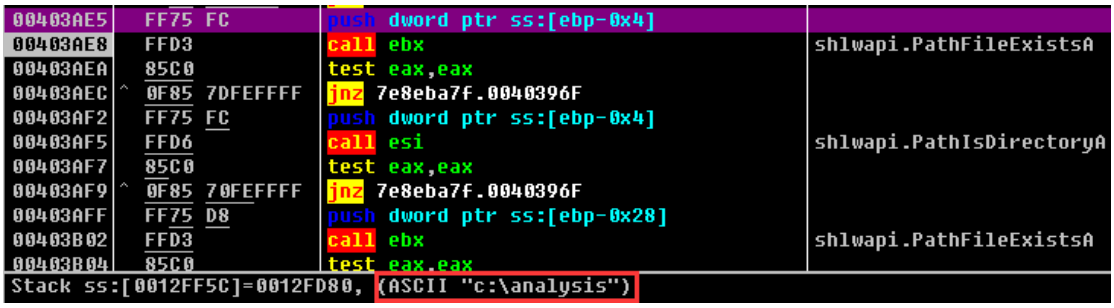
0xE84126B8 – sample.exe
0x0A84E285 – ?
0x3C164BED – ?
0xC19DADCE – ?

0xA07ACEDD – ?
0xD254F323 – ?
0xF3C4E556 – ?
0xF8782263 – ?
0xCA96016D – ?

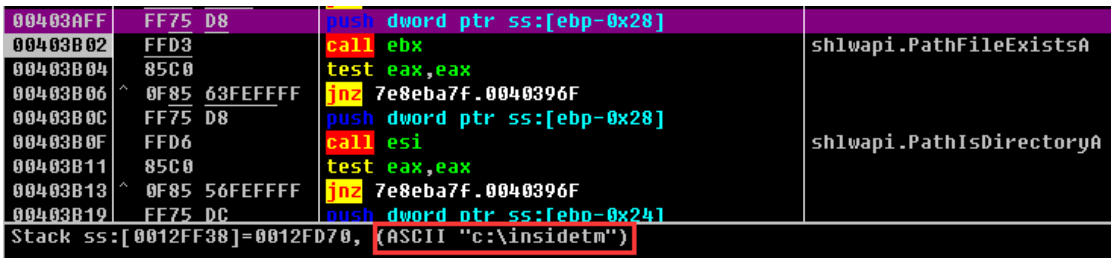
Then the sample searches for specific files and directories such as c:\analysis\sandboxstarter.exe as shown in the following figure:



The c:\analysis directory is retrieved:



The c:\insidetm directory is retrieved:



The c:\windows\system32\drivers\vmmouse.sys directory is retrieved:

```
00403B19   FF75 DC        push dword ptr ss:[ebp-0x24]
00403B1C   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B1E   85C0           test eax,eax
00403B20 ^ 0F85 49FEFFFF  jnz 7e8eba7f.0040396F
00403B26   FF75 DC        push dword ptr ss:[ebp-0x24]
00403B29   FFD6           call esi                      shlwapi.PathIsDirectoryA
00403B2B   85C0           test eax,eax
00403B2D ^ 0F85 3CFEFFFF  jnz 7e8eba7f.0040396F
00403B33   FF75 E8        push dword ptr ss:[ebp-0x18]
Stack ss:[0012FF3C]=0012FD44, (ASCII "c:\windows\system32\drivers\vmmouse.sys")
```

The c:\windows\system32\drivers\vmhgfs.sys directory is retrieved:

```
00403B33   FF75 E8        push dword ptr ss:[ebp-0x18]
00403B36   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B38   85C0           test eax,eax
00403B3A ^ 0F85 2FFEFFFF  jnz 7e8eba7f.0040396F
00403B40   FF75 E8        push dword ptr ss:[ebp-0x18]
00403B43   FFD6           call esi                      shlwapi.PathIsDirectoryA
00403B45   85C0           test eax,eax
00403B47 ^ 0F85 22FEFFFF  jnz 7e8eba7f.0040396F
00403B4D   FF75 F0        push dword ptr ss:[ebp-0x10]
Stack ss:[0012FF48]=0012FD18, (ASCII "c:\windows\system32\drivers\vmhgfs.sys")
```

The c:\windows\system32\drivers\vboxmouse.sys directory is retrieved:

```
00403B4D   FF75 F0        push dword ptr ss:[ebp-0x10]
00403B50   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B52   85C0           test eax,eax
00403B54 ^ 0F85 15FEFFFF  jnz 7e8eba7f.0040396F
00403B5A   FF75 F0        push dword ptr ss:[ebp-0x10]
00403B5D   FFD6           call esi                      shlwapi.PathIsDirectoryA
00403B5F   85C0           test eax,eax
00403B61 ^ 0F85 08FEFFFF  jnz 7e8eba7f.0040396F
00403B67   FF75 E0        push dword ptr ss:[ebp-0x20]
00403B6A   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B6C   85C0           test eax,eax
Stack ss:[0012FF50]=0012FCE8, (ASCII "c:\windows\system32\drivers\vboxmouse.sys")
```

The c:\iDEFENSE directory is retrieved:

```
00403B67   FF75 E0        push dword ptr ss:[ebp-0x20]
00403B6A   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B6C   85C0           test eax,eax
00403B6E ^ 0F85 FBFDFFFF  jnz 7e8eba7f.0040396F
00403B74   FF75 E0        push dword ptr ss:[ebp-0x20]
00403B77   FFD6           call esi                      shlwapi.PathIsDirectoryA
00403B79   85C0           test eax,eax
00403B7B ^ 0F85 EEFDFFFF  jnz 7e8eba7f.0040396F
00403B81   FF75 EC        push dword ptr ss:[ebp-0x14]
00403B84   FFD3           call ebx                      shlwapi.PathFileExistsA
00403B86   85C0           test eax,eax
Stack ss:[0012FF40]=0012FCD8, (ASCII "c:\iDEFENSE")
```

The c:\popupkiller.exe directory is retrieved:

```
00403B81   FF75 EC          push dword ptr ss:[ebp-0x14]
00403B84   FFD3             call ebx                      shlwapi.PathFileExistsA
00403B86   85C0             test eax,eax
00403B88 ^ 0F85 E1FDFFFF    jnz 7e8eba7f.0040396F
00403B8E   FF75 EC          push dword ptr ss:[ebp-0x14]
00403B91   FFD6             call esi                      shlwapi.PathIsDirectoryA
00403B93   85C0             test eax,eax
00403B95 ^ 0F85 D4FDFFFF    jnz 7e8eba7f.0040396F
00403B9B   FF75 E4          push dword ptr ss:[ebp-0x1C]
00403B9E   FFD3             call ebx                      shlwapi.PathFileExistsA
00403BA0   85C0             test eax,eax
Stack ss:[0012FF4C]=0012FCC4, (ASCII "c:\popupkiller.exe")
```

The c:\tools\execute.exe directory is retrieved:

```
00403B9B   FF75 E4          push dword ptr ss:[ebp-0x1C]
00403B9E   FFD3             call ebx                      shlwapi.PathFileExistsA
00403BA0   85C0             test eax,eax
00403BA2 ^ 0F85 C7FDFFFF    jnz 7e8eba7f.0040396F
00403BA8   FF75 E4          push dword ptr ss:[ebp-0x1C]
00403BAB   FFD6             call esi                      shlwapi.PathIsDirectoryA
00403BAD   85C0             test eax,eax
Stack ss:[0012FF44]=0012FCAC, (ASCII "c:\tools\execute.exe")
```

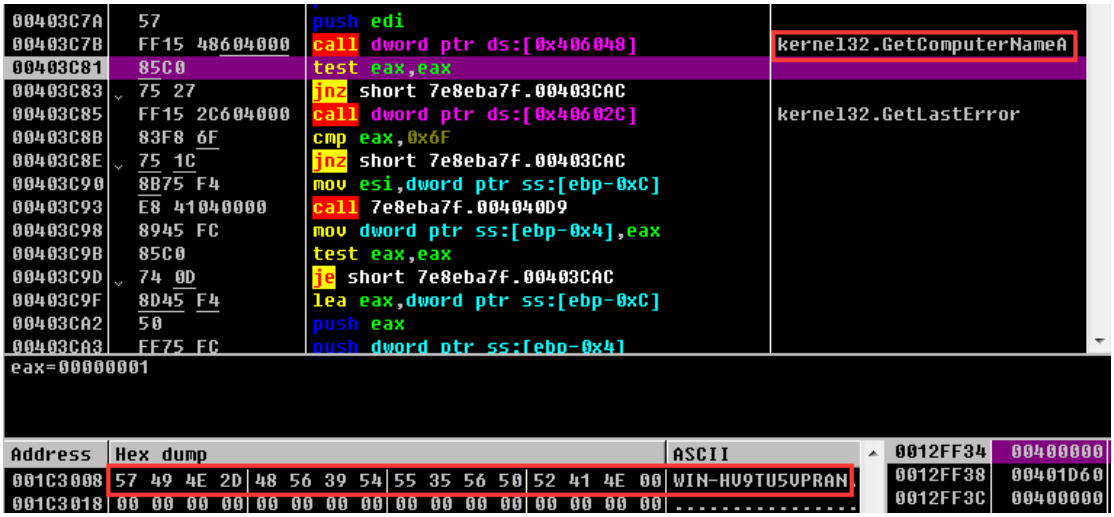
The c:\Perl directory is retrieved:

```
00403BB5   FF75 F8          push dword ptr ss:[ebp-0x8]
00403BB8   FFD3             call ebx                      shlwapi.PathFileExistsA
00403BBA   85C0             test eax,eax
00403BBC ^ 0F85 ADFDFFFF    jnz 7e8eba7f.0040396F
00403BC2   FF75 F8          push dword ptr ss:[ebp-0x8]
Stack ss:[0012FF58]=0012FCA0, (ASCII "c:\Perl")
```

The c:\Python27 directory is retrieved:

```
00403BCF   57               push edi
00403BD0   FFD3             call ebx                      shlwapi.PathFileExistsA
00403BD2   85C0             test eax,eax
00403BD4 ^ 0F85 95FDFFFF    jnz 7e8eba7f.0040396F
00403BDA   57               push edi
00403BDB   FFD6             call esi                      shlwapi.PathIsDirectoryA
00403BDD   85C0             test eax,eax
00403BDF ^ 0F85 8AFDFFFF    jnz 7e8eba7f.0040396F
00403BE5   2145 FC          and dword ptr ss:[ebp-0x4],eax
00403BE8   E8 D1EEFFFF      call 7e8eba7f.00402ABE
edi=0012FC90, (ASCII "c:\Python27")
```

- Comparison with computer names and user names. The computer name is obtained:

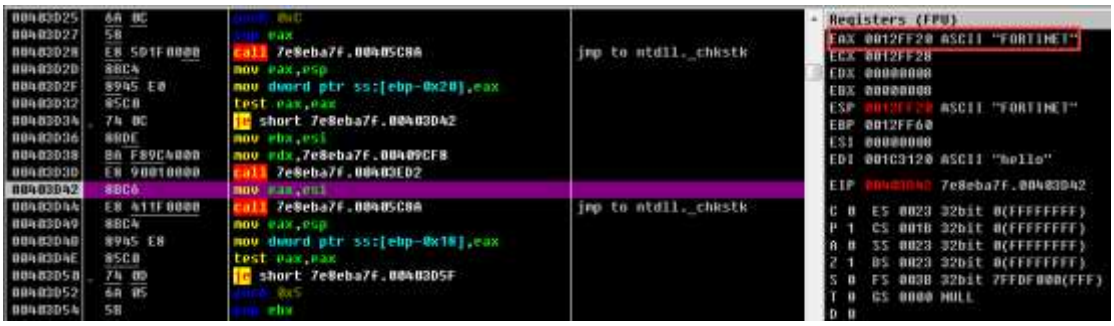


The user name is obtained:

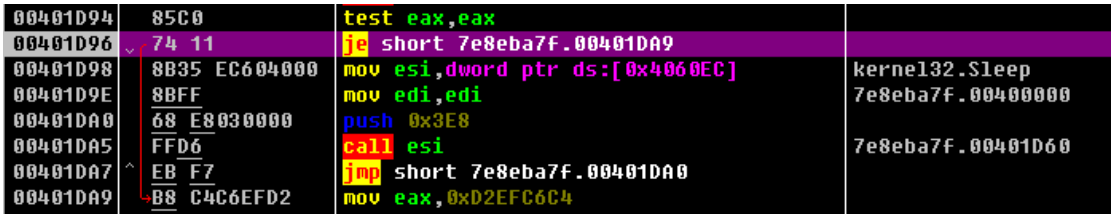
```
  if ( !GetComputerNameA(v0, &nSize) && GetLastError() == 111 )
  {
    lpBuffer = (LPSTR)sub_4040D9();
    if ( lpBuffer )
      GetComputerNameA(lpBuffer, &nSize);
  }
}
v20 = 260;
v1 = (const CHAR *)sub_40408A();
lpFirst = v1;
if ( v1 )
{
  if ( !GetUserNameA((LPSTR)v1, &v20) && GetLastError() == 122 )
  {
    v2 = (const CHAR *)sub_4040D9();
    lpFirst = v2;
    if ( v2 )
      GetUserNameA((LPSTR)v2, &v20);
  }
}
```

The obtained user name is compared with the following character strings (common sandbox user names):

SANDBOX
FORTINET
VIRUS
MALWARE
MALNETVM



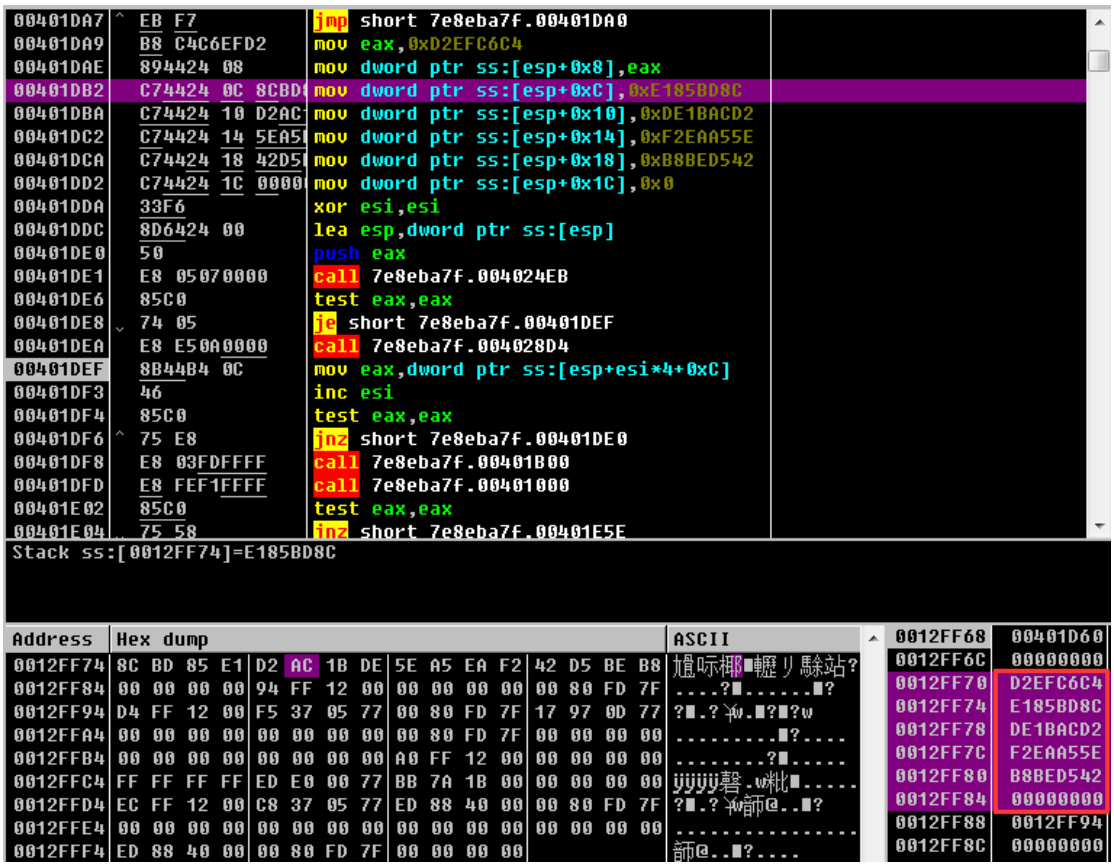
If a matching character string is found, the sample enters the sleep infinite loop:



The sample searches for the following processes:

0xD2EFC6C4 – python.exe
0xE185BD8C – pythonw.exe
0xDE1BACD2 – perl.exe
0xF2EAA55E – autoit3.exe
0xB8BED542 – ?

Process names are encrypted with CRC32 as follows:



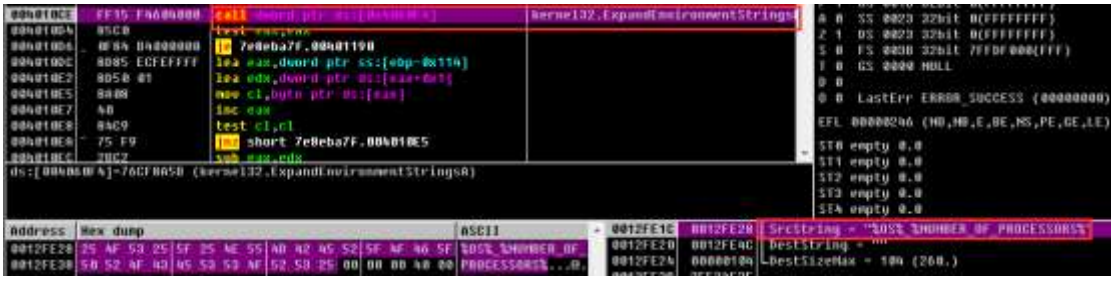
If the preceding processes are running, terminate them (ending the internal control of the sandbox). If the sample runs in the Windows XP (32-bit) environment, it also enters the sleep loop.

- Check of current privileges

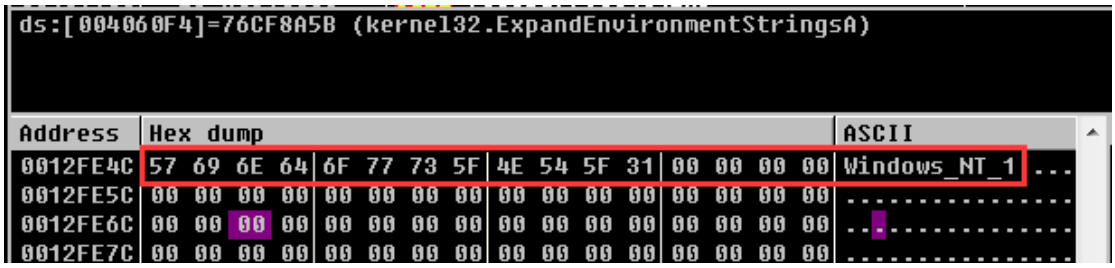
If the sample runs with insufficient process privileges or in Window 7 or Windows Server 2008, vulnerability exploit code will be executed; otherwise, the original handling process will be followed.

- Actions with high privileges

The environment variable is obtained and expanded:



Here, Windows_NT_1 is obtained in the Windows 7 (x86) environment:



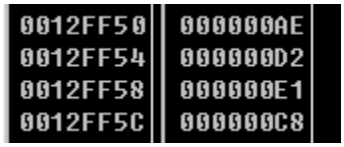
Then the CRC32 value of 0xFFFFFFFF, 0xEEEEEEEE, 0xAAAAAAAA, and 0x77777777 is calculated:

```
v12 = sub_4042A5((int)&Dst, v5, 0xFFFFFFFF);// calculate 0xFFFFFFFF CRC32 value
BYTE3(v13) = sub_4042A5((int)&Dst, v5, 0xEEEEEEEE);// calculate 0xEEEEEEEE CRC32 value
BYTE3(v14) = sub_4042A5((int)&Dst, v5, 0xAAAAAAAA);// same
BYTE3(v15) = sub_4042A5((int)&Dst, v5, 0x77777777);// same
```

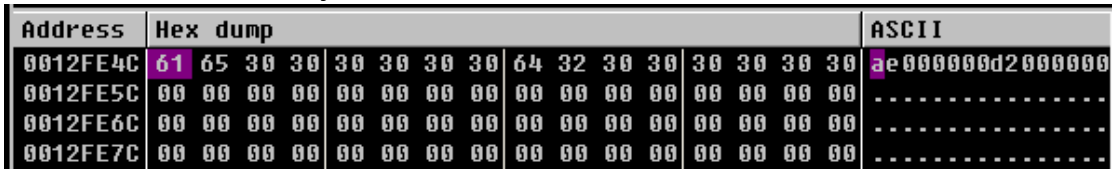
The calculated CRC32 values are as follows:

0x395693AE
0xB24495D2
0xF39F86E1
0xBAE0B5C8

The last two bits of each of the preceding hexadecimal values are stored in the stack, as shown in the following figure:

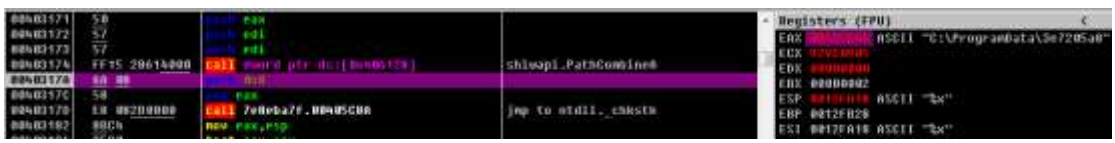


0xAE000000 followed by 0xD2000000 will be used as the Atom name:



GlobalAtom named ae000000d2000000 is added:

A folder is created at the current time:



The folder is opened:

```
0012FB10  0012FC3C  FileName = "C:\Users\hello\Desktop\7e8eba7fb31ceab049fe43d020dc34bf.bin"
0012FB14  80000000  Access = GENERIC_READ
0012FB18  00000003  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012FB1C  00000000  pSecurity = NULL
0012FB20  00000003  Mode = OPEN_EXISTING
0012FB24  00000080  Attributes = NORMAL
0012FB28  00000000  hTemplateFile = NULL
0012FB2C  00400000  7e8eba7f.00400000
0012FB30  00400000  7e8eba7f.00400000
```

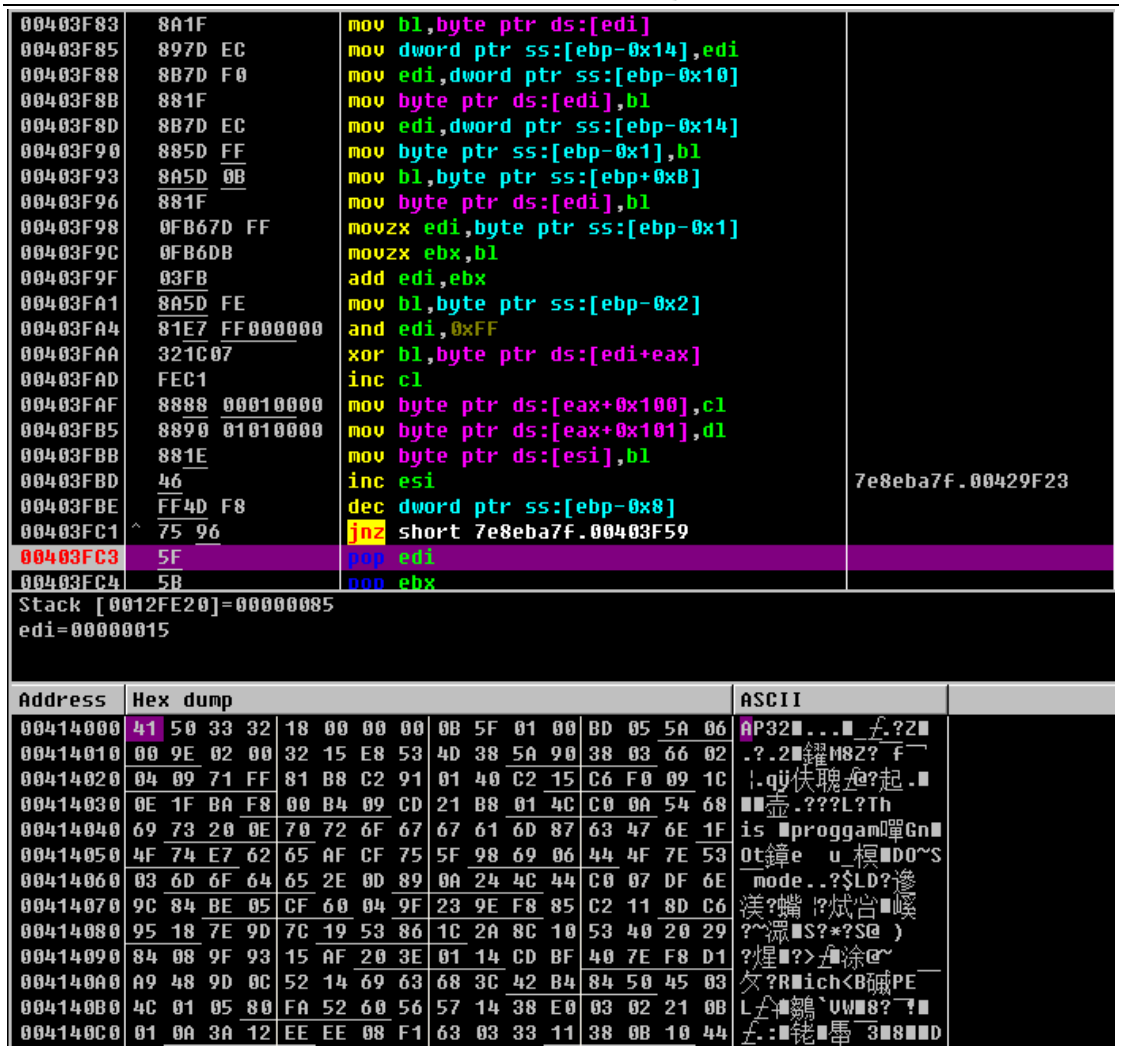
A file named **3e7205a8.exe** is created under C:\ProgramData.

```
0012FB10  0012FE44  FileName = "C:\ProgramData\3e7205a8.exe"
0012FB14  C0000000  Access = GENERIC_READ|GENERIC_WRITE
0012FB18  00000003  ShareMode = FILE_SHARE_READ|FILE_SHARE_WRITE
0012FB1C  00000000  pSecurity = NULL
0012FB20  00000002  Mode = CREATE_ALWAYS
0012FB24  00000080  Attributes = NORMAL
0012FB28  00000000  hTemplateFile = NULL
0012FB2C  00400000  7e8eba7f.00400000
0012FB30  00400000  7e8eba7f.00400000
```

Then the sample reads the content in its own .rsrc and .tls segments and initializes a 256-byte array as follows: v17[]={0, 1, 2, 3, …, 255}. After that, this sample reads the .rsrc segment from the beginning, performs the XOR operation with each bit in the array, and stores the result in the array.

```
v19 = sub_402C51();
v0 = (_BYTE *)sub_402BA3(".tls");              // read segment .tls bytes
v1 = (const char *)sub_402BA3(".rsrc");       // read segment .rsrc bytes
v2 = strlen(v1);
v3 = 0;
v18 = 1;
v4 = 0;
do
{
  v17[v4] = v4;
  ++v4;
}
while ( v4 < 256 );                           // set v17[256] as [0,1,2,...,255]
v5 = 0;
do
{
  v17[v5] ^= v1[v3++];                        // xor 256 bytes in .rsrc use v17[]
  if ( v3 >= v2 )                             // and put result in v17[]
    v3 = 0;
  ++v5;
}
while ( v5 < 256 );
```

The content in the .tls segment is decrypted:



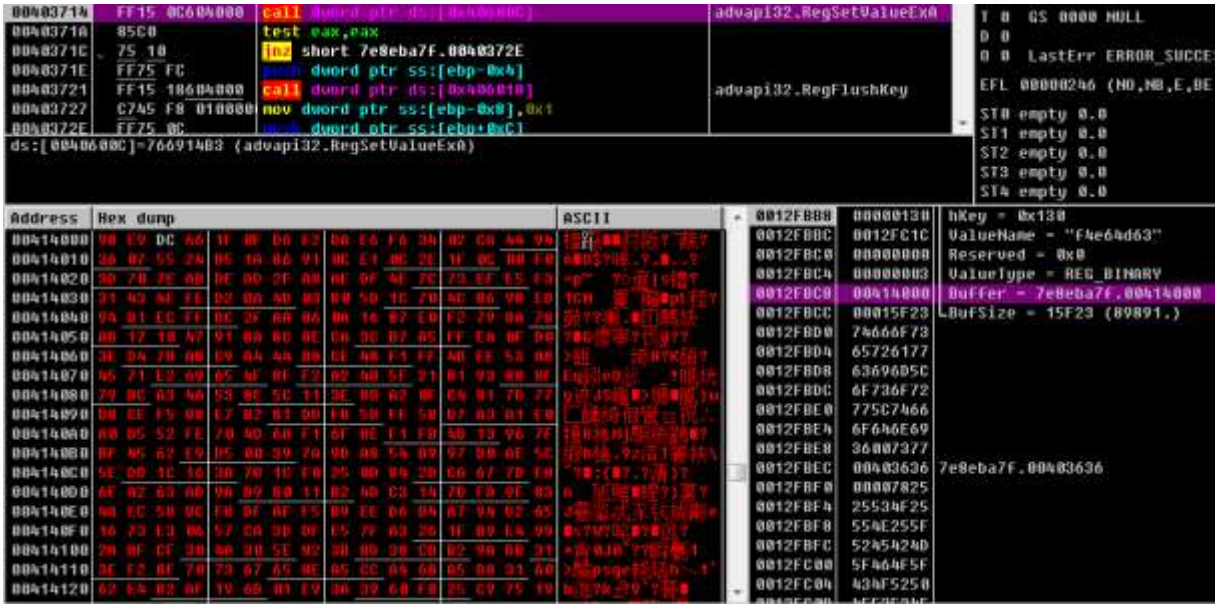
Roughly speaking, the decryption result is a PE file compressed by aPLib compression library.

The content in this file will be encrypted:

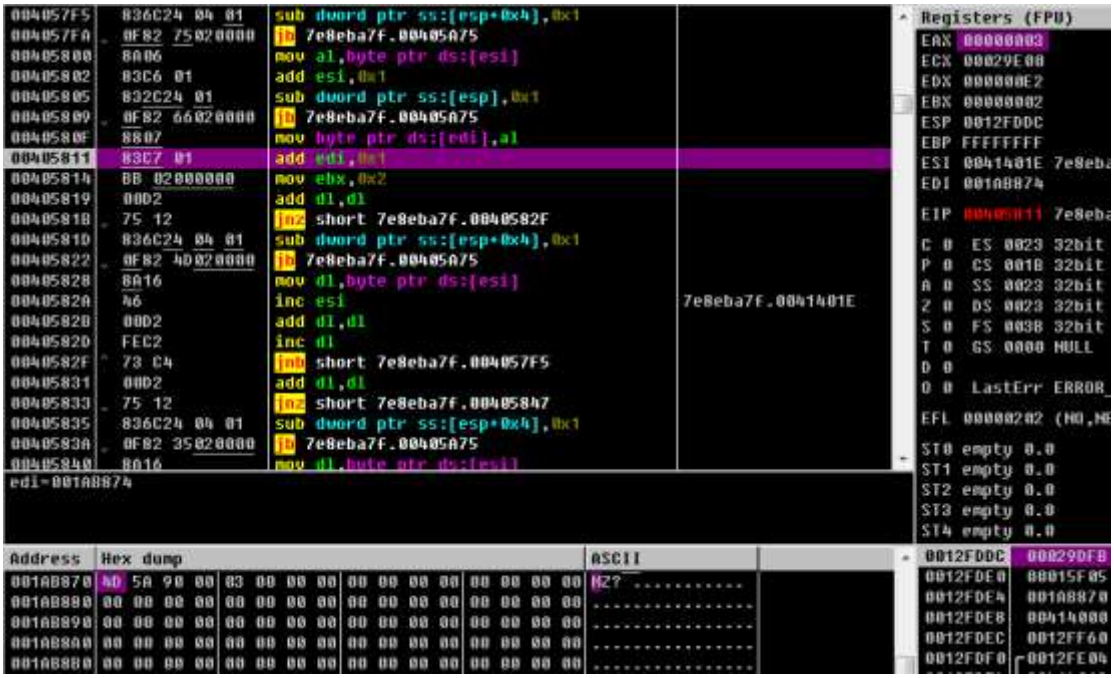
```
do
{
  v8[v5] = v5;
  ++v5;
}
while ( v5 < 256 );                          // set v8[] as [0,1,2,....,255]
v6 = 0;
do
{
  v8[v6] ^= a1[v4++];                        // encrypt PE file
  if ( v4 >= v3 )
    v4 = 0;
  ++v6;
}
while ( v6 < 256 );
```

The encrypted content will be written into the registry:

The content is written as follows:



The compressed file is decompressed as follows:



The flag bit in the file header is changed:

```
00401704   66:8911          mov word ptr ds:[ecx],dx
00401707   C70408 393F000   mov dword ptr ds:[eax+ecx],0x3F39
0040170E   5F               pop edi                                  7e8eba7
0040170F   5E               pop esi                                  7e8eba7
00401710   B8 01000000      mov eax,0x1
00401715   5B               pop ebx                                  7e8eba7
00401716   8BE5             mov esp,ebp
00401718   5D               pop ebp                                  7e8eba7
dx=2024
ds:[001AB870]=5A4D

Address   Hex dump                                          ASCII
001AB870  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ? ... |...ÿÿ..
001AB880  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ?.......@.......
001AB890  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001AB8A0  00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............?..
001AB8B0  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ■■?.???L?Th
001AB8C0  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
```

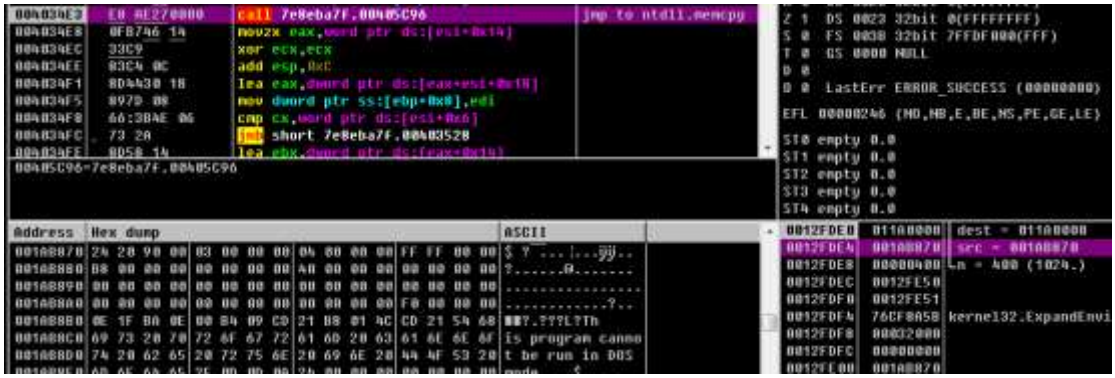
The file is retrieved:

A process is created:

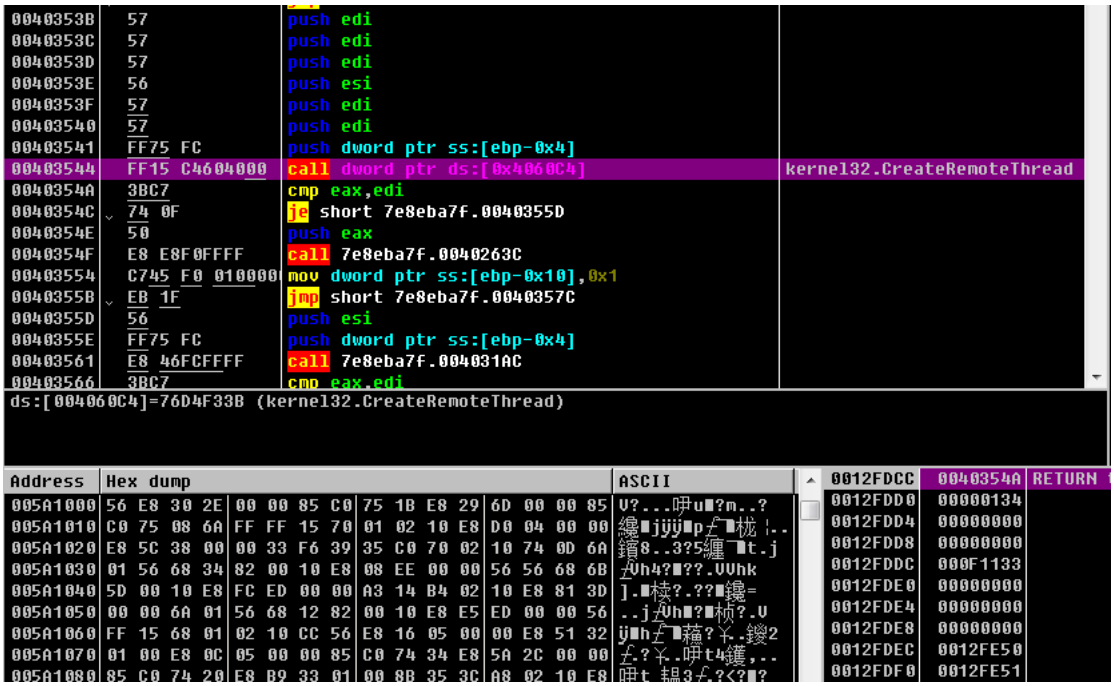
```
0012F9BC   0012FCBC   ModuleFileName = "C:\Windows\system32\svchost.exe"
0012F9C0   004117F0   CommandLine = "C:\ProgramData\3e7205a8.exe"
0012F9C4   00000000   pProcessSecurity = NULL
0012F9C8   00000000   pThreadSecurity = NULL
0012F9CC   00000000   InheritHandles = FALSE
0012F9D0   0000000C   CreationFlags = CREATE_SUSPENDED|DETACHED_PROCESS
0012F9D4   00000000   pEnvironment = NULL
0012F9D8   00000000   CurrentDir = NULL
0012F9DC   0012FDC0   pStartupInfo = 0012FDC0
0012F9E0   0012FE0C   pProcessInfo = 0012FE0C
0012F9E4   0012FE50
```

0xFFFFFFFF is written into the memory:

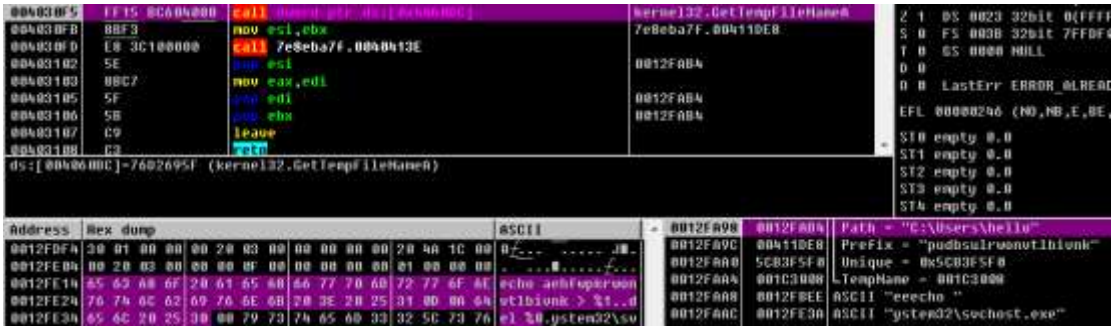
The file is written into the memory where svchost.exe is stored:



A remote thread is created:

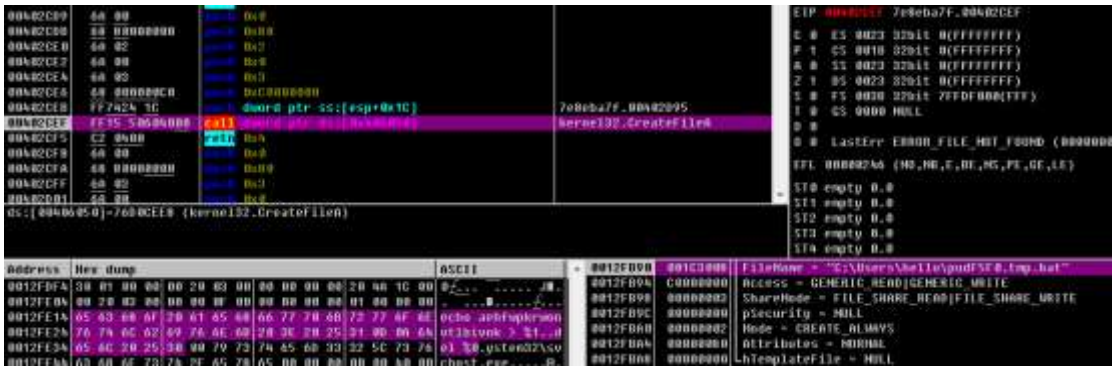


The process shown in the following figure is the one created when the file path is passed. The sample obtains the **tmp** file path to copy itself to the directory indicated in the following figure:

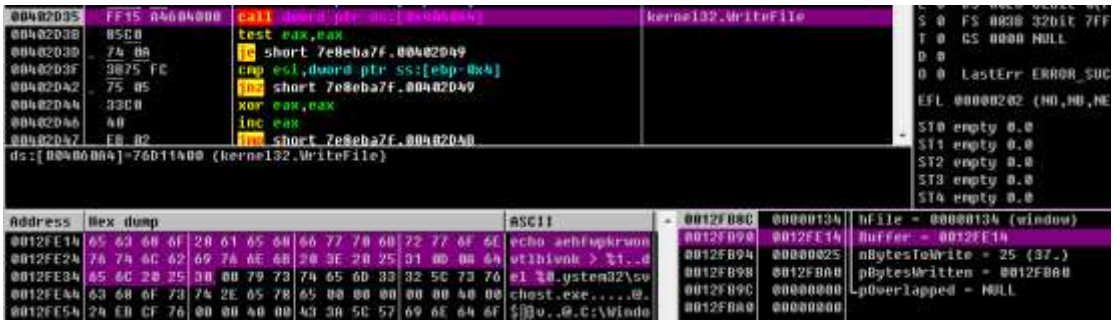


The file path, C:\Users\hello\pudF5F0.tmp, is obtained:

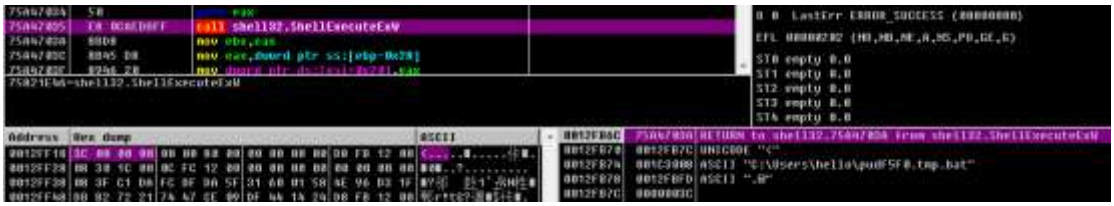
A file is created:



This file is written as follows:



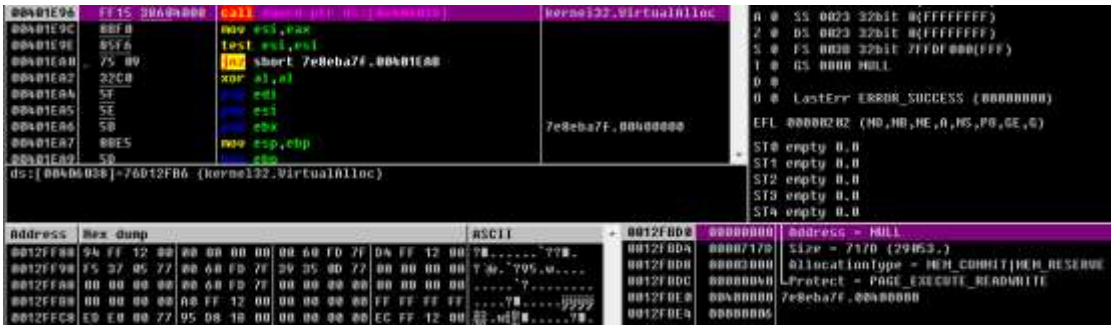
This file is executed：



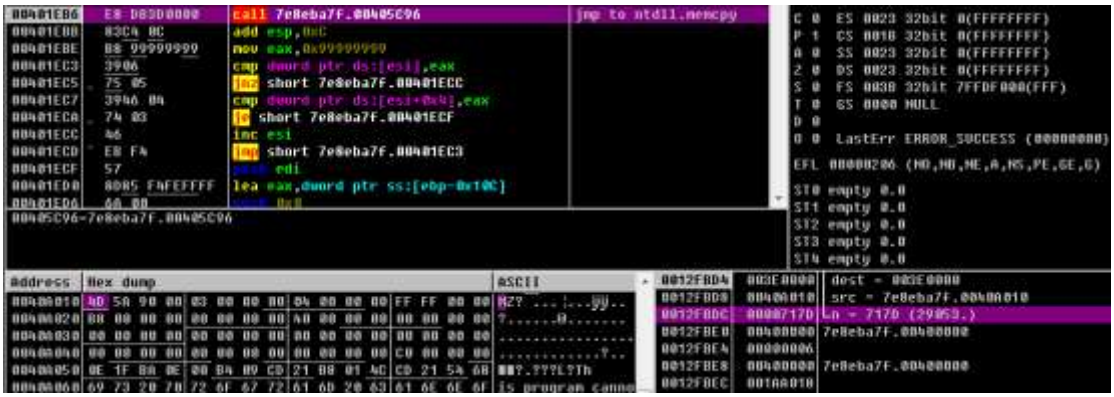
Then the sample exits.

- Actions with low privileges. In the case of insufficient privileges, the sample invokes the exploit function for privilege escalation.

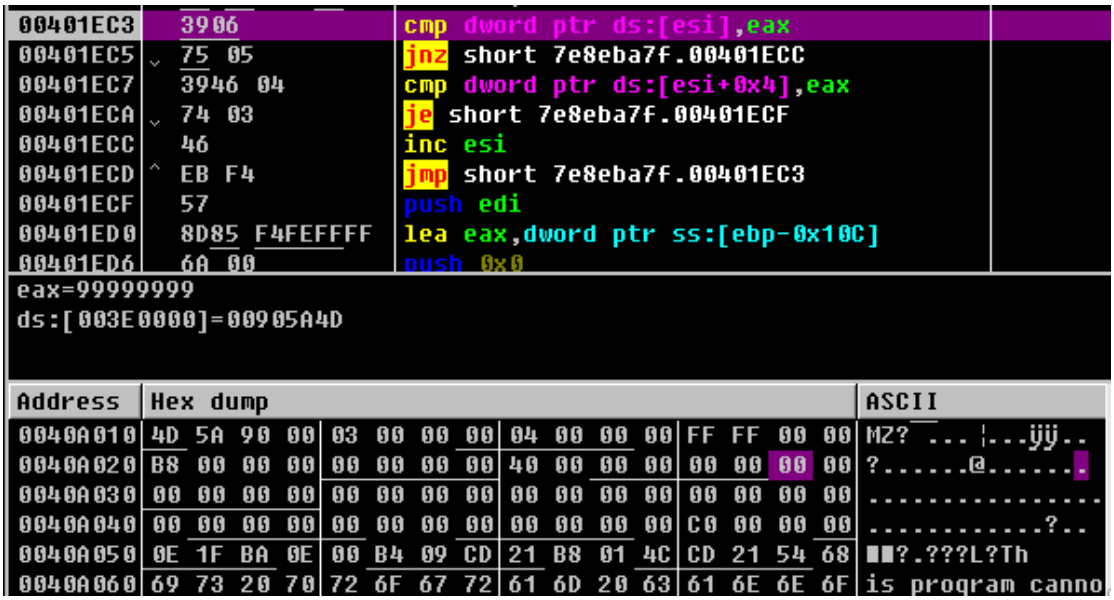
The sample first applies for a segment of memory to store the exploit program temporarily.

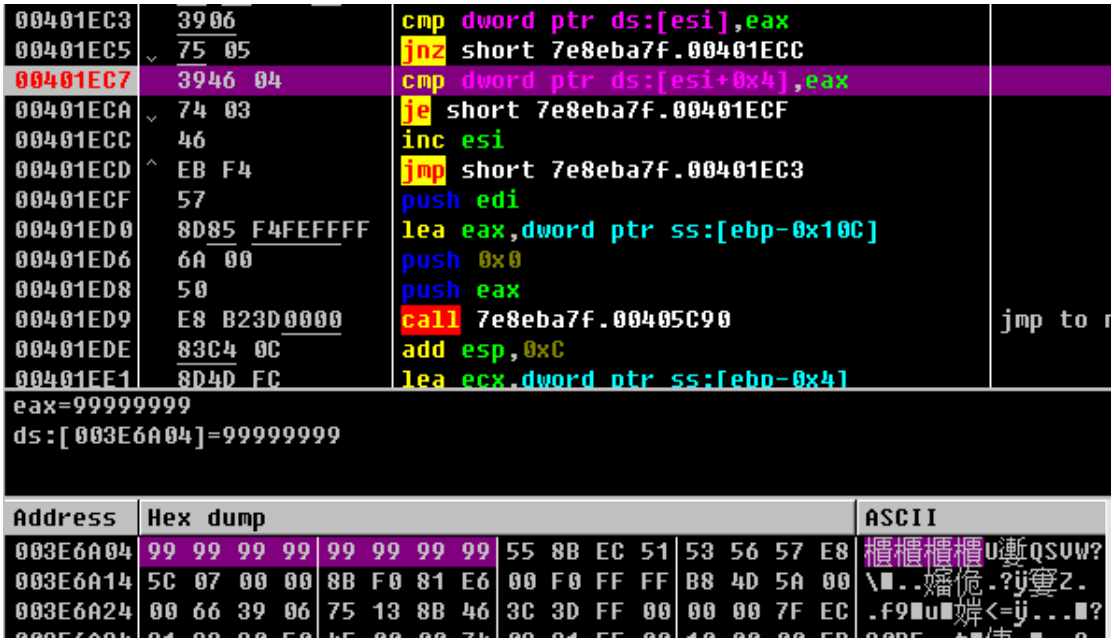


The PE file is copied to the memory:

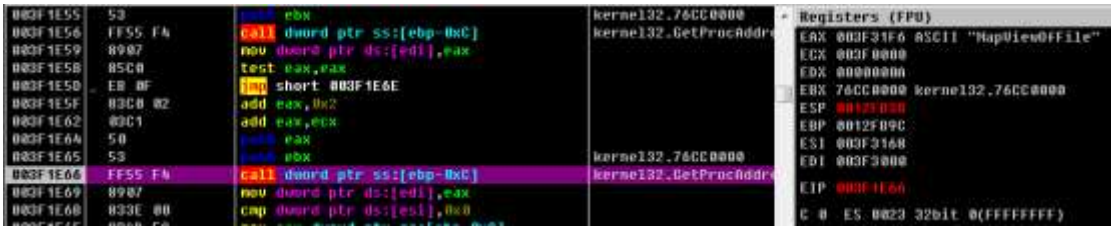


0x99999999 is retrieved in the PE file:

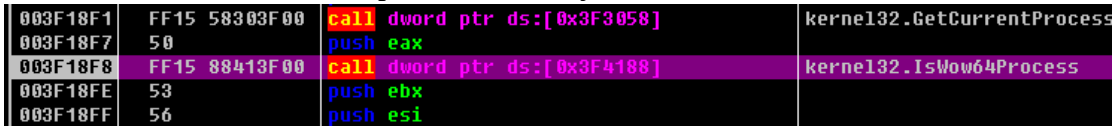




You can see that 0x99999999, followed by the content of the function, identifies the start of the function.
After execution, some functions required for vulnerability exploitation will be loaded at first.



After threat creation, the sample checks the system version:



The vulnerability exploit file is dropped:

```
003F1859   E8 22FFFFFF      call 003F1780
003F185E   B8 4D5A0000      mov eax,0x5A4D
003F1863   66:8906          mov word ptr ds:[esi],ax
003F1866   8B46 3C          mov eax,dword ptr ds:[esi+0x3C]
003F1869   5F               pop edi
003F186A   8B4430 28        mov eax,dword ptr ds:[eax+esi+0x28]
003F186E   03C6             add eax,esi
003F1870   5E               pop esi
003F1871   C3               retn
003F1872   6A 00            push 0x0
003F1874   E8 B7FEFFFF      call 003F1730
eax=015DB200
```

```
Address   Hex dump                                        ASCII
015C0000  00 00 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  ..?  ... ¦...ÿÿ..
015C0010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ?.......@.......
015C0020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
015C0030  00 00 00 00 00 00 00 00 00 00 00 00 D0 00 00 00  ............?..
015C0040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ■■?.???L?Th
015C0050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
015C0060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS
```

The event named WaitEventX is created:

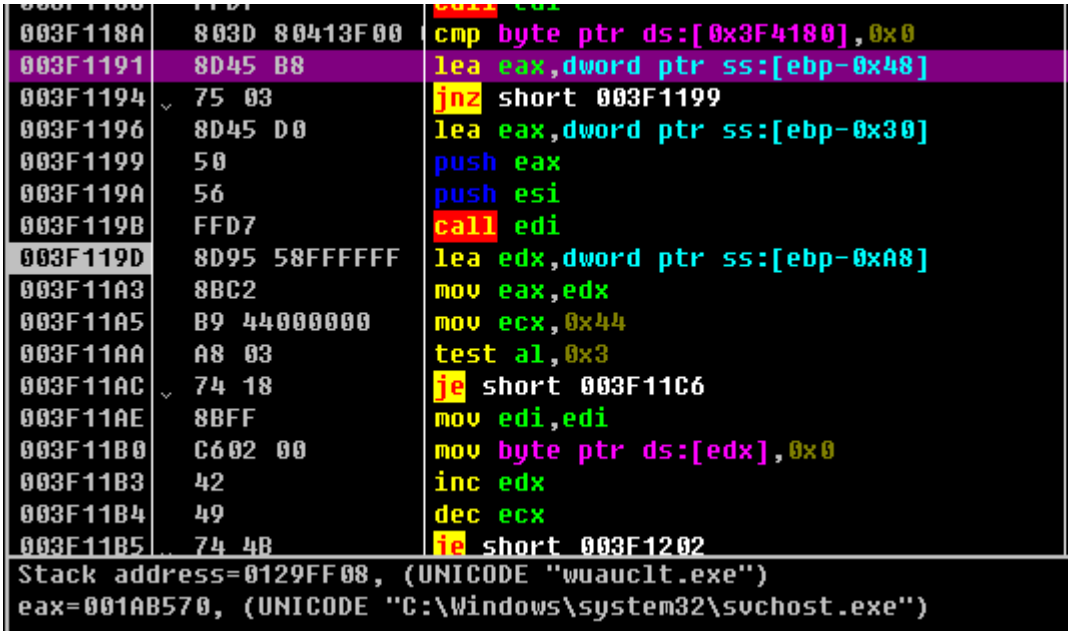
The vulnerability exploit is executed:

```
003F1982   56               push esi
003F1983   FFD7             call edi
003F1985   68 30750000      push 0x7530
003F198A   53               push ebx
003F198B   FF15 5C303F00    call dword ptr ds:[0x3F305C]
003F1991   68 C8000000      push 0xC8
003F1996   FF15 60303F00    call dword ptr ds:[0x3F3060]
003F199C   53               push ebx
003F199D   FF15 3C303F00    call dword ptr ds:[0x3F303C]
003F19A3   5F               pop edi
003F19A4   5E               pop esi
edi=015C1C00
```

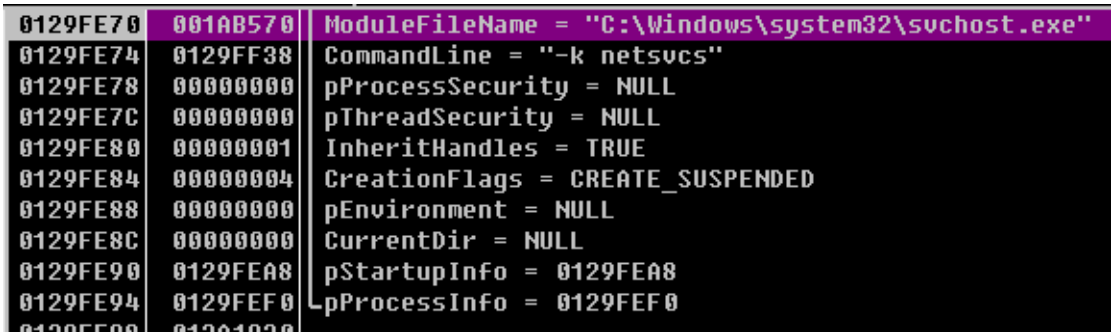
The following processes are retrieved:

```
003F14A4   B9 8C303F00      mov ecx,0x3F308C      ASCII "uiSeAgnt.exe"
003F14A9   E8 92000000      call 003F1540
003F14AE   B9 9C303F00      mov ecx,0x3F309C      ASCII "PtSessionAgent.exe"
003F14B3   E8 88000000      call 003F1540
003F14B8   B9 B0303F00      mov ecx,0x3F30B0      ASCII "PwmSvc.exe"
003F14BD   E8 7E000000      call 003F1540
003F14C2   B9 BC303F00      mov ecx,0x3F30BC      ASCII "coreServiceShell.exe"
003F14C7   E8 74000000      call 003F1540
```

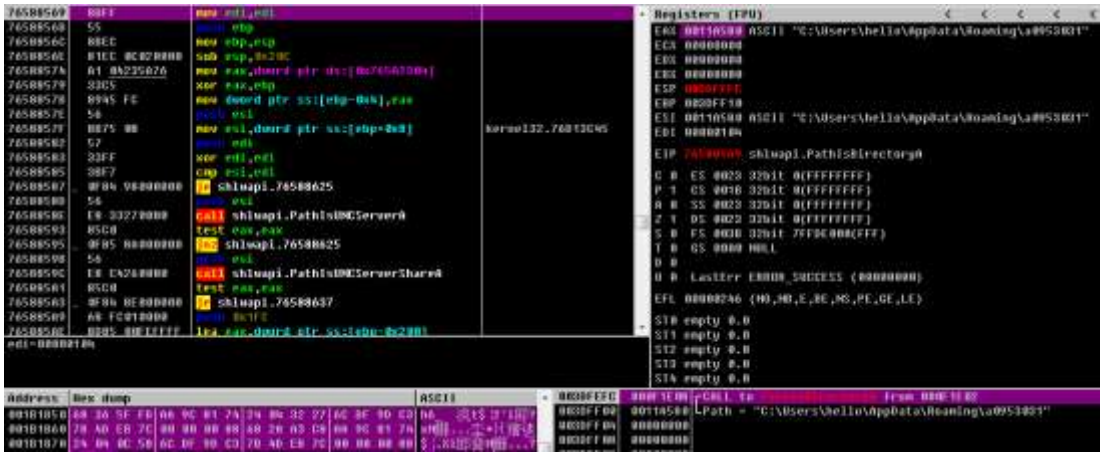
If the preceding processes are found, a suspended wuauclt.exe is created; otherwise, svchost.exe is created and "-k netsvcs" is passed to it. You can see that the value at 0x3F4180 is checked during process creation. If there is antivirus software, the value is set to 1; otherwise, it is set to 0.



The C:\Windows\system32\svchost.exe process with the "-k netsvcs" parameter is shown as follows:



The sample check whether this directory exists. If not, the sample creates the process.



The global mutex, Global\{DAN6J0-ae000000d2000000e100}, is created:

```
003DFD8C   000F168D  ┌CALL to CreateMutexA from 000F1687
003DFD90   003DFF00  │pSecurity = 003DFF00
003DFD94   00000000  │InitialOwner = FALSE
003DFD98   0011A260  └MutexName = "Global\{DAN6J0-ae000000d2000000e100}"
003DFD9C   626F6C47
```

The registry entry of the sample is retrieved:

```
003DFE84   0010427F  ┌CALL to RegQueryValueExA from 00104279
003DFE88   000000D0  │hKey = 0xD0
003DFE8C   0011D578  │ValueName = "ae000000"
003DFE90   00000000  │Reserved = NULL
003DFE94   00000000  │pValueType = NULL
003DFE98   0011A818  │Buffer = 0011A818
003DFE9C   003DFEA0  └pBufSize = 003DFEA0
003DFEA0   000009AC
```

- Running of child processes:

The sample hooks a function:

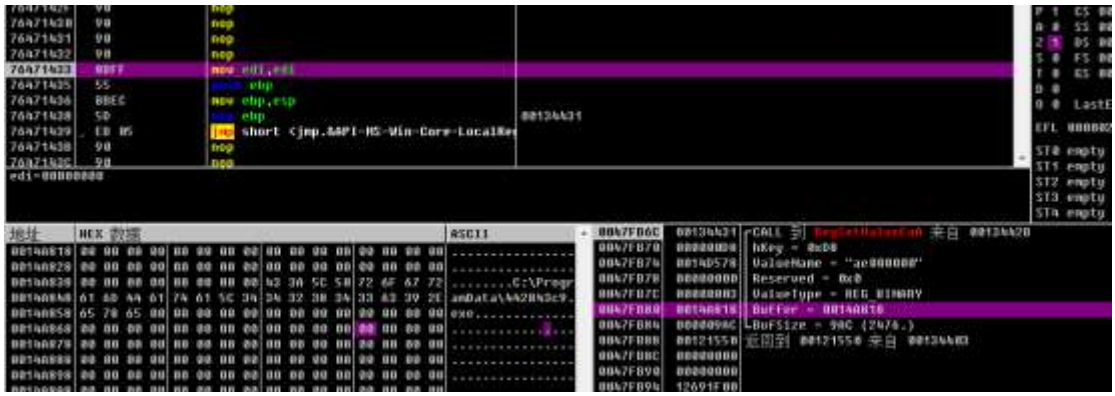
Two threads are created.

Thread 1: Registers a callback function to monitor changes to the registry entry and updates the following browser version.

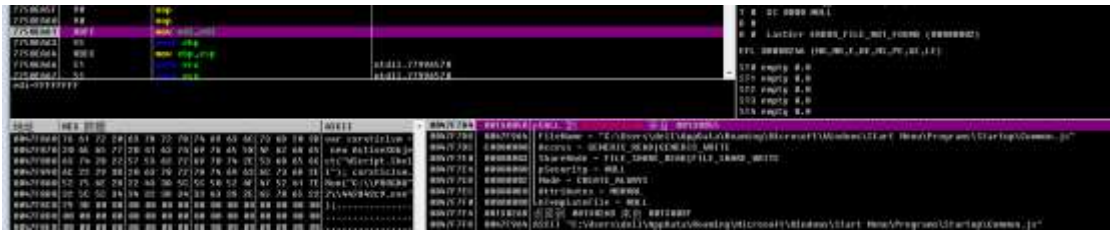
```
Address   ASCII dump
0025F410  ....厝0.豉■.?..iexplore.exe|firefox.exe|chrome.exe|opera.exe|br
0025F450  owser.exe|dragon.exe|epic.exe|sbrender.exe|vivaldi.exe|maxthon.e
0025F490  xe|ybrowser.exe|microsoftedgecp.exe.\∠澗d■.■R0.?..■...荟%.?e■.
```

Thread 2: Performs operations concerning network communications.

The sample sets the registry key HKCU\software\Microsoft\windows.



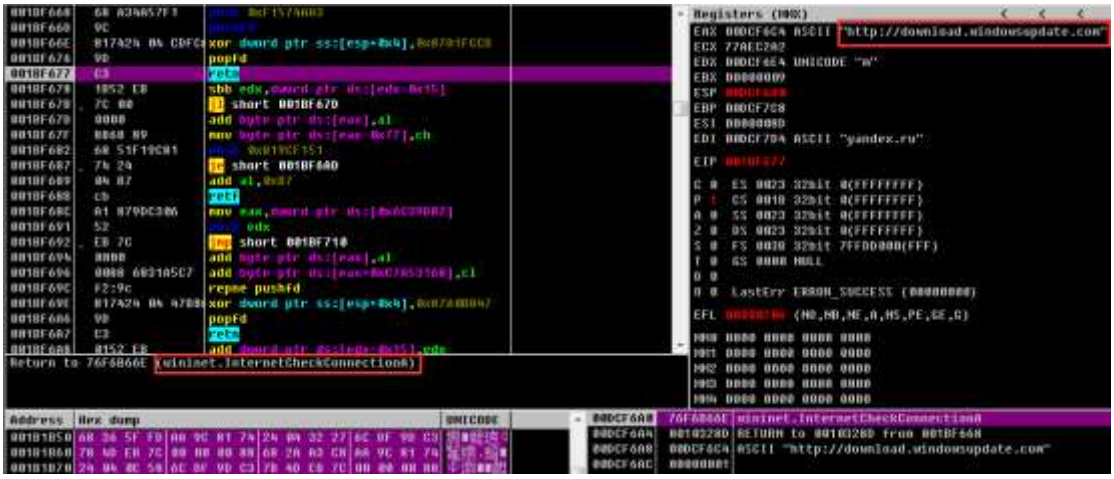
The sample creates a JavaScript file and copies it to the startup folder to set automatic startup items.



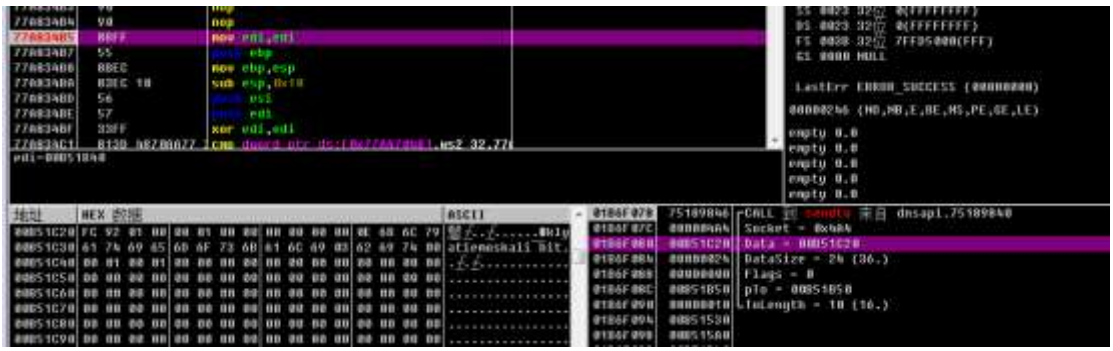
Code for setting automatic startup items in the JavaScript script is as follows:

```
var mxakulvhjn = new ActiveXObject("WScript.Shell");
  mxakulvhjn.Run("C:\\PROGRA~2\\442843c9.exe");
```

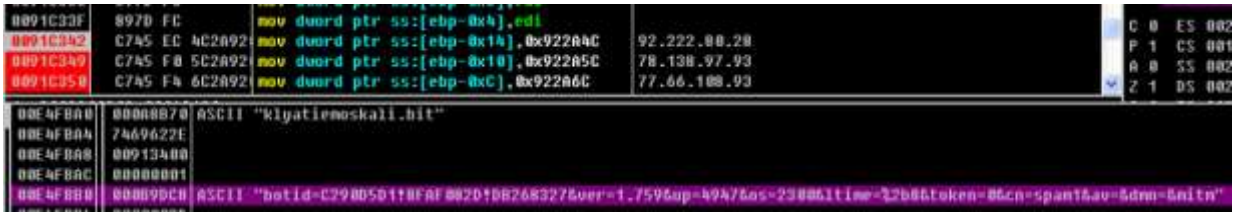
It accesses windowsupdate.com to test network connectivity.



It uses the NMC technique to initiate a DNS request to the DNS server, querying the address of the C&C server.



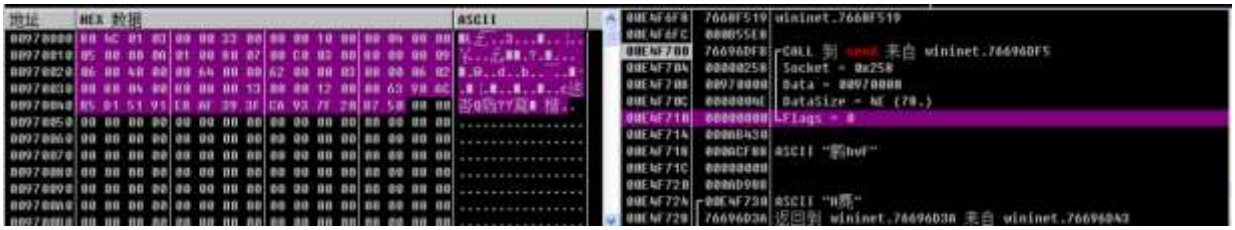
It obtains operating system information, types it as the setting of the "get" parameter, and sends it to the C&C server.



Certain parameters included in the registration package are described as follows:

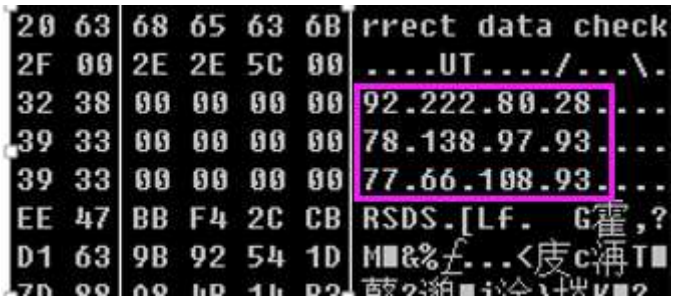
Botid	User ID generated randomly
Ver	Version number of the sample (1.759)
Up	Local timestamp when the upload occurs
Os	Operating system version
Ltime	Time zone (+8) of the user
Av	Antivirus software used by the user

The "get" parameter will be encrypted before being sent to the C&C server.



(5) Network Behaviors

The sample queries the address of the C&C server from three DNS servers: 92.222.80.28, 78.138.97.93, and 77.66.108.93.



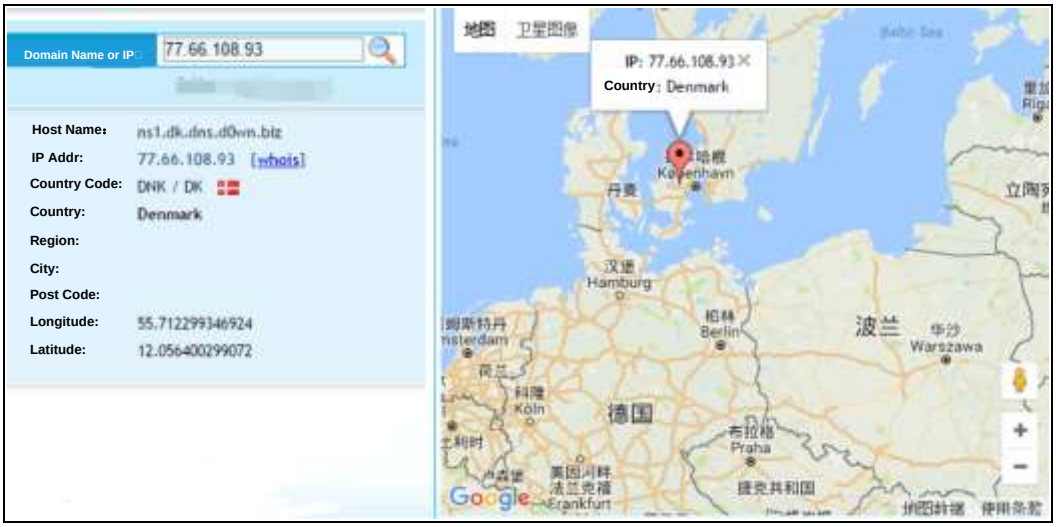
Using a .bit (with the NMC technique) domain name, this C&C server is low-cost and difficult to track and control.
Namecoin is a decentralized domain name system that provides similar functions as traditional DNS providers. It uses a .bit domain name. Unlike traditional DNS providers, Namecoin shares the DNS lookup table via a point-to-point network. In this case, as long as the Namecoin server software runs in the network, domain names can be queried, preventing network censorship.

(6) Launch Mode

By setting startup items and creating the Common.json script on the Start menu, this sample completes automatic startup upon the host restart, achieving persistent attacks.

Attack Location

The sample queries the region where each of three DNS servers (92.222.80.28, 78.138.97.93, and 77.66.108.93) used by the actual C&C server is located.



Recommended Solution

1. NSFOCUS Detection Services

(1) NSFOCUS engineers provide onsite detection services.

(2) NSFOCUS provides online cloud detection services. You can visit the following link to apply for the trial use of NSFOCUS Threat Analysis Center (TAC): **https://cloud.nsfocus.com/#/krosa/views/initcdr/productandservice?service_id=1018**

2. NSFOCUS Solutions for Removing Trojans

(1) Short-term service: NSFOCUS engineers provide the onsite trojan backdoor removal service (manual services + NIPS + TAC + Kingsoft V8+ terminal security system) to ensure that risk points are immediately eliminated in the network and the event impact is minimized. After the handling, an event analysis report is provided.

(2) Mid-term service: NSFOCUS provides 3- to 6-month risk monitoring and preventive maintenance inspection (PMI) services (NIPS + TAC + manual services) to eradicate risks and prevent events from recurring.

(3) Long-term service: NSFOCUS provides industry-specific risk mitigation solutions (threat intelligence + attack traceback + professional security service).

Conclusion

After using the embedded local system vulnerability exploitation module to escalate its privileges, this sample, with system-level privileges, steals user login credentials of the online banking business to cause harm.

As this sample provides various anti-debugging and analysis and detection means, common antivirus software regards it secure and passes it through. Also, it is able to escape common sandbox detection. Therefore, this sample is dangerous to users. Using .bit domain names for network communication, this sample is more covert and anonymous, making it difficult to track.

Appendix

The encrypted data in the original file of the sample is decrypted with the key 0x8D as follows:

```
AddMandatoryAce
ADVAPI
Advapi32.dll
advapi32.dll
ws2_32.dll
WPUCloseEventW
PUCloseSocketHandle
WPUCreateEvent
WPUCreateSocketHandle
WPUFDIsSet
WPUGetProviderPath
WPUModifyIFSHandle
WPUPostMessage
WPUQueryBlockingCallback
WPUQuerySocketHandleContext
WPUQueueApc
WPUResetEvent
WPUSetEvent
WPUOpenCurrentThread
WPUCloseThread
WSPStartup
> %1\r\ndel %0
software\\microsoft\\windows\\currentversion\\run
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/echo
rundll32.exe shell32.dll, ShellExec_RunDLL %s
Microsoft\\Microsoft AntimalwareSoftware\\Coranti
Software\\risingSoftware\\TrendMicroSoftware\\Symantec
Software\\ComodoGroup
Software\\Network Associates\\TVD
Software\\Data Fellows\\F-SecureSoftware\\Eset\\Nod
Software\\Softed\\ViGUARD
Software\\Zone Labs\\ZoneAlarm
Software\\Avg
```

Software\\VBA32
Software\\Doctor Web
Software\\G Data
Software\\Avira
Software\\AVAST
Software\\Avast
Software\\KasperskyLab\\protectedSoftware\\BitdefenderSoftware\\Panda
SoftwareSoftware\\Sophos.bat\\\\.\\%C:
|$$$}rstuvwxyz{$$$$$$$>?@ABCDEFGHIJKLMNOPQRSTUVW$$$$$$XYZ[\\]^_`abcdefg
hijklmnopq
conhost
CreateProcessInternalW
ConvertStringSecurityDescriptorToSecurityDescriptorW
Content-Type: multipart/form-data; boundary=----------%s\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Host: %s\r\n%d.%d.%d.%d
%d.%d.%d.%d.%x
%temp%\\debug_file.txt
[%u][%s:%s:%u][0x%x;0x%x]%sDnsFlushResolverCache
.
dnsapi.dll
DnsGetCacheDataTable.dll.exe
download.windowsupdate.com
vk.com
yandex.ru
HTTP/1.1
https://http://%s
IsWow64Process
kernel
kernel32.dll
LdrGetProcedureAddress
Microsoft
NtAllocateVirtualMemory
CLOSED
LAST_ACKTIME_WAIT
DELETE_TCB
LISTEN
SYN_SENTSYN_RCVDESTAB
FIN_WAIT1
FIN_WAIT2
CLOSE_WAIT
CLOSING
TCP\t%s:%d\t%s:%d\t%s\n
netstat\nProto\tLocal address\tRemote address\tState\n

ntdll.dll
NtResumeProcess
NtSuspendProcess
\\\\?\\globalroot\\systemroot\\system32\\drivers\\null.sys
NtWriteVirtualMemory
openRegisterApplicationRestart
RtlCreateUserThread
ResetSR
RtlComputeCrc32
rundll32
SeDebugPrivilege
SystemDrive\\StringFileInfo\\%04x%04x\\ProductName
software\\microsoft\\windows nt\\currentversion\\winlogon
shell
Sleep
srclient.dll
SeShutdownPrivilege
\"%s\"
%d\t%s\ntaskmgr\nPID\tProcess name\nnet user\n
the computer is joined to a domain\n..
\\VarFileInfo\\Translation
GET%windir%\\system32\\%windir%\\syswow64\\POST*.exe
Low
%SystemDrive%
SYSTEM%02x%s:Zone.
Identifier
GetProcessUserModeExceptionPolicy
etProcessUserModeExceptionPolicy
%ws\\%ws\n%x
WORKGROUP
HOME
Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\ExplorerDisableCurrentUser
Run%s.datsoftware\\microsoft\\windows
%OS%_%NUMBER_OF_PROCESSORS%S
S:(ML;;NRNWNX;;;LW)D:(A;;GA;;;WD)
S:(ML;;NRNWNX;;;LW)D:(A;;GA;;;WD)(A;;GA;;;AC)
\\\\.\\AVGIDSShim
FFD3\\\\.\\NPF_NdisWanIpc:\\sample\\pos.exe
ANALYSERS
SANDBOX
VIRUS
MALWARE
FORTINET
MALNETVM

c:\\analysis\\sandboxstarter.exe
c:\\analysis
c:\\insidetm
c:\\windows\\system32\\drivers\\vmmouse.sys
c:\\windows\\system32\\drivers\\vmhgfs.sys
c:\\windows\\system32\\drivers\\vboxmouse.sys
c:\\iDEFENSE
c:\\popupkiller.exe
c:\\tools\\execute.exe
c:\\Perl
c:\\Python27
api_log.dll
dir_watch.dll
pstorec.dll
dbghelp.dll
Process32NextW
Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Zones\\31406.bit
MiniDumpWriteDump\r\nReferer: %s\r\n
\\Google\\Chrome\\User Data\\Default\\Cachevar
%s = new ActiveXObject("WScript.Shell"); %s.Run("%s");
IntelPowerAgent32
%OS%_%NUMBER_OF_PROCESSORS%
%s\cmd.exe
ComSpecConsoleWindowClass.exe
kernel32.dll
ntdll.dll
ZwQuerySystemInformation
ZwAllocateVirtualMemory
PsLookupProcessByProcessId
PsReferencePrimaryTokenClassWindow
open "%s" -q%windir%\\system32\\sdbinst.exe
/c "start "" "%s" -d"
%windir%\\system32\\sndvol.exe
"%s" -u /c "%s\\SysWOW64\\SysSndVol.exe /c "start "" "%s" -d"
"%temp%\\%u
%u.tmp
Wow64DisableWow64FsRedirection
Wow64RevertWow64FsRedirection
runas.exe
%systemroot%\\system32\\svchost.exe
%systemroot%\\system32\\wscript.exe
snxhk.dll
sbiedll.dll
/c start "" "%s" " "

```
cmd.exe
runas
--crypt-test:3
It work's!


--vm-test
```

The strings included in code injected in **svchost.exe** are decrypted with 0x8D as follows:

```
AddMandatoryAce
ADVAPI
Advapi32.dlladvapi32.dllws2_32.dll
WPUCloseEvent
WPUCloseSocketHandleWPUCreateEvent
WPUCreateSocketHandle
WPUFDIsSet
WPUGetProviderPath
WPUModifyIFSHandle
WPUPostMessage
WPUQueryBlockingCallbackWPUQuerySocketHandleContext
WPUQueueApc
WPUResetEvent
WPUSetEvent
WPUOpenCurrentThreadWPUCloseThread
WSPStartup
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/echo
> %1\r\ndel %0
rundll32.exe shell32.dll, ShellExec_RunDLL %s
software\\microsoft\\windows\\currentversion\\run
Microsoft\\Microsoft AntimalwareSoftware\\Coranti
Software\\risingSoftware\\TrendMicroSoftware\\Symantec
Software\\ComodoGroup
Software\\Network Associates\\TVD
Software\\Data Fellows\\F-SecureSoftware\\Eset\\Nod
Software\\Softed\\ViGUARD
Software\\Zone Labs\\ZoneAlarm
Software\\Avg
Software\\VBA32
Software\\Doctor WebSoftware\\G DataSoftware\\Avira
Software\\AVAST Software\\Avast
Software\\KasperskyLab\\protected
Software\\Bitdefender
Software\\Panda SoftwareSoftware\\Sophos.bat|$$$}rstuvwxyz{$$$$$$$>
@ABCDEFGHIJKLMNOPQRSTUVW$$$$$$XYZ[\\]^_`abcdefghijklmnop
```

q
\\\\.\\%C:
conhost
CreateProcessInternalW
ConvertStringSecurityDescriptorToSecurityDescriptorWContent-Type: application/x-www-form-urlencoded\r\n
Content-Type: multipart/form-data; boundary=----------%s\r\n
Host: %s\r\n%d.%d.%d.%d
%d.%d.%d.%d.%x
%temp%\\debug_file.txt
[%u][%s:%s:%u][0x%x;0x%x] %sDnsFlushResolverCache
.
dnsapi.dll
DnsGetCacheDataTable.dll.exedownload.windowsupdate.com
vk.com
yandex.ru
HTTP/1.1https://http://%s
IsWow64Process
kernel
kernel32.dllLdrGetProcedureAddress
Microsoft
NtAllocateVirtualMemory
CLOSED
LAST_ACKTIME_WAIT
DELETE_TCB
LISTEN
SYN_SENTSYN_RCVDESTAB
FIN_WAIT1
FIN_WAIT2
CLOSE_WAIT
CLOSING
TCP\t%s:%d\t%s:%d\t%s\n
netstat\nProto\tLocal address\tRemote address\tState\n
ntdll.dll
NtResumeProcess
NtSuspendProcess\\\\ \\globalroot\\systemroot\\system32\\drivers\\null.sys
NtWriteVirtualMemoryopenRegisterApplicationRestart
RtlCreateUserThread
ResetSR
RtlComputeCrc32
rundll32SeDebugPrivilegeSystemDrive
\\StringFileInfo\\%04x%04x\\ProductName
software\\microsoft\\windows nt\\currentversion\\winlogon
shell

Sleep
srclient.dllSeShutdownPrivilege
\"%s\"
%d\t%s\ntaskmgr\nPID\tProcess name\nnet user\n
the computer is joined to a domain\n..
\\VarFileInfo\\Translation
%windir%\\system32\\%windir%\\syswow64\\POST*.exe
%SystemDrive%\\
SYSTEM%02x%s:Zone.Identifier
GetProcessUserModeExceptionPolicy
SetProcessUserModeExceptionPolicy
%ws\\%ws\n
WORKGROUP
HOMEsoftware\\microsoft\\windowsSoftware\\Microsoft\\Windows\\CurrentVersion\\Policies\\
ExplorerDisableCurrentUserRun
%s.dat
%OS%_%NUMBER_OF_PROCESSORS%
S:(ML;;NRNWNX;;;LW)D:(A;;GA;;;WD)
S:(ML;;NRNWNX;;;LW)D:(A;;GA;;;WD)(A;;GA;;;AC)
\\\\.\\AVGIDSShim
FFD3\\\\.\\NPF_NdisWanIpc:\\sample\\pos.exe
ANALYSERS
SANDBOX
VIRUS
MALWARE
FORTINETMALNETVMc:\\analysis\\sandboxstarter.exec:\\analysisc:\\insidetmc:\\windows\\s
ystem32\\drivers\\vmmouse.sys
c:\\windows\\system32\\drivers\\vmhgfs.sys
c:\\windows\\system32\\drivers\\vboxmouse.sys
c:\\iDEFENSEc:\\popupkiller.exe
c:\\tools\\execute.exe
c:\\Perlc:\\Python27api_log.dll
dir_watch.dll
pstorec.dll
dbghelp.dll
Process32NextW
1406Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings\\Zones\\3
.bitMiniDumpWriteDump
\r\nReferer: %s\r\n
\\Google\\Chrome\\User Data\\Default\\Cache
var %s = new ActiveXObject("WScript.Shell"); %s.Run("%s");
GenuineIntelAuthenticAMDCentaurHauls7z
fnbqooqdaixfueangywblgabirdgvkewdyqgfqaioluesyrpryfkjerfsouemaxnavrkguxmcmhckwprun
urmhehclermtufwiyjbqhwlunbun

uumeowfjmerxppxrgaxukyx
PowerManager_M5VKII_%d
[type=ftp]\n[botid=%s]\n[proc=%s]\n[data=%s]\n
[type=pop3]\n[botid=%s]\n[proc=%s]\n[data=%s]\n
%OS%_%NUMBER_OF_PROCESSORS%
[type=post]\n[botid=%s]\n[url=%s]\n[ua=%s]\n[proc=%s]\n[ref=%s]\n[keys=%s]\n[data=%s]\
n
name=%s&ok=%s&id=%d&res_code=%d&res_text=%s_%x
name=%s&ok=%s&id=%d&res_code=%d&res_text=%s
botid=%s&ver=%s.%u&up=%u&os=%u&ltime=%s%d&token=%d&cn=%s&av=%s&dmn=%
s&mitm=%u
java.exe|javaw.exe|plugin-container.exe|acrobat.exe|acrod32.exe
tellerplus|bancline|fidelity|micrsolv|bankman|vanity|episys|jack
henry|cruisenet|gplusmain|silverlake|v48d0250s1Root|TrustedPeople|SMS|Remote
Desktop|REQUEST
TREASURE|BUH|BANK|ACCOUNT|CASH|FINAN|MONEY|MANAGE|OPER|DIRECT|RO
SPIL|CAPO|BOSS|TRADEactive_bc
----------%s\r\nContent-Disposition: form-data;
name=\"pcname\"\r\n\r\n%s!%s\r\n----------
%s\r\nContent-Disposition: form-data; name=\"file\"; filename=\"report\"\r\nContent-Type:
text/plain\r\n\r\n%s\r\n----------
----------%s--\r\n
%domain%deactivebc

inject
kill_os
loadactive_sk
deactive_sk
wipe_cookiesmitm_modmitm_script
mitm_geterr
get_keylog
get_sols!active_bc\[(\d+)\] (\S+) (\d+)
!deactive_bc\[(\d+)\]
!inject\[(\d+)\] (\S+)
!kill_os\[(\d+)\]
!get_keylog\[(\d+)\]!load\[(\d+)\] (\S+)!update\[(\d+)\] (\S+)
!wipe_cookies\[(\d+)\]
!active_sk\[(\d+)\] (\S+) (\d+)
!deactive_sk\[(\d+)\]
!mitm_mod\[(\d+)\] (\S+) (\d+) (\S+)!mitm_script\[(\d+)\] (\S+)
!mitm_geterr\[(\d+)\]
!get_sols\[(\d+)\]
ATCASH
ATLOCAL

CERTCERTX
COLVCRAIF
CRYPT
CTERM
SCREEN
INTER
ELBALOCAL
ELBAWEB
ELBAWEB
ELBAWEB
PUTTY
VNCVIEW
MCLOCAL
MCSIGN
OPENVPN
PIPEK
PIPEK
PIPEK
PIPEK
POSTSAP
chrome.dll
mxwebkit.dlldragon_s.dlliron.dllvivaldi.dll
nspr4.dll
nss3.dllbrowser.dll
Advapi32.dllrsaenh.dll
kernel32.dllIprivLibEx.dll
cryptui.dll
crypt32.dll
ntdll.dll
ssleay32.dllurlmon.dll
user32.dll
Wininet.dll
Ws2_32.dll
PSAPI.dll
NzBrco.dll
VirtualProtect
LoadLibraryExW
ZwQuerySystemInformationWSARecv
WSASend
ZwDeviceIoControlFile
URLDownloadToCacheFileW
URLDownloadToFileW
TranslateMessageSSL_get_fd
SSL_write

PFXImportCertStore
CryptEncryptCPExportKey
CreateProcessInternalW
CreateDialogParamW
GetClipboardDatagetaddrinfo
gethostbyname
GetAddrInfoExW
GetMessageA
GetMessageW
DeleteFileA
GetModuleBaseNameW
bad port value
can't find plug-in path
can't get bot path
can't download file
can't encrypt file
can't save inject config to filecan't get temp file
file is not valid PEcan't delete original file
can't replace original file
can't close handle
can't protect file
original file not found
can't execute file
can't create directory
can't unzip file #1
can't unzip file #2
mitm_mod is inactivehttpd.exe is anactive
microsoft.com
dropbox.com
KEYGRAB
PasswordTELEMACOScelta e Login dispositivo
TLQ Web
db Corporate Banking WebSecureStoreCSP - enter PIN
google.com
Software\\SimonTatham\\PuTTYreg.txt
Software\\Microsoft\\Internet Explorer\\MainTabProcGrowth
Temp\\Low
crc32[%x]
ACCT
AUTHINFO PASS
AUTHINFO USER
Authorization
:BA:[bks]
%X!%X!%08X

btc_path.txtbtc_wallet.dat
bitcoin\\wallet.dat
%s%s\\%u_cert.pfx
cmdline.txt
1.3.6.1.5.5.7.3.3
CodeSign\n
Software\\Microsoft\\Windows NT\\CurrentVersion
[del]
Default
.exeELBA5\\ELBA_dataftp://anonymous:ftp://%s:%s@%s:%d\n
HBPData\\hbp.profileHH:mm:ssdd:MMM:yyyy
I_CryptUIProtect\\exe\\
infected.exx%s%s\\%u_info.txt
[ins]
InstallDate
%02u.jpg%s\\%02d.jpgKEYLOG
%s\\keylog.txt
[TOKEN ON]
\n\n[%s (%s-%s) - %s (%s)]\n[pst]%s[/pst]
ltcd_path.txt
ltcd_wallet.dat
litecoind\\wallet.dat
ltc_path.txtltc_wallet.dat
litecoin\\wallet.dat\\MacromediaMultiCash@Sign
C:\\Omikron\\MCSign
[ML][MR]Global\\{4C470E-%08x-%08x-%08x}
Global\\{DAN6J0-%s}
noneopera.exe
PASS
password.txt\\\\.\\pipe\\%s
pop3://%s:%s@%s:%d\n%PROCESSOR_ARCHITECTURE%Referer
[ret]
%08x\\system32\\rstrui.exe
\\scrs\\send%s%s%s%d%s:%s
sysinfo.txt
[tab]
data.txt<unnamed>
<untitled>
update
USER
User-agent
vkeys
%x\r\n
\r\n%x%x%x.tmp

*.txt
%02x%2b
torrent
-config config.vnc
--config
config.ovpn
data.txt[type=post]\n
CreateFileW
pos.exe
bank.exePOS
secure.
.mozgoogle.com
CertVerifyCertificateChainPolicyCertGetCertificateChain
SSL_AuthCertificateHook
USERNAMESoftware\\ESET\\ESET Security\\CurrentVersion\\Info
C8FFAD27AE1BBE28BE24DDF20AF36EF901C609968930ED82CEFBC64808BA34102C4F
ABA0560523FB4CCBF33684F77C8401DFB
3A7D2D598E872DD78033E7F900B78A0C710CDF0941662FF7745A435D4BC18D5661E05
82B21B2DB8FCA1C0CA3401D0FC9F051
85A558AB6A76A010F606CD77B35A480B6B7176F0903299B91F1BBD141B4D33615849C
35557357DAB819BC3D4A8722BB433DE
B66C7A326BE859BD94930331B37DEE6EF4C475EA4B33DE4699FFDBCD34E196E19FE6
30E631D2C612705048620183BCF56709B
484A4380C4B00D8D94D131C31DB53AE6BCDCCC14131BAC99A68C59A604D0AE9116E
9196F7FA3EA5F86F67E9B175CC09D3E17
997728B7D
10001
get=1
COMPNAMEAppDataDir
updfiles\\upd.ver
updfiles\\lastupd.ver
SYSTEM\\CurrentControlSet\\services\\Avg\\SystemValues
Local AppData
Avg2015
Avg2014
Avg2013
Avg2012
Avg2011
update
Software\\Microsoft\\Windows\\CurrentVersion\\explorer\\Browser Helper
Objects\\{8CA7E745-EF75-4E7B-BB86-
8065C0CE29CA}
Software\\Microsoft\\Windows\\CurrentVersion\\explorer\\Browser Helper
Objects\\{BB62FFF4-41CB-4AFC-BB8C-

2A4D4B42BBDC}
Software\\Microsoft\\Internet Explorer\\MainEnable Browser Extensions
httpd.exe
%s\\httpd.exe
connect
data\\index.php
logs\\error.log
error.log
< \n';\n$bot_id = '
$bot_net = '$key_log_file = '
$process_file = '
127.0.0.1
Listen %s:%u\n
conf\\httpd.confSSL_PORT%u>\n
[type=post]\n
[type=screen]\n
[type=knock]\n
74 834E0440B832FFFFFF
74 834E04405F5EB832FFFFFF
DEBUG
memory.dmp
config.xml
php5ts.dll
zend_stream_fixup
zend_compile_file
index.php
config.php
content.php
iexplore.exe|firefox.exe|chrome.exe|opera.exe|browser.exe|dragon.exe|epic.exe|sbrender.exe|viv
aldi.exe|maxthon.exe|ybr
owser.exe|microsoftedgecp.exe
InternetQueryDataAvailable
InternetReadFileInternetReadFileExA
InternetReadFileExW
InternetSetStatusCallbackA
InternetSetStatusCallbackW
HttpSendRequestAHttpSendRequestExA
HttpSendRequestExW
HttpSendRequestW\r\n0\r\n\r\n
.rdata
\r\n\r\nHTTP/1.
Transfer-Encoding
chunked
Content-Length

close
Proxy-ConnectionHostAccept-Encoding
x-xss-protectionx-content-security-policy
x-frame-options
x-content-type-options
If-Modified-Since
If-None-Match
content-security-policy
x-webkit-cspConnection
http://
https://NSS layer
Content-TypeBasic
PR_ClosePR_Connect
PR_GetNameForIdentity
PR_Read
PR_SetError
PR_WriteReferer:
Accept-Encoding:\r\n1406SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3
data_after\ndata_before\n
data_enddata_inject\n
set_url %BOTID%
%BOTNET%InternetCloseHandle

HTMLc:\\inject.txt
Dalvik/1.6.0 (Linux; U; Android 4.1.2; GT-N7000 Build/JZO54K)
xxx_process_0x%08x
Common.js

About NSFOCUS

==========

NSFOCUS IB is a wholly owned subsidiary of NSFOCUS, an enterprise application and network security provider, with operations in the Americas, Europe, the Middle East, Southeast Asia and Japan. NSFOCUS IB has a proven track record of combatting the increasingly complex cyber threat landscape through the construction and implementation of multi-layered defense systems. The company's Intelligent Hybrid Security strategy utilizes both cloud and on-premises security platforms, built on a foundation of real-time global threat intelligence, to provide unified, multi-layer protection from advanced cyber threats.

For more information about NSFOCUS, please visit:

http://www.nsfocusglobal.com.

QR code of NSFOCUS at Sina Weibo

QR code of NSFOCUS at WeChat