HTML Application files are being used to distribute Smoke Loader malware

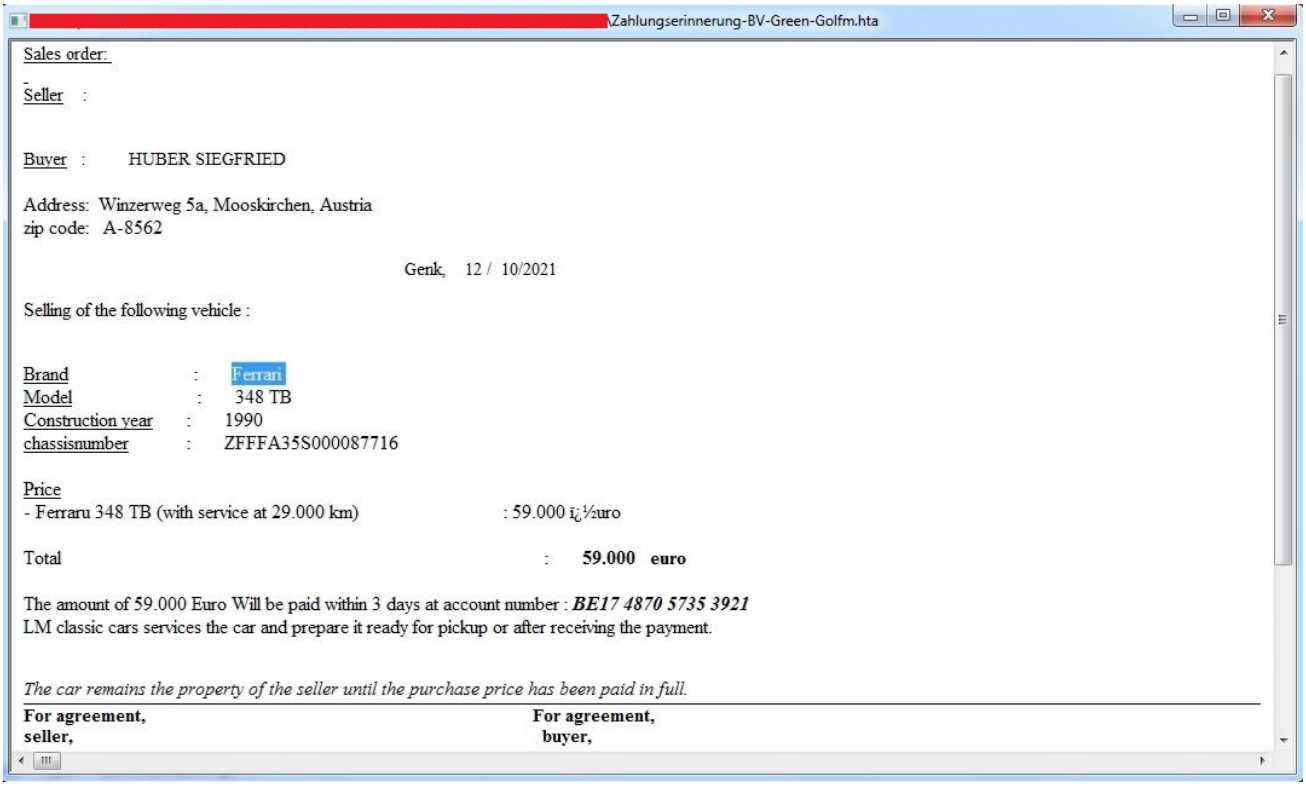
securitynews.sonicwall.com/xmlpost/html-application-hta-files-are-being-used-to-distribute-smoke-loader-malware/

June 21, 2022

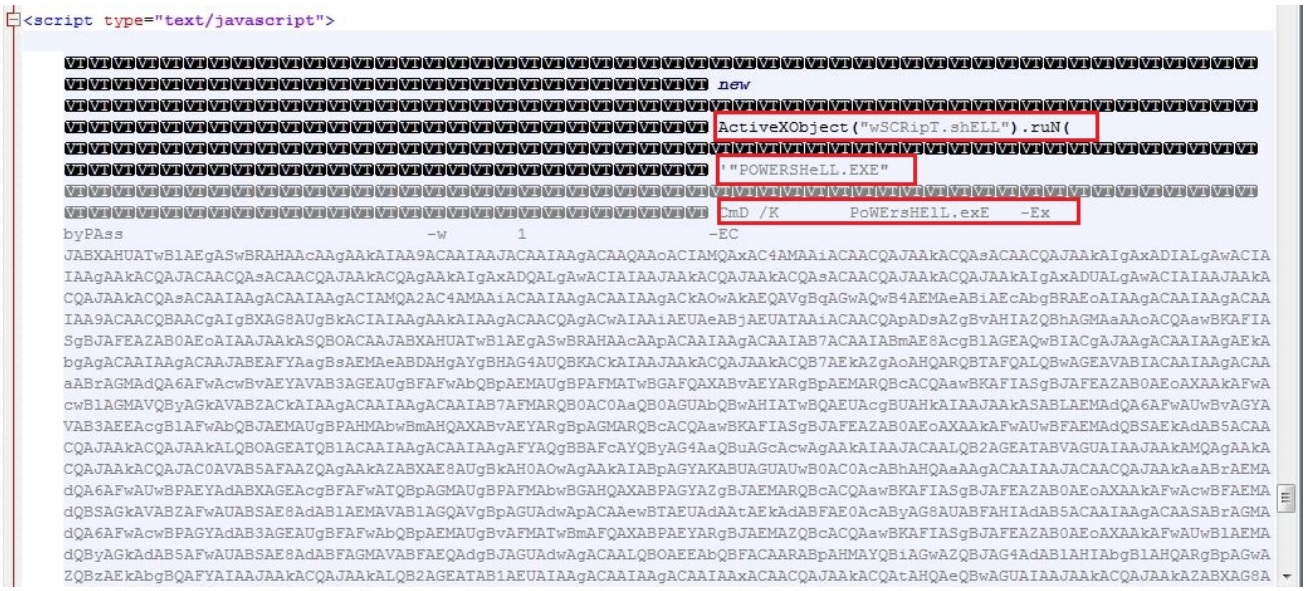
Threat actor always targets under the radar file types to deliver malware to the victim's machine. HTML Applications (HTA) files are known as less suspicious file types by various security providers. SonicWall Capture Labs Threat Research team has observed an HTA file inside an archive is being delivered to the victim's machine, which further downloads and executes Smoke Loader malware.

Infection Cycle:

The archive file name is in German "Zahlungserinnerung-BV-Green-Golfm.zip" acted as a payment reminder for the victim. The HTA file has HTML code to display service estimation by "LM Classic Cars" for Ferrari 348 TB for an Autria customer, additionally it includes JavaScript code to download malware using PowerShell script:



The JavaScript code executes the PowerShell executable which further executes another instance of the PowerShell executable using Command Prompt:



The PowerShell script contains code to perform below actions on MS Office files:

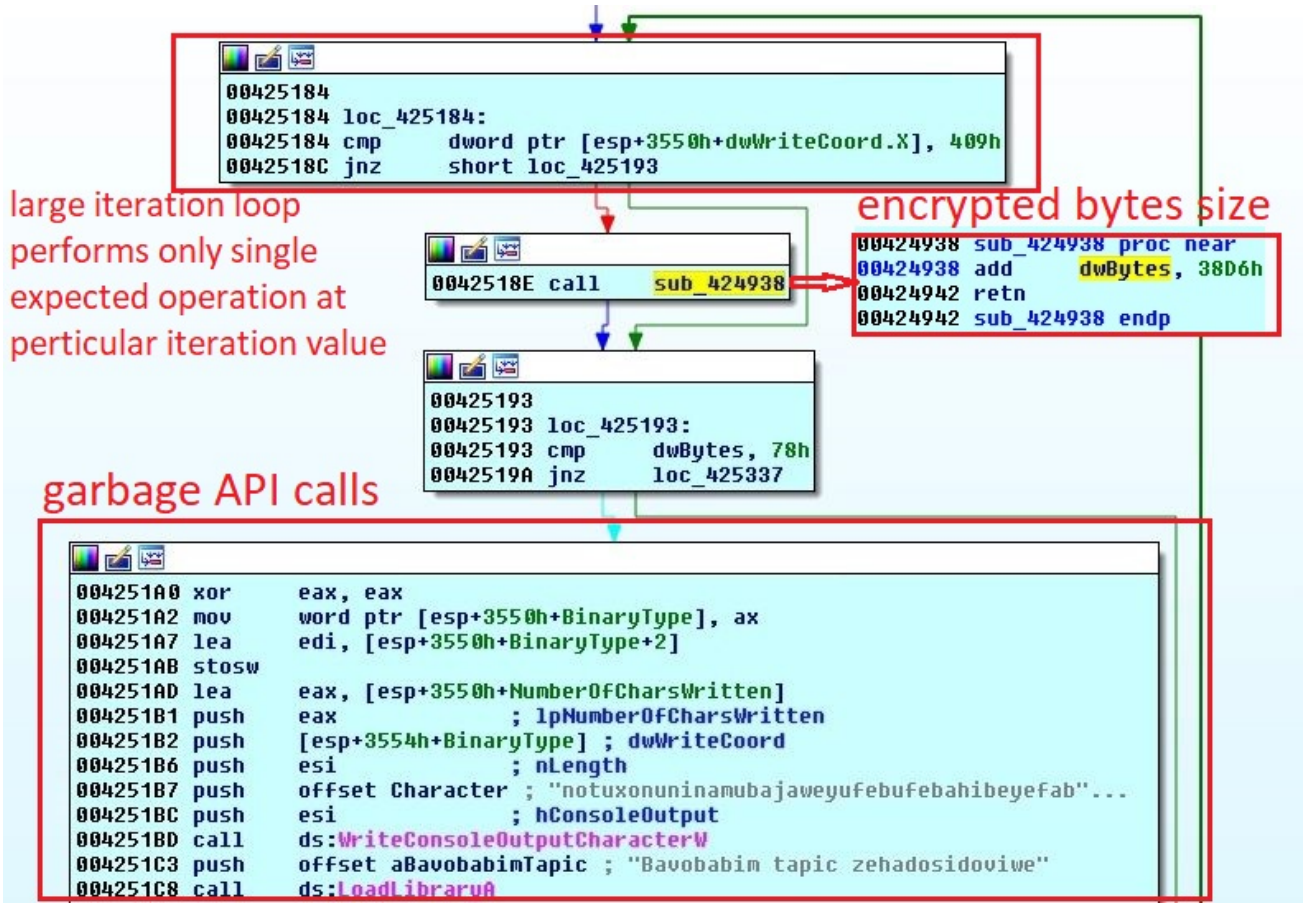
- Enables all macros
- Disable protected view for files belongs to internet zone
- Disable protected view for attachments opened in Outlook
- Disable protected view for files in unsafe locations

The PowerShell downloads malware from URL h[t][t]p://www.trimm.at/error/upx.exe

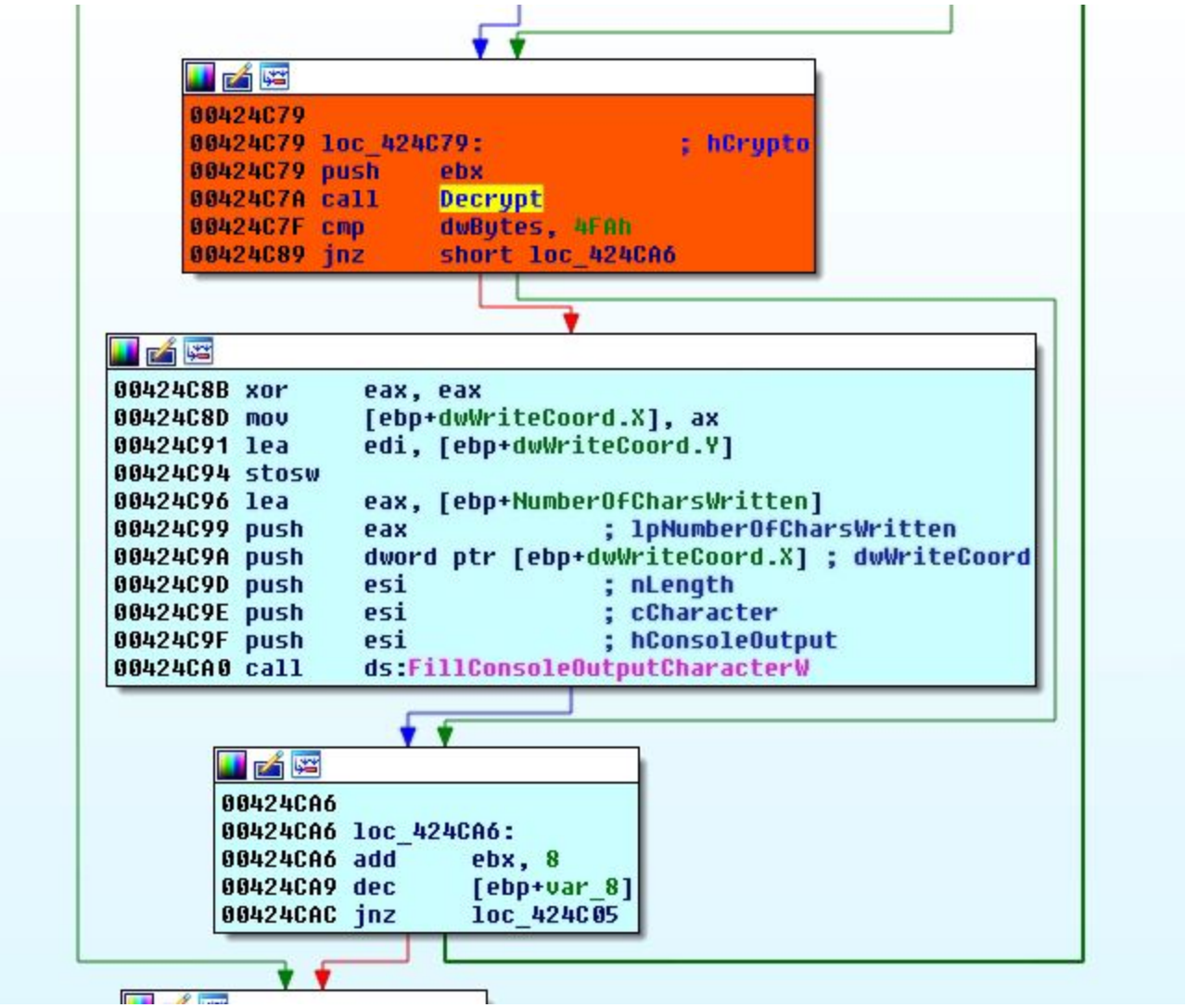
The Smoke Loader malware works in multi stages and layers. It uses code obfuscation, anti debugging, anti VM and Living of The Land techniques. The malware makes sure that a memory dump should not expose its intention at any point of time.

First Stage Executable

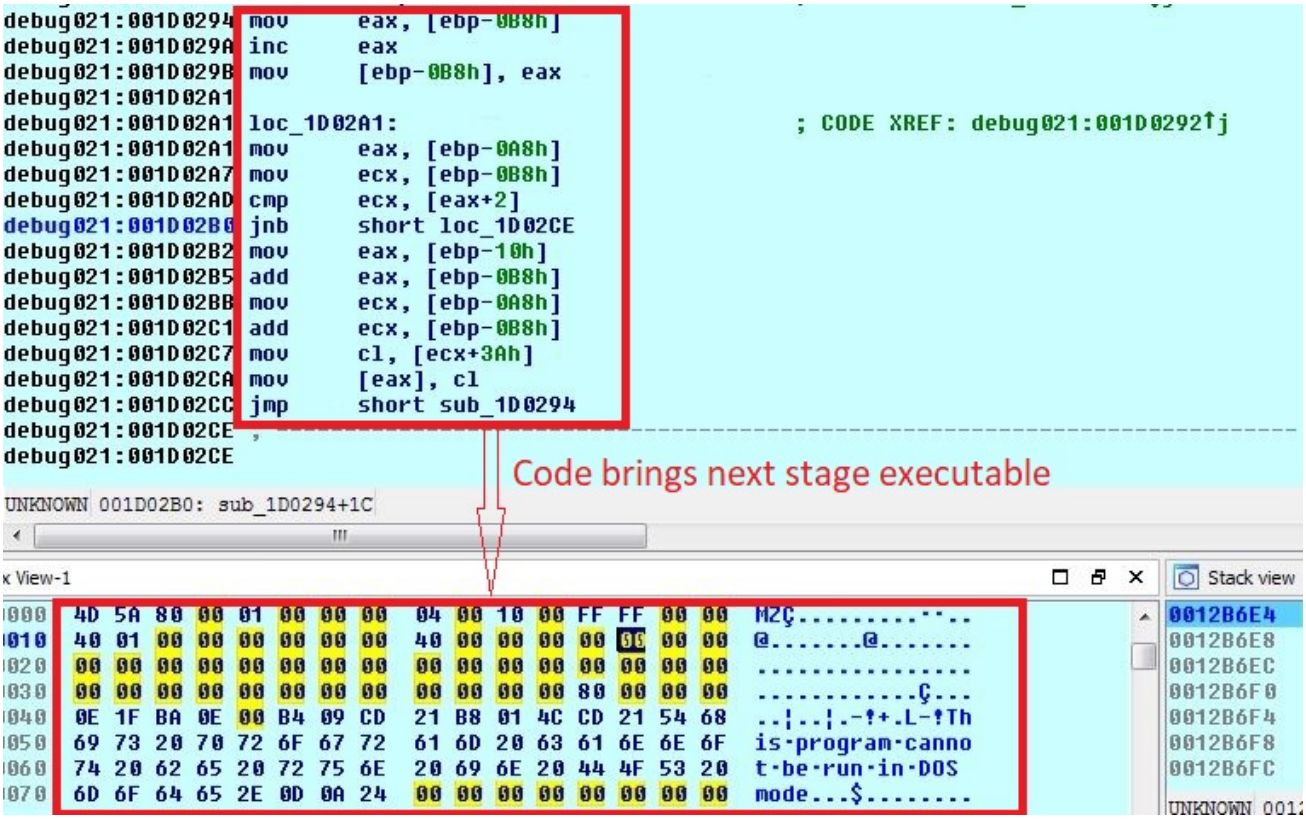
The first stage executable is highly obfuscated, it contains large loops with garbage API calls followed by a conditional jump. The malware uses opaque predicate technique as control never goes to garbage API calls, they are just kept to make analysis difficult. In a long iterations loop, only few operations are actually required by the malware which are executed on a particular iteration. The below iteration loop is intended to calculate the encrypted bytes size at 0x40Ath iteration:



The malware decrypts the shellcode into memory which further brings second stage executable:



The shellcode uses PEB_LDR_DATA from Process Environment Block, iterates through InLoadOrderModuleList to get the API addresses. The shellcode decrypts next stage executable in memory and does process hollowing to replace current process from the address space and starts execution of new process from entry point:



Second Stage Executable:

Second stage executable code is full of techniques used to investigate the controlled environment execution.

Anti-Debug

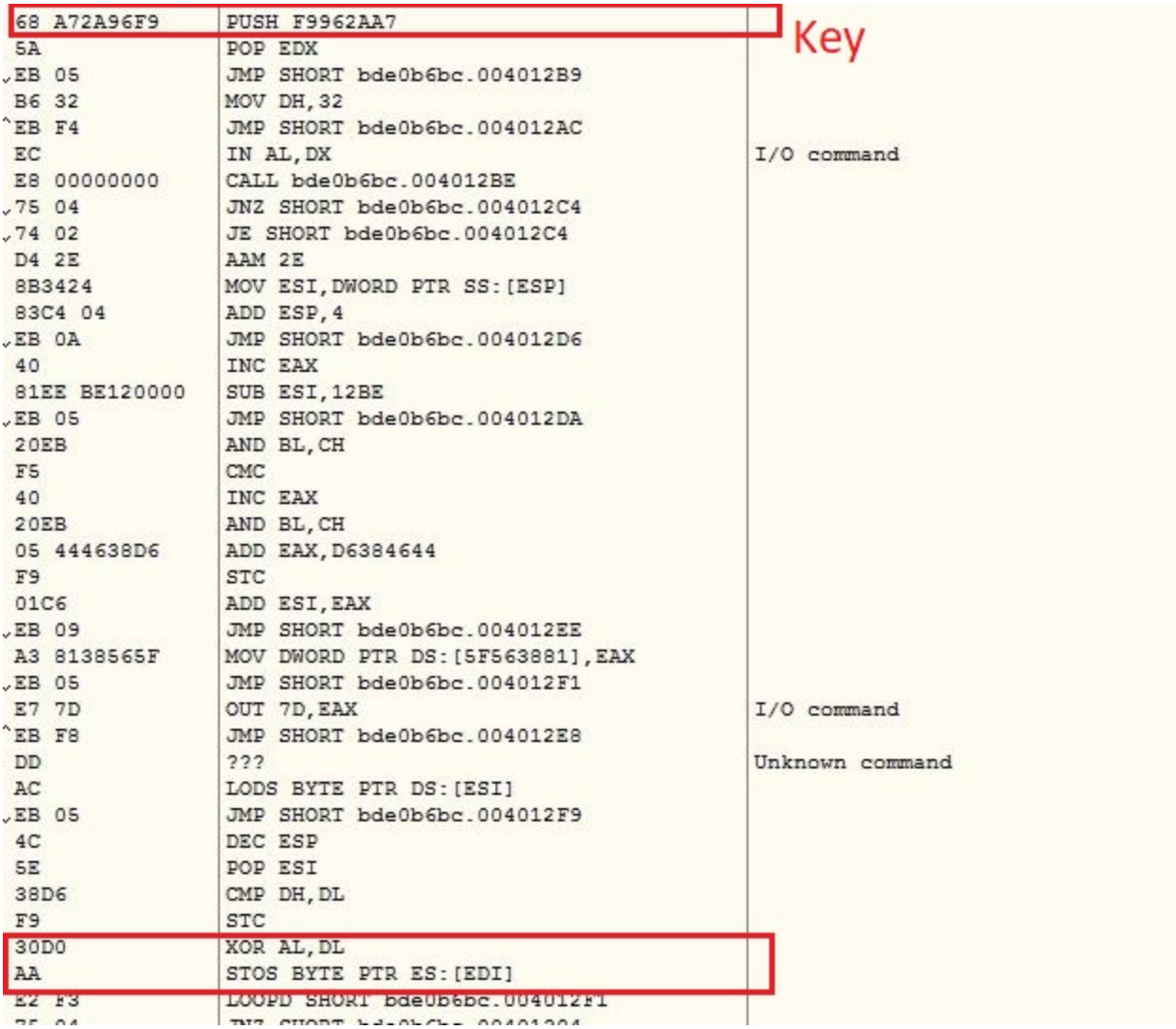
Checking the *BeingDebugged* and *NtGlobalFlag* in Process Environment Block is common across the malware. Here the tricky part is, instead of branching the code based on the flag values, the malware uses the flag values to compute a jump offset. If the malware is running inside a debugger then it will compute a invalid address which makes an impression of corrupted file to the researcher:

```
0FB648 02       MOVZX ECX,BYTE PTR DS:[EAX+2]       BeingDebugged
EB 05           JMP SHORT bde0b6bc.0040304A
CC              INT3
ED              DB ED
1AD6            SBB DL,DH
F9              STC
83C1 01         ADD ECX,1                           Adding into BeingDebugged Value
EB 0C           JMP SHORT bde0b6bc.0040305B
CE              DB CE
D8              DB D8
27              DB 27                               CHAR '''
96              DB 96
47              DB 47                               CHAR 'G'
85              DB 85
89C8            MOV EAX,ECX
EB 05           JMP SHORT bde0b6bc.0040305E
AC              DB AC
B2              DB B2
EB F8           JMP SHORT bde0b6bc.00403055
9F              LAHF
EB 0D           JMP SHORT bde0b6bc.0040306D
B9              DB B9
10              DB 10
A5              DB A5
CD              DB CD
B9 412F0000     MOV ECX,2F41
EB 05           JMP SHORT bde0b6bc.00403070
D4 4F           AAM 4F
EB F5           JMP SHORT bde0b6bc.00403064
E4              DB E4
EB 01           JMP SHORT bde0b6bc.00403073
43              INC EBX
F7E1            MUL ECX                             Computing jump offset
EB 05           JMP SHORT bde0b6bc.0040307C
C8 DB1AD6       ENTER 1ADB,0D6
F9              STC
01D8            ADD EAX,EBX                         Adding imagebase
74 05           JE SHORT bde0b6bc.00403085
75 03           JNZ SHORT bde0b6bc.00403085
9F              LAHF
B1 FC           MOV CL,0FC
50              PUSH EAX
C3              RETN
```

```
0FB646 68       MOVZX EAX,BYTE PTR DS:[ESI+68]      NtGlobalFlag
EB 01           JMP SHORT bde0b6bc.00402F50
89              DB 89
40              INC EAX                             Increasing NtGlobalFlag value
74 08           JE SHORT bde0b6bc.00402F5B
75              DB 75                               CHAR 'u'
06              DB 06
A8              DB A8
11F0            ADC EAX,ESI
5E              POP ESI
B7 12           MOV BH,12
68 D12E0000     PUSH 2ED1                           stack push
75 04           JNZ SHORT bde0b6bc.00402F66
74 02           JE SHORT bde0b6bc.00402F66
DEC5            FADDP ST(5),ST
8B0C24          MOV ECX,DWORD PTR SS:[ESP]          stack pop
83C4 04         ADD ESP,4
EB 01           JMP SHORT bde0b6bc.00402F6F
70              DB 70                               CHAR 'p'
F7E1            MUL ECX                             Computing jump offset
EB 05           JMP SHORT bde0b6bc.00402F78
CC              INT3
DF              DB DF
05              DB 05
D6              SALC
F9              STC
01D8            ADD EAX,EBX                         Adding Imagebase
EB 02           JMP SHORT bde0b6bc.00402F7E
8B              DB 8B
45              INC EBP
FFE0            JMP EAX
```

On-Demand Decryption

The malware decrypts the code on demand just before executing it and once the code is executed, the malware encrypts it back. The malware does this, to prevent its complete code exposure in one shot:




```
68 A72A96F9       PUSH F9962AA7
5A                POP EDX
EB 05             JMP SHORT bde0b6bc.004012B9
B6 32             MOV DH,32
EB F4             JMP SHORT bde0b6bc.004012AC
EC                IN AL,DX
E8 00000000       CALL bde0b6bc.004012BE
75 04             JNZ SHORT bde0b6bc.004012C4
74 02             JE SHORT bde0b6bc.004012C4
D4 2E             AAM 2E
8B3424            MOV ESI,DWORD PTR SS:[ESP]
83C4 04           ADD ESP,4
EB 0A             JMP SHORT bde0b6bc.004012D6
40                INC EAX
81EE BE120000     SUB ESI,12BE
EB 05             JMP SHORT bde0b6bc.004012DA
20EB              AND BL,CH
F5                CMC
40                INC EAX
20EB              AND BL,CH
05 444638D6       ADD EAX,D6384644
F9                STC
01C6              ADD ESI,EAX
EB 09             JMP SHORT bde0b6bc.004012EE
A3 8138565F       MOV DWORD PTR DS:[5F563881],EAX
EB 05             JMP SHORT bde0b6bc.004012F1
E7 7D             OUT 7D,EAX
EB F8             JMP SHORT bde0b6bc.004012E8
DD                ???
AC                LODS BYTE PTR DS:[ESI]
EB 05             JMP SHORT bde0b6bc.004012F9
4C                DEC ESP
5E                POP ESI
38D6              CMP DH,DL
F9                STC
30D0              XOR AL,DL
AA                STOS BYTE PTR ES:[EDI]
E2 F3             LOOPD SHORT bde0b6bc.004012F1
```

Loaded module

The malware checks for below modules in the current process, if any of them is loaded malware terminates the execution.

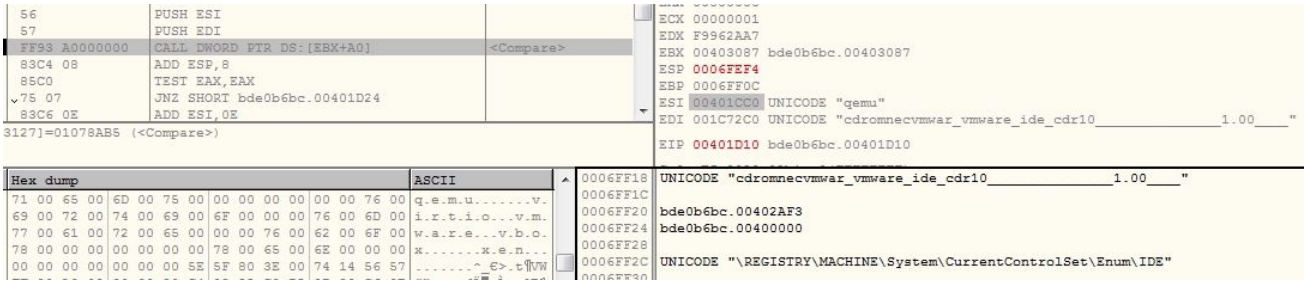
- sbiedll (Sandboxie module)
- aswhook (Avast module)
- snxhk (Avast module)

```
FF53 18           CALL DWORD PTR DS:[EBX+18]    kernel32.GetModuleHandleA
85C0              TEST EAX,EAX
0F85 1B020000     JNZ bde0b6bc.004025B4
```

Virtual Environment

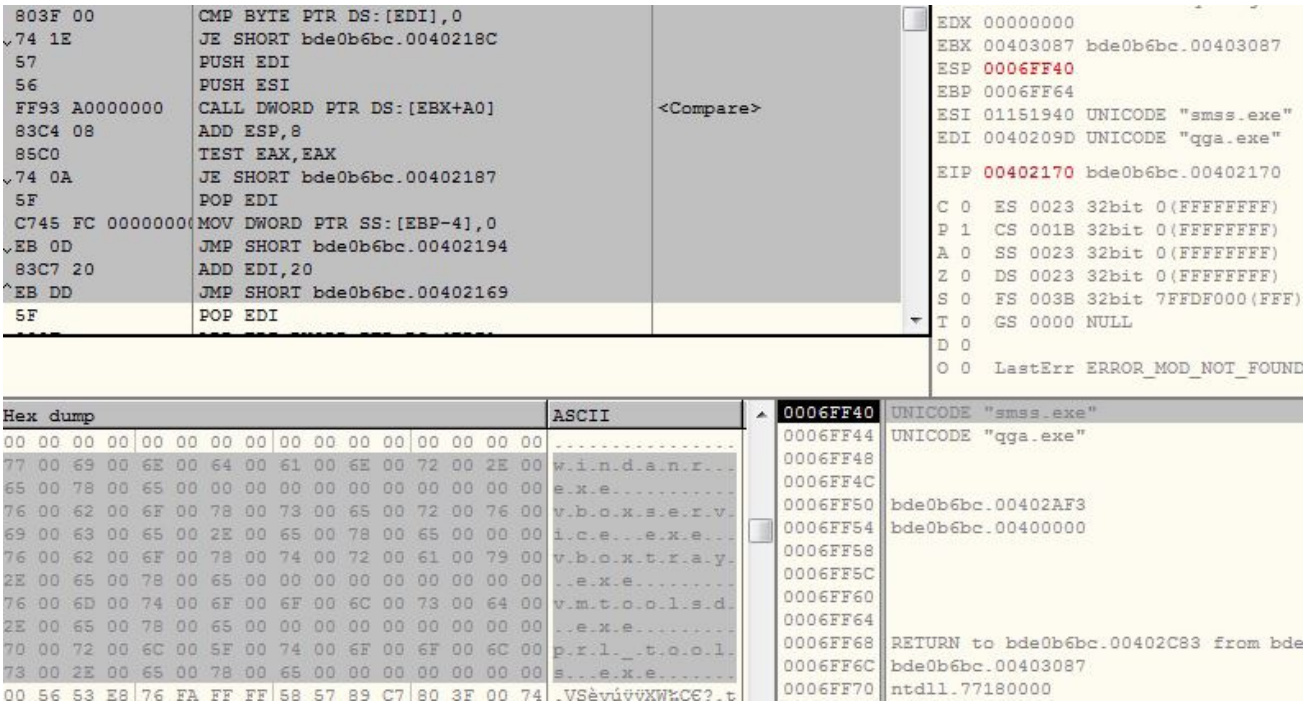
The malware examines registry values “\REGISTRY\MACHINE\System\CurrentControlSet\Enum\IDE” and “\REGISTRY\MACHINE\System\CurrentControlSet\Enum\SCSI” for below substrings to check for virtual environment.

- qemu
- virtio
- vmware
- vbox
- xen



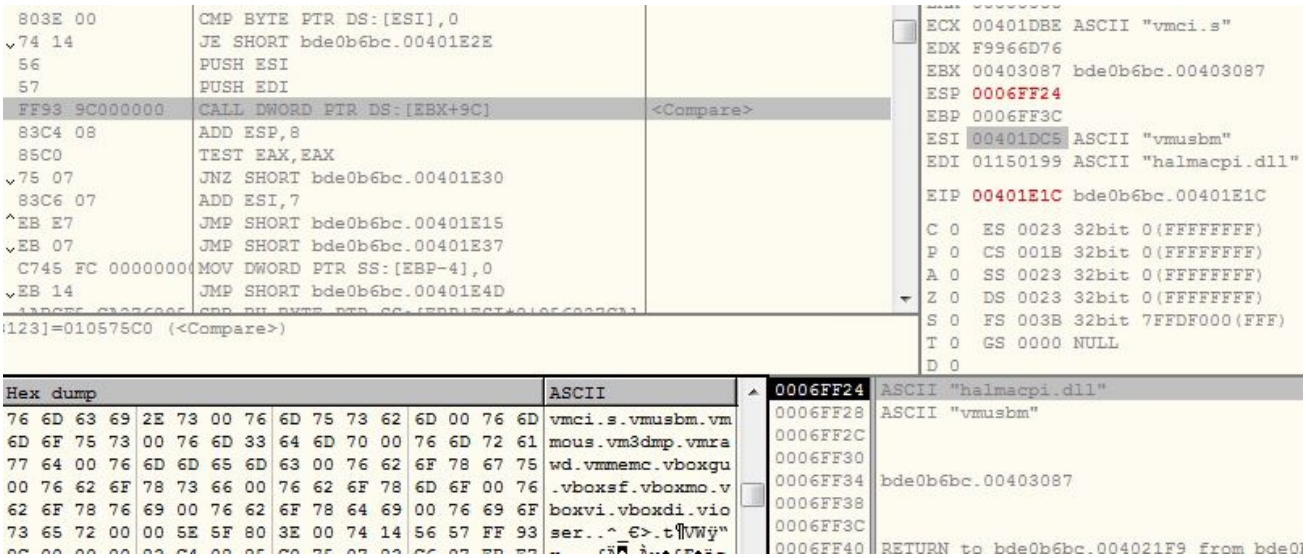
The malware enumerates through all the running processes and looks for below processes. If any of the process is found the malware terminates the execution. The malware shows laziness in the code here, instead of dynamic size for individual process name, the malware keeps the size to 0x20 bytes for all the process names:

- qemu-ga.exe
- qga.exe
- windanr.exe
- vboxservice.exe
- vboxtray.exe
- vmtoolsd.exe
- prl_tools.exe



The malware looks for below 7 bytes substrings of filenames into victim's machine. If any of them is found the malware terminates the execution:

- vmci.s
- vmusbm
- vmmous
- vm3dmp
- vmrawd
- vmmemc
- vboxgu
- vboxsf
- vboxmo
- vboxvi
- vboxdi
- vioser



Code Injection

The malware gets the explorer.exe process id using APIs *GetShellWindow* and *GetWindowThreadProcessId*:

```
FF53 48          CALL DWORD PTR DS:[EBX+48]      user32.GetShellWindow
85C0             TEST EAX,EAX
0F84 17030000    JE bde0b6bc.00401902
8945 A4          MOV DWORD PTR SS:[EBP-5C],EAX
8D75 A0          LEA ESI,DWORD PTR SS:[EBP-60]
893E             MOV DWORD PTR DS:[ESI],EDI
56               PUSH ESI
50               PUSH EAX
FF53 4C          CALL DWORD PTR DS:[EBX+4C]      user32.GetWindowThreadProcessId
8B06             MOV EAX,DWORD PTR DS:[ESI]
85C0             TEST EAX,EAX
0F84 FE020000    JE bde0b6bc.00401900
```

The malware creates and maps two sections in explorer.exe, one section has **PAGE_READWRITE** access attributes to store data and second section has **PAGE_EXECUTE_READ** access attributes to inject shellcode. Not enabling **WRITE** access to the shellcode memory makes the debugging little more difficult as this will prevent from putting software breakpoints and modifying code as per researcher's need:

```
FF53 74         CALL DWORD PTR DS:[EBX+74]          NtCreateSection
85C0            TEST EAX,EAX
0F85 06020000   JNZ bde0b6bc.00401900
837D F8 00      CMP DWORD PTR SS:[EBP-8],0
0F84 FC010000   JE bde0b6bc.00401900
FF75 B0         PUSH DWORD PTR SS:[EBP-50]
8F45 C8         POP DWORD PTR SS:[EBP-38]
8D45 B8         LEA EAX,DWORD PTR SS:[EBP-48]
8938            MOV DWORD PTR DS:[EAX],EDI
8D4D C8         LEA ECX,DWORD PTR SS:[EBP-38]
6A 04           PUSH 4
57              PUSH EDI
6A 01           PUSH 1
51              PUSH ECX
57              PUSH EDI
57              PUSH EDI
57              PUSH EDI
50              PUSH EAX
6A FF           PUSH -1
FF36            PUSH DWORD PTR DS:[ESI]
FF53 78         CALL DWORD PTR DS:[EBX+78]          NtMapViewOfSection
85C0            TEST EAX,EAX
0F85 D5010000   JNZ bde0b6bc.00401900
8D45 C0         LEA EAX,DWORD PTR SS:[EBP-40]
8938            MOV DWORD PTR DS:[EAX],EDI
8D4D C8         LEA ECX,DWORD PTR SS:[EBP-38]
6A 20           PUSH 20                             PAGE_EXECUTE_READ
57              PUSH EDI
6A 01           PUSH 1
51              PUSH ECX
57              PUSH EDI
57              PUSH EDI
57              PUSH EDI
50              PUSH EAX
FF75 F4         PUSH DWORD PTR SS:[EBP-C]
FF36            PUSH DWORD PTR DS:[ESI]
FF53 78         CALL DWORD PTR DS:[EBX+78]          NtMapViewOfSection
```

The malware injects shellcode into the mapped section and does NtCreateThreadEx passing data section address as parameter:

```
51              PUSH ECX
8B4A 10         MOV ECX,DWORD PTR DS:[EDX+10]
85C9            TEST ECX,ECX
74 0E           JE SHORT bde0b6bc.0040182C
8B7A 0C         MOV EDI,DWORD PTR DS:[EDX+C]
037D B8         ADD EDI,DWORD PTR SS:[EBP-48]
8B72 14         MOV ESI,DWORD PTR DS:[EDX+14]
0375 0C         ADD ESI,DWORD PTR SS:[EBP+C]
F3:A4           REP MOVS BYTE PTR ES:[EDI],BYTE PTR DS:[ESI]
83C2 28         ADD EDX,28
59              POP ECX
E2 E4           LOOPD SHORT bde0b6bc.00401816
```

ShellCode Execution:

The Injected shellcode into explorer.exe spawns two sub-threads which keep an eye on monitoring tools. If the researcher opens any of the monitoring tool or analysis tool that will be immediately terminated by the sub-threads while the main thread doing its job.

Thread 1

This thread enumerates through all running processes, computes hash of the running process name and compares it with its list of hashes to terminate below processes:

- 56DAB1A9 → Autoruns.exe
- F3E35F5E → procexp.exe
- 2407724B → procexp64.exe
- FBC25850 → procmon.exe
- 27151A96 → procmon64.exe
- E6ED4551 → Tcpview.exe
- 27D7E006 → Wireshark.exe
- 2CEB6C62 → ProcessHacker.exe
- EDCD7F5E → ollydbg.exe
- 70A30042 → x32dbg.exe
- 4EA30D45 → x64dbg.exe
- 0CCD4A10 → idaq.exe
- 0CCD4C3A → idaw.exe
- 0956AD95 → idaq64.exe
- 337CAD95 → idaw64.exe

```
FF96 370D0000   CALL DWORD PTR DS:[ESI+D37]          CreateToolhelp32Snapshot
8BF8            MOV EDI,EAX
83FF FF         CMP EDI,-1
74 5A           JE SHORT 01B7321A
8D4424 10       LEA EAX,DWORD PTR SS:[ESP+10]
C74424 10 28010( MOV DWORD PTR SS:[ESP+10],128
50              PUSH EAX
57              PUSH EDI
FF96 3B0D0000   CALL DWORD PTR DS:[ESI+D3B]          Process32First
EB 39           JMP SHORT 01B7320F
8D4C24 34       LEA ECX,DWORD PTR SS:[ESP+34]
E8 4A0E0000     CALL <ComputeHash>
35 C967DF52     XOR EAX,52DF67C9
8BCB            MOV ECX,EBX
3981 C010B701   CMP DWORD PTR DS:[ECX+1B710C0],EAX   Compare hash value
74 0A           JE SHORT 01B731F8
83C1 04         ADD ECX,4
83F9 3C         CMP ECX,3C
72 F0           JB SHORT 01B731E6
EB 0B           JMP SHORT 01B73203
8B5424 18       MOV EDX,DWORD PTR SS:[ESP+18]
8BCE            MOV ECX,ESI
E8 7E070000     CALL 01B73981                        TerminateProcess
8D4424 10       LEA EAX,DWORD PTR SS:[ESP+10]
50              PUSH EAX
57              PUSH EDI
FF96 3F0D0000   CALL DWORD PTR DS:[ESI+D3F]          kernel32.Process32Next
85C0            TEST EAX,EAX
75 C3           JNZ SHORT 01B731D6
```

Thread 2

The malware enumerates through windows, computes hash value of windows name and compares it to terminate processes attached with below windows list:

- 61C75CDC → Autoruns
- 4DFA76EB → PROCEXPL
- 95E8B472 → PROCMON_WINDOW_CLASS
- 62DC4674 → TCPViewClass

- 6A0FAA84 → Wireshark
- 7FF991A1 → ProcessHacker
- BEDA6295 → OLLYDBG
- 62DD69FD → IDA

```
FF96 070E0000   CALL DWORD PTR DS:[ESI+E07]          EnumWindows
6A 64           PUSH 64
FF96 D30C0000   CALL DWORD PTR DS:[ESI+CD3]
83BE 430C0000 0(CMP DWORD PTR DS:[ESI+C43],0
^75 E3          JNZ SHORT 01B73241
6A 00           PUSH 0
FF96 B30C0000   CALL DWORD PTR DS:[ESI+CB3]
5E              POP ESI
C2 0400         RETN 4
55              PUSH EBP
8BEC            MOV EBP,ESP
81EC 04010000   SUB ESP,104
8D85 FCFEFFFF   LEA EAX,DWORD PTR SS:[EBP-104]
56              PUSH ESI
8B75 0C         MOV ESI,DWORD PTR SS:[EBP+C]
68 04010000     PUSH 104
50              PUSH EAX
FF75 08         PUSH DWORD PTR SS:[EBP+8]
FF96 FF0D0000   CALL DWORD PTR DS:[ESI+DFF]
85C0            TEST EAX,EAX
⌄74 3F          JE SHORT 01B732CF
8D8D FCFEFFFF   LEA ECX,DWORD PTR SS:[EBP-104]
E8 8E0D0000     CALL <ComputeHash>
35 C967DF52     XOR EAX,52DF67C9
33C9            XOR ECX,ECX
3981 5010B701   CMP DWORD PTR DS:[ECX+1B71050],EAX   Compare Hash Value
⌄74 0A          JE SHORT 01B732B4
83C1 04         ADD ECX,4
83F9 20         CMP ECX,20
^72 F0          JB SHORT 01B732A2
⌄EB 1B          JMP SHORT 01B732CF
8365 0C 00      AND DWORD PTR SS:[EBP+C],0
8D45 0C         LEA EAX,DWORD PTR SS:[EBP+C]
50              PUSH EAX
FF75 08         PUSH DWORD PTR SS:[EBP+8]
FF96 030E0000   CALL DWORD PTR DS:[ESI+E03]          USER32.GetWindowThreadProcessId
8B55 0C         MOV EDX,DWORD PTR SS:[EBP+C]
8BCE            MOV ECX,ESI
E8 B2060000     CALL 01B73981                        TerminateProcess
```

Main Thread

The main thread starts with **Process Environment Block (PEB)** traversal, to get *ImageBase* of *ntdll.dll* and *kernel32.dll*. The malware then enumerates the export functions to get the the addresses of required APIs. Instead of direct API names the malware keeps the hash values list, which is being compared to the hash value of the exported function name:

```
64:A1 30000000   MOV EAX,DWORD PTR FS:[30]          Process Environment Block
53               PUSH EBX
55               PUSH EBP
56               PUSH ESI
8B68 0C          MOV EBP,DWORD PTR DS:[EAX+C]       _PEB_LDR_DATA
8BF1             MOV ESI,ECX
83C5 0C          ADD EBP,0C                         InLoadOrderModuleList
57               PUSH EDI
896C24 10        MOV DWORD PTR SS:[ESP+10],EBP
33FF             XOR EDI,EDI
8B55 00          MOV EDX,DWORD PTR SS:[EBP]
8B5A 30          MOV EBX,DWORD PTR DS:[EDX+30]      module base name
33C0             XOR EAX,EAX
66:3903          CMP WORD PTR DS:[EBX],AX
74 1C            JE SHORT 01B7173D
33ED             XOR EBP,EBP
8A03             MOV AL,BYTE PTR DS:[EBX]
8D5B 02          LEA EBX,DWORD PTR DS:[EBX+2]
24 DF            AND AL,0DF
0FB6C0           MOVZX EAX,AL
33F8             XOR EDI,EAX
C1C7 08          ROL EDI,8
03F8             ADD EDI,EAX
66:392B          CMP WORD PTR DS:[EBX],BP
75 EA            JNZ SHORT 01B71723
8B6C24 10        MOV EBP,DWORD PTR SS:[ESP+10]
81F7 C967DF52    XOR EDI,52DF67C9
81FF 7D09DE08    CMP EDI,8DE097D                    ntdll.dll hash value compare
74 13            JE SHORT 01B7175E
81FF E5A1273B    CMP EDI,3B27A1E5                   kernel32.dll hash value compare
75 14            JNZ SHORT 01B71767
8B42 18          MOV EAX,DWORD PTR DS:[EDX+18]
8986 730E0000    MOV DWORD PTR DS:[ESI+E73],EAX     storing kernel32.dll imagebase
EB 09            JMP SHORT 01B71767
8B42 18          MOV EAX,DWORD PTR DS:[EDX+18]
8986 6F0E0000    MOV DWORD PTR DS:[ESI+E6F],EAX     storing ntdll.dll imagebase
```

Computing module name hash value

The malware keeps list of RC4 encrypted strings in a structure, in which first bytes tells the string size followed by encrypted string. The malware perform RC4 decryptions just before using them:

```
6A 04            PUSH 4                             Size of key
57               PUSH EDI                           Size of encrypted string
8D5424 18        LEA EDX,DWORD PTR SS:[ESP+18]      Key (06d9708c)
8BCE             MOV ECX,ESI                        Encrypted String
E8 08000000      CALL <RC4>
5F               POP EDI
```

```
Hex dump                                                 ASCII
2D A0 E3 AE DE DD C5 69 C8 14 6B 01 D3 83 03 3A   - ã®ÞÝÅiÈ¶k Óƒ¹:
68 D7 E4 F9 98 2F 52 FE 96 32 EE FA B2 36 93 61   h×äù7Rþ–2îú²6"a
D1 33 1F 3E 46 83 23 D6 23 32 FA BD 6B F6 24 9B   Ñ3>Fƒ#Ö#2ú½kö$›
F8 BC DA D9 9E 34 82 2C 48 1B 9E 96 03 26 60 DD   ø¼ÚÙž4‚,H←ž–└&`Ý
F5 8A A3 24 55 F4 88 2A EE B1 FC 12 86 74 80 31   õŠ£$Uô^*î±ü┘†t€1
04 38 46 0C A9 F3 AC CF DE 96 75 D5 5E 61 1E 91   ┘8F.©ó¬ÏÞ–uÕ^a'
09 84 F8 B9 CF DA 96 29 89 4A 0B B8 FB AF C9 C7   .„ø¹ÏÚ–)‰J♂¸û¯ÉÇ
91 19 94 19 7F 17 0C AD EF AA C2 C1 8D 23 95 5E   '├"┤ ┤.-ï²ÂÁ#•^
60 0A 98 06 BD E4 BF DC 9D CD 08 A9 F3 AC CF DE   `.˜½ä¿ÜÍ◘©ó¬ÏÞ
96 75 D5 06 BD E5 B6 C3 C1 91 05 A7 FB BF 9D 9C   –uÕ½å¶ÃÁ'|§û¿œ
07 BF FE B4 C6 DA 8B 36 06 BF E4 E8 F1 9D CD 06   •¿þ´ÆÚ‹6–¿äèñÍ–
AC F9 A9 CF DE 96 07 BB FF BF C2 C2 CC 74 07 BB   ¬ù©ÏÞ–•»ÿ¿ÂÂÌt•»
```

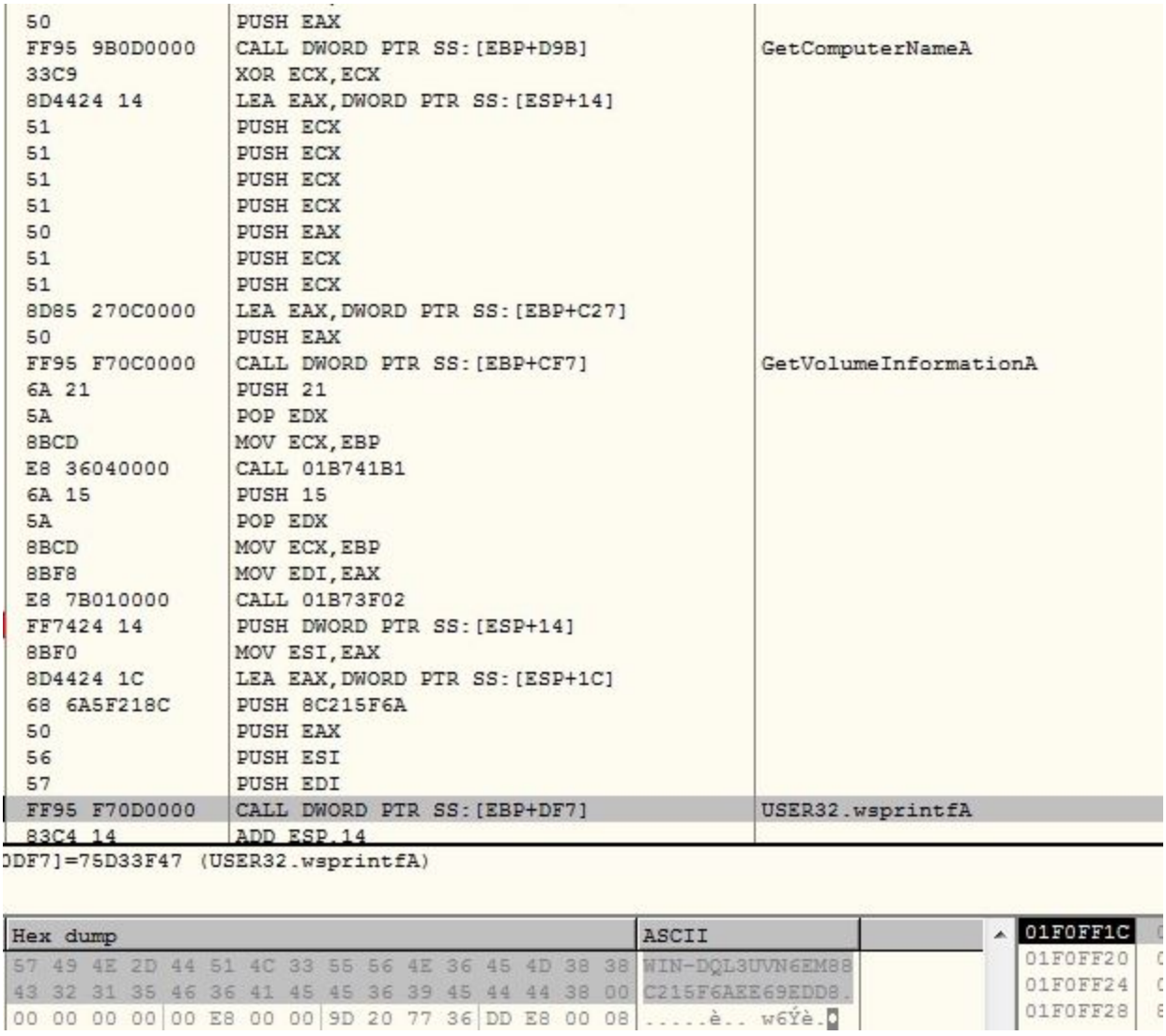
Encrypted strings with their sizes

```
01F0FF10  000
01F0FF14  000
01F0FF18  02C
01F0FF1C  01A
01F0FF20  01A
01F0FF24  01A
01F0FF28  8C7
01F0FF2C  01E
01F0FF30  01A
01F0FF34  01A
01F0FF38  01E
01F0FF3C  01A
01F0FF40  000
01F0FF44  EE6
```

The malware computes a unique identifier for the victim's machine using below formula:

MD5(computer name + hardcoded DWORD value + system drive serial number) + system drive serial number

The malware creates mutex with the unique identifier to restrict execution of another instance of the shellcode and if another instance is already running malware terminates its execution:



The malware reads *Internet Explorer* version information from registry and gets user agent string for it:

```
FF96 630E0000   CALL DWORD PTR DS:[ESI+E63]   ObtainUserAgentString
5E              POP ESI                      020707D0
5B              POP EBX
```

```
Hex dump                                        ASCII
4D 6F 7A 69 6C 6C 61 2F 34 2E 30 20 28 63 6F 6D Mozilla/4.0 (com
70 61 74 69 62 6C 65 3B 20 4D 53 49 45 20 38 2E patible; MSIE 8.
30 3B 20 57 69 6E 64 6F 77 73 20 4E 54 20 36 2E 0; Windows NT 6.
31 3B 20 54 72 69 64 65 6E 74 2F 34 2E 30 3B 20 1; Trident/4.0;
53 4C 43 43 32 3B 20 2E 4E 45 54 20 43 4C 52 20 SLCC2; .NET CLR
32 2E 30 2E 35 30 37 32 37 3B 20 2E 4E 45 54 20 2.0.50727; .NET
43 4C 52 20 33 2E 35 2E 33 30 37 32 39 3B 20 2E CLR 3.5.30729; .
4E 45 54 20 43 4C 52 20 33 2E 30 2E 33 30 37 32 NET CLR 3.0.3072
39 3B 20 4D 65 64 69 61 20 43 65 6E 74 65 72 20 9; Media Center
50 43 20 36 2E 30 3B 20 2E 4E 45 54 34 2E 30 43 PC 6.0; .NET4.0C
3B 20 2E 4E 45 54 34 2E 30 45 3B 20 49 6E 66 6F ; .NET4.0E; Info
50 61 74 68 2E 33 29 00 00 00 00 00 00 00 00 00 Path.3).........
```

The malware drops self copy into %APPDATA% directory and the file name is computed by encoding initial 7 bytes from the unique identifier:

```
FF95 070D0000   CALL DWORD PTR SS:[EBP+D07]   kernel32.CopyFileW
85C0            TEST EAX,EAX
0F84 C1000000   JE 01B71C43
56              PUSH ESI
FF95 1B0D0000   CALL DWORD PTR SS:[EBP+D1B]
6A 17           PUSH 17
5A              POP EDX
8BCD            MOV ECX,EBP
E8 6F230000     CALL <DecryptRC4>
0D07]=772F67C3 (kernel32.CopyFileW)

01F0FF38  01A3024B  UNICODE "E:\samples\bde0b6bcc975d490386b320ab8559083a770ab73e751ab688b8dc0e89e2dcea1.exe.bin"
01F0FF3C  01A307B3  UNICODE "C:\Users\Deepak\AppData\Roaming\udvvgjv"
```

The malware deletes the current instance of the malware and it deletes zone identifier from the self copy dropped in %APPDATA%:

```
FF95 1B0D0000   CALL DWORD PTR SS:[EBP+D1B]   kernel32.DeleteFileW
D1B]=77300F62 (kernel32.DeleteFileW)

0  02070818  UNICODE "C:\Users\Deepak\AppData\Roaming\udvvgjv:Zone.Identifier"
4  00000000
```

The malware sets dropped file property as FILE_ATTRIBUTE_HIDDEN and FILE_ATTRIBUTE_SYSTEM. The malware steals creation time from advapi32.dll and mark the same creation time for the dropped file to avoid being red flagged from any of the security providers.

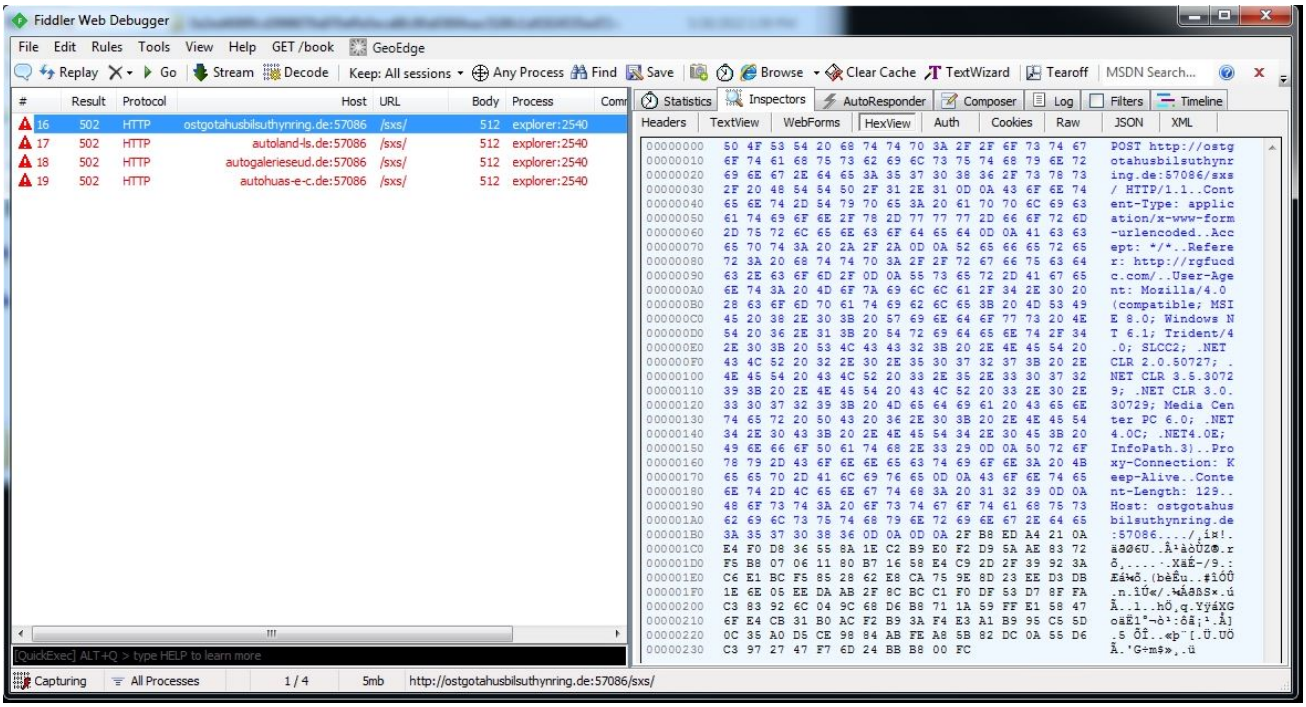
C&C Communication

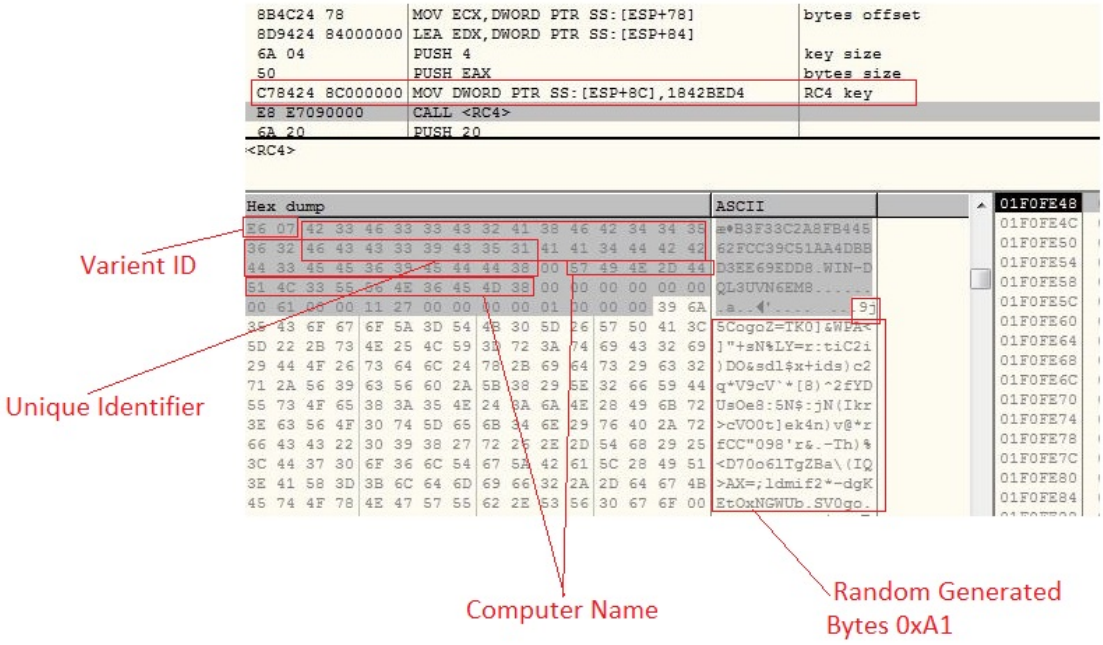
The malware contains 4 C&C servers:

- ostgotahusbilsuthynring.de
- autoland-ls.de
- autogalerieseud.de

- autohuas-e-c.de

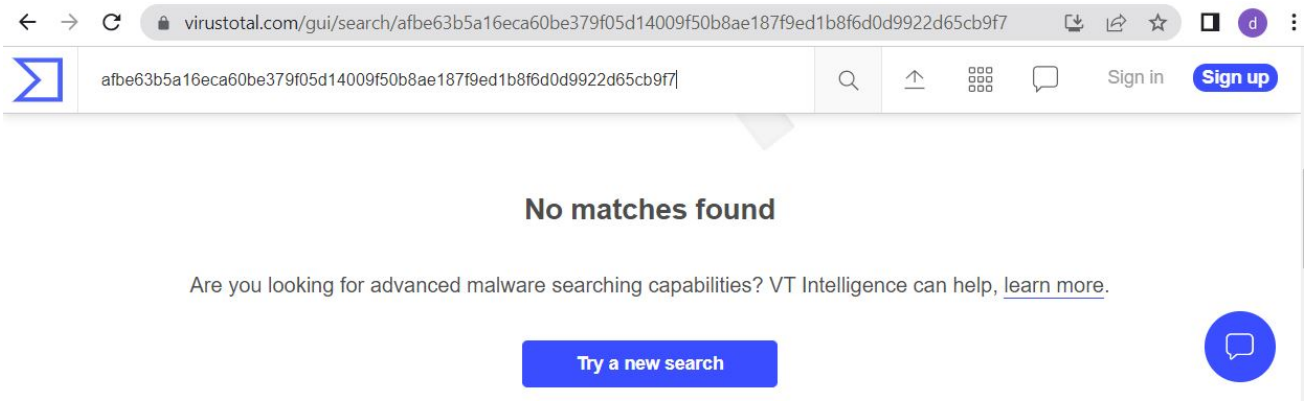
The malware calculate CRC32 checksum for one of the C&C server before communicating, to make sure that the C&C has not been modified by the researcher and if the C&C is modified malware terminates the execution. The malware prepares post data which includes the variant id, unique identifier for the victim's machine, computer name and random 0xA1 bytes. The data is then encrypted by RC4 algorithm and sent to its C&C server:





At the time of analysis all 4 C&C server were not responding but digging deep into the malware code reveals that malware is expecting response from C&C server which should contain Variant ID (0x7E6), Plugin size and plugin modules.

Unavailability of the archive file in any of the popular threat intelligence sharing portals like the VirusTotal and the ReversingLabs indicates its uniqueness and limited distribution:



Evidence of detection by RTDMI ™ engine can be seen below in the Capture ATP report for this file: