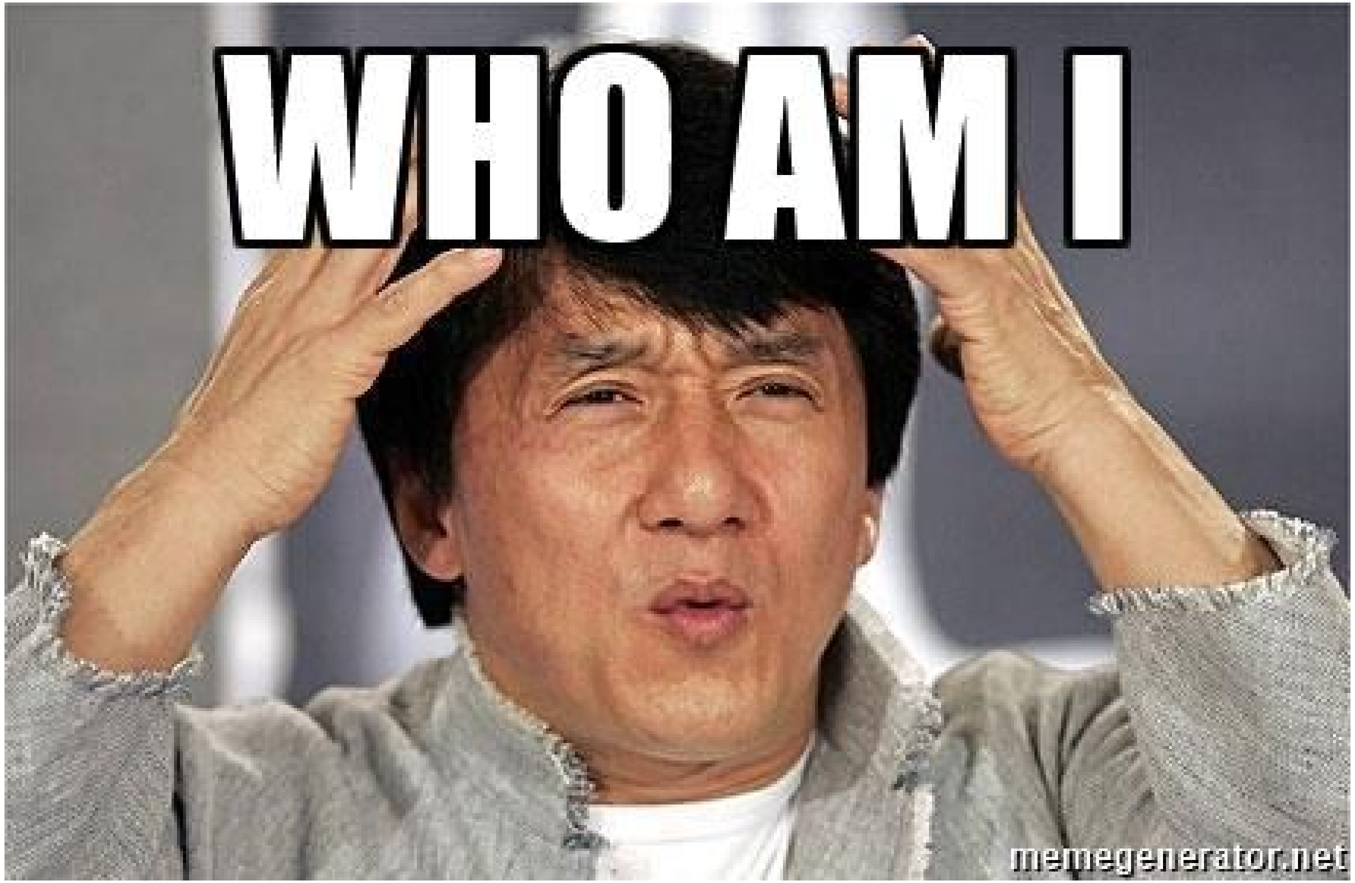
LabyREnth 2017 Binary #3 WriteUp

Fako


WHO AM I
memegenerator.net

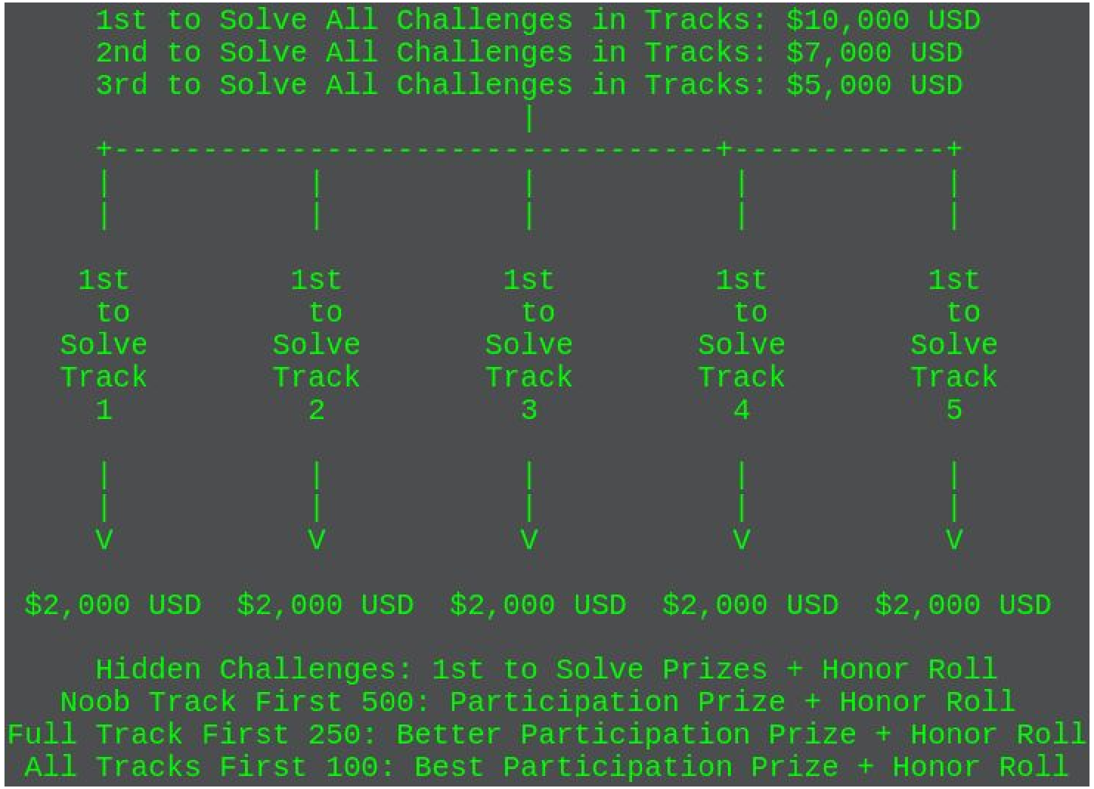
Me.

- Senior Principal System Engineer at GDP Labs
- DevSecOps by day, Reverse Engineer by night

What is LabyREnth?

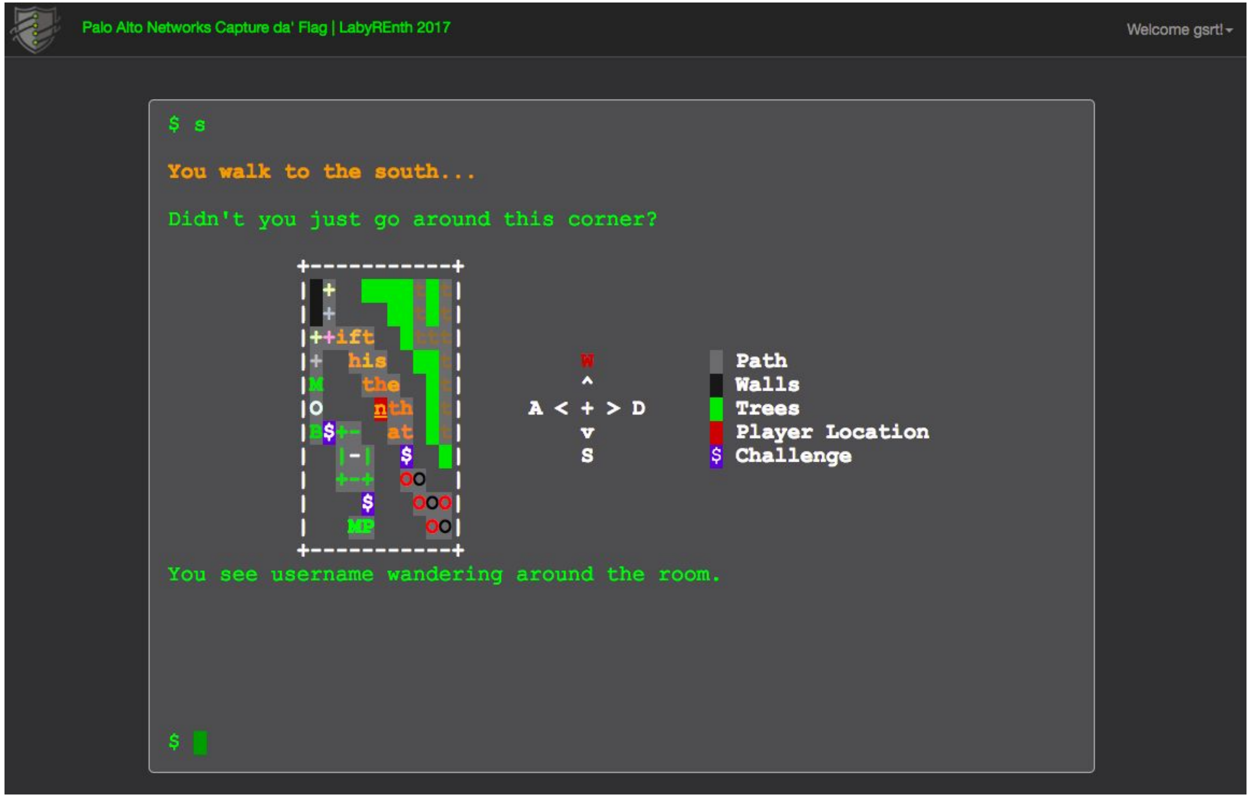
- Palo Alto Networks Annual CTF Event
- Mostly Reverse Engineering
- 5 Tracks, 5 Challenges per track
  - Binary
  - Programming
  - Docs
  - Mobile
  - Threat
- Additional Tracks
  - 6 Random Tracks
  - Final Boss??

Prizes!



```
     1st to Solve All Challenges in Tracks: $10,000 USD
     2nd to Solve All Challenges in Tracks: $7,000 USD
     3rd to Solve All Challenges in Tracks: $5,000 USD
                             |
     +----------------------------------+------------+
     |           |           |           |           |
     |           |           |           |           |

    1st         1st         1st         1st         1st
     to          to          to          to          to
   Solve       Solve       Solve       Solve       Solve
   Track       Track       Track       Track       Track
     1           2           3           4           5

     |           |           |           |           |
     |           |           |           |           |
     V           V           V           V           V

 $2,000 USD  $2,000 USD  $2,000 USD  $2,000 USD  $2,000 USD

     Hidden Challenges: 1st to Solve Prizes + Honor Roll
   Noob Track First 500: Participation Prize + Honor Roll
Full Track First 250: Better Participation Prize + Honor Roll
 All Tracks First 100: Best Participation Prize + Honor Roll
```

Interface



Palo Alto Networks Capture da' Flag | LabyREnth 2017

Welcome gsrt!

```
$ s

You walk to the south...

Didn't you just go around this corner?

+-----------+
| +       t t|
| +       t t|
|++ift    ttt|
|+  his     t|
|M   the   t|
|O    nth  t|
|B$+-  at  t|
|  |-|  $    |
|  +-+  OO   |
|    $   OOO|
|   MP    OO|
+-----------+        W                   Path
                     ^                   Walls
               A  <  +  >  D             Trees
                     v                   Player Location
                     S                 $ Challenge

You see username wandering around the room.

$
```

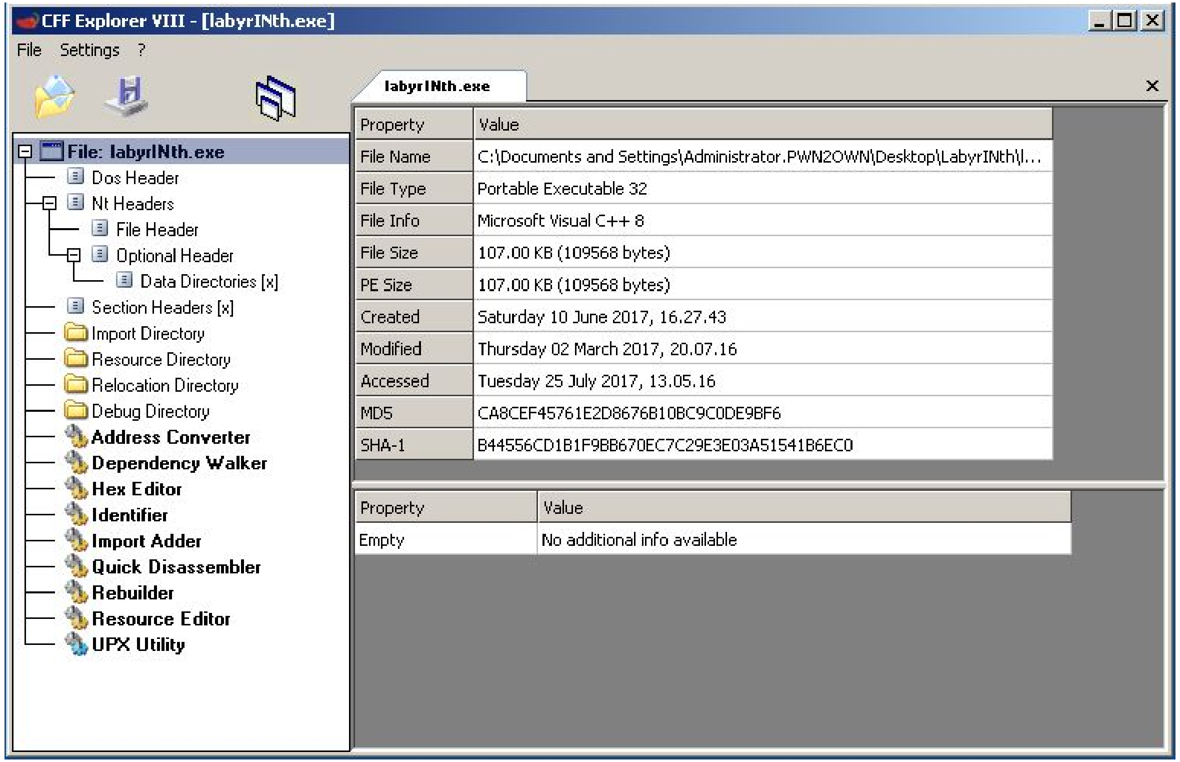
▨	Path
⬛	Walls
🟩	Trees
🟥	Player Location
$	Challenge

Binary Track #3

You walk to the east...
The goblin guarding the door giggles as he describes the next challenge.
7z Download
7z Password: labyrenth
Hint: You are going to need a virtual machine for this one.
Author(s): @xedi25

http://dl.labyrenth.com/labyrinth/d88b07e6d10481cb716e0c8a78519d2c8bfef2778e0332aef6c4f0699d74be6e.7z

The Binary



Property	Value
File Name	C:\Documents and Settings\Administrator.PWN2OWN\Desktop\LabyrINth\l...
File Type	Portable Executable 32
File Info	Microsoft Visual C++ 8
File Size	107.00 KB (109568 bytes)
PE Size	107.00 KB (109568 bytes)
Created	Saturday 10 June 2017, 16.27.43
Modified	Thursday 02 March 2017, 20.07.16
Accessed	Tuesday 25 July 2017, 13.05.16
MD5	CA8CEF45761E2D8676B10BC9C0DE9BF6
SHA-1	B44556CD1B1F9BB670EC7C29E3E03A51541B6EC0

Property	Value
Empty	No additional info available

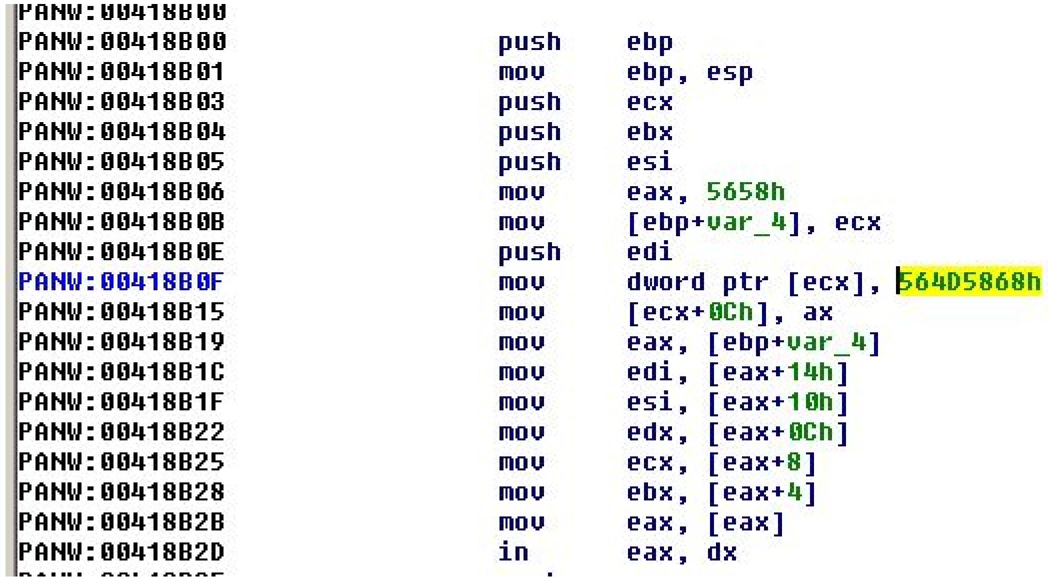
Strings

```
flag: %s
doesn't look like valid flag to me: %s
VMware version: %u
I don't think you can finish this today.
I don't think you can finish this today. Not with this attitude.
Slow.
Talk to you later.
```

Clue

5658 = “VX”

564D5868 = “VMXh”



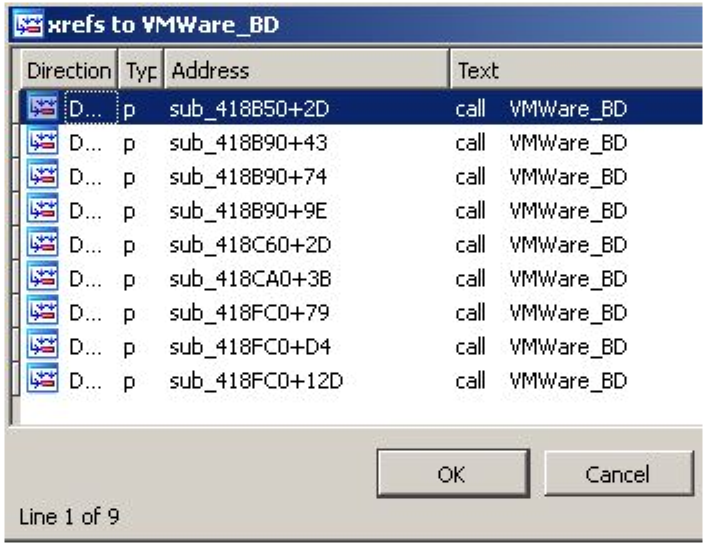
```
PANW:00418B00
PANW:00418B00                 push    ebp
PANW:00418B01                 mov     ebp, esp
PANW:00418B03                 push    ecx
PANW:00418B04                 push    ebx
PANW:00418B05                 push    esi
PANW:00418B06                 mov     eax, 5658h
PANW:00418B0B                 mov     [ebp+var_4], ecx
PANW:00418B0E                 push    edi
PANW:00418B0F                 mov     dword ptr [ecx], 564D5868h
PANW:00418B15                 mov     [ecx+0Ch], ax
PANW:00418B19                 mov     eax, [ebp+var_4]
PANW:00418B1C                 mov     edi, [eax+14h]
PANW:00418B1F                 mov     esi, [eax+10h]
PANW:00418B22                 mov     edx, [eax+0Ch]
PANW:00418B25                 mov     ecx, [eax+8]
PANW:00418B28                 mov     ebx, [eax+4]
PANW:00418B2B                 mov     eax, [eax]
PANW:00418B2D                 in      eax, dx
```

The VMWare Backdoor

https://sites.google.com/site/chitchatvmback/backdoor

```
/* in Intel syntax (MASM and most Windows based assemblers) */

  MOV EAX, 564D5868h                 /* magic number     */
  MOV EBX, command-specific-parameter
  MOV CX,  backdoor-command-number
  MOV DX,  5658h                     /* VMware I/O Port  */

  IN  EAX, DX (or OUT DX, EAX)
```

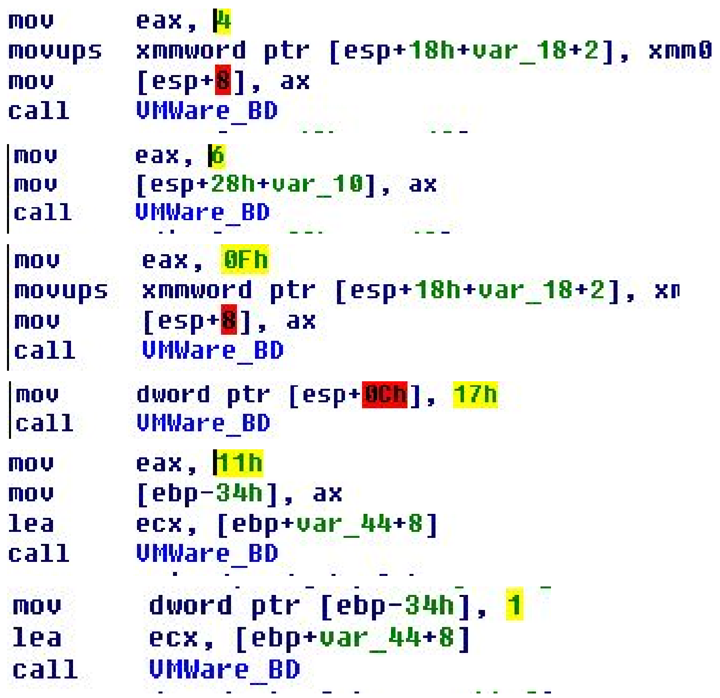
Call References


xrefs to VMWare_BD

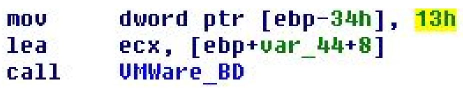
Direction	Typ	Address	Text
D...	p	sub_418B50+2D	call VMWare_BD
D...	p	sub_418B90+43	call VMWare_BD
D...	p	sub_418B90+74	call VMWare_BD
D...	p	sub_418B90+9E	call VMWare_BD
D...	p	sub_418C60+2D	call VMWare_BD
D...	p	sub_418CA0+3B	call VMWare_BD
D...	p	sub_418FC0+79	call VMWare_BD
D...	p	sub_418FC0+D4	call VMWare_BD
D...	p	sub_418FC0+12D	call VMWare_BD

Line 1 of 9

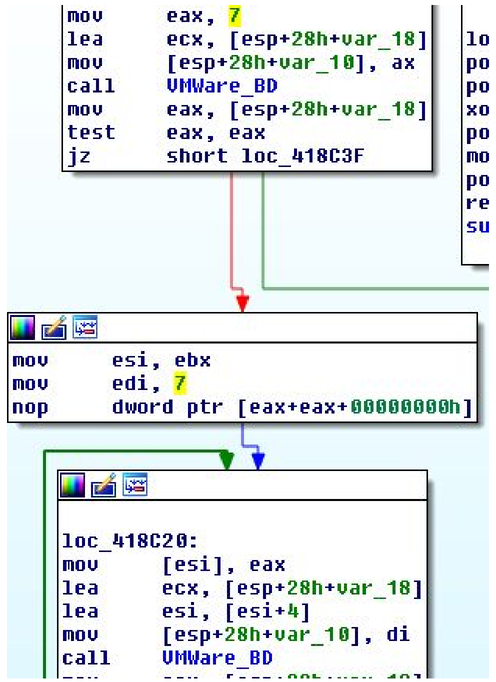
Functions Used



```
mov     eax, 4
movups  xmmword ptr [esp+18h+var_18+2], xmm0
mov     [esp+8], ax
call    VMWare_BD

mov     eax, 6
mov     [esp+28h+var_10], ax
call    VMWare_BD

mov     eax, 0Fh
movups  xmmword ptr [esp+18h+var_18+2], xm
mov     [esp+8], ax
call    VMWare_BD

mov     dword ptr [esp+0Ch], 17h
call    VMWare_BD

mov     eax, 11h
mov     [ebp-34h], ax
lea     ecx, [ebp+var_44+8]
call    VMWare_BD

mov     dword ptr [ebp-34h], 1
lea     ecx, [ebp+var_44+8]
call    VMWare_BD
```



```
mov     dword ptr [ebp-34h], 13h
lea     ecx, [ebp+var_44+8]
call    VMWare_BD
```



```
mov     eax, 7
lea     ecx, [esp+28h+var_18]
mov     [esp+28h+var_10], ax
call    VMWare_BD
mov     eax, [esp+28h+var_18]
test    eax, eax
jz      short loc_418C3F
```

```
lo
po
po
xo
po
mo
po
re
su
```

```
mov     esi, ebx
mov     edi, 7
nop     dword ptr [eax+eax+00000000h]
```

```
loc_418C20:
mov     [esi], eax
lea     ecx, [esp+28h+var_18]
lea     esi, [esi+4]
mov     [esp+28h+var_10], di
call    VMWare_BD
```

Functions Used

- 01h = Get processor speed (MHz)
- 04h = Get mouse cursor position
- 06h = Get text length from clipboard
- 07h = Get text from clipboard
- 0Fh = Get host screen size
- 11h = Get virtual hardware version
- 13h = Get BIOS UUID
- 17h = Get host's system time (GMT)

Problem?

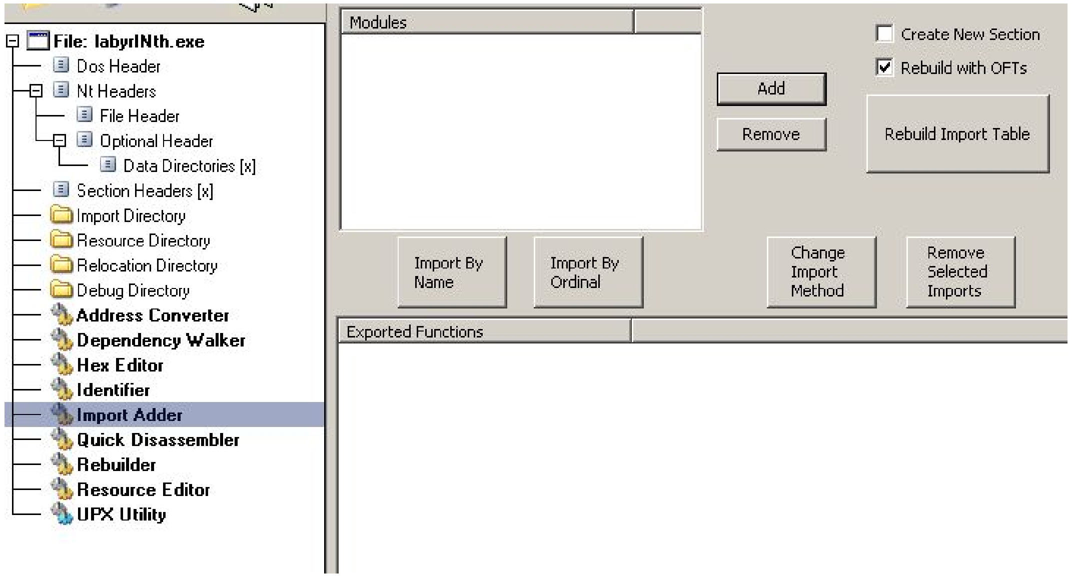
- No VMWare on my laptop
- Limited access to internet

Solution

- Dll Injection
- Redirect call

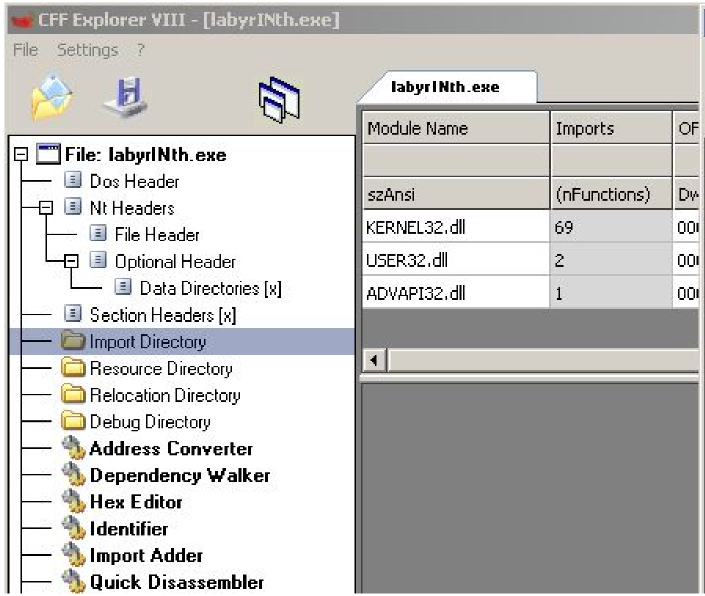
DLL Injection

- CFF Explorer to the rescue! (http://www.ntcore.com/exsuite.php)
- Import adder

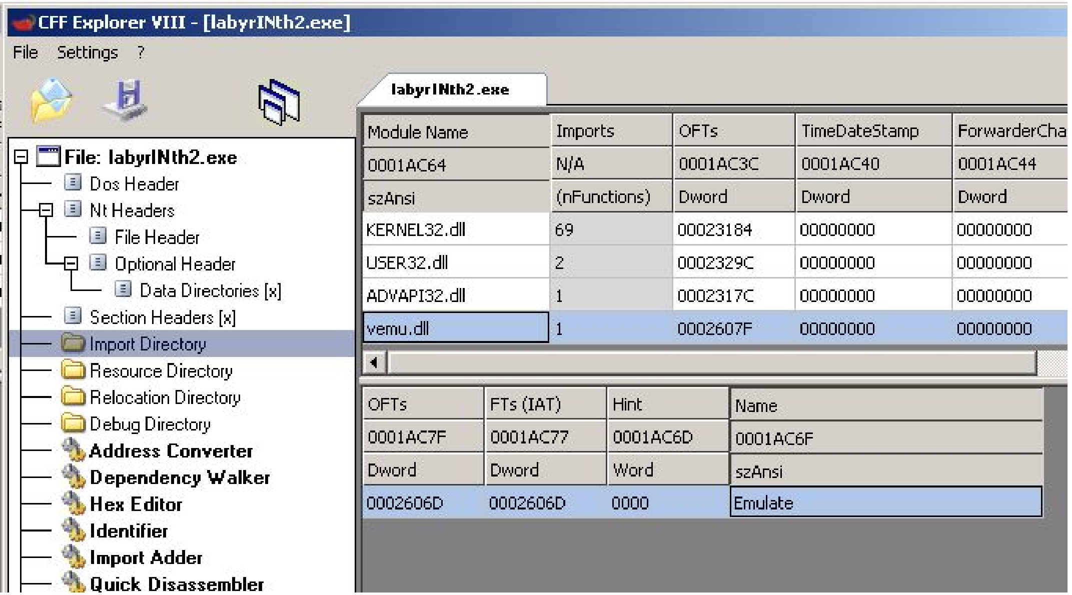


Create New Section: [ ] · Rebuild with OFTs: [✓]

DLL Injection



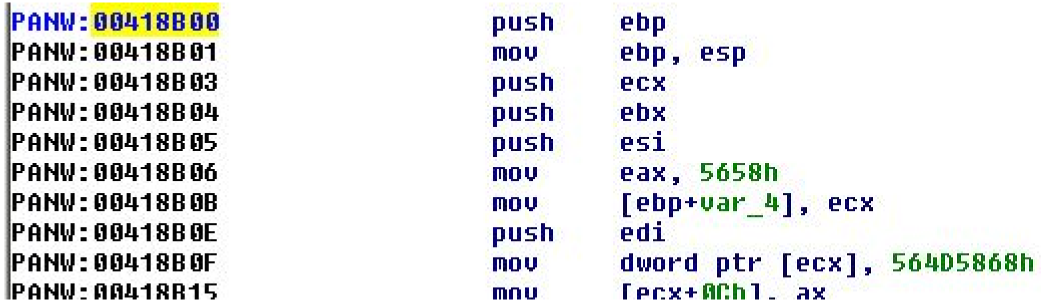
Module Name	Imports	OF
szAnsi	(nFunctions)	Dw
KERNEL32.dll	69	00
USER32.dll	2	00
ADVAPI32.dll	1	00



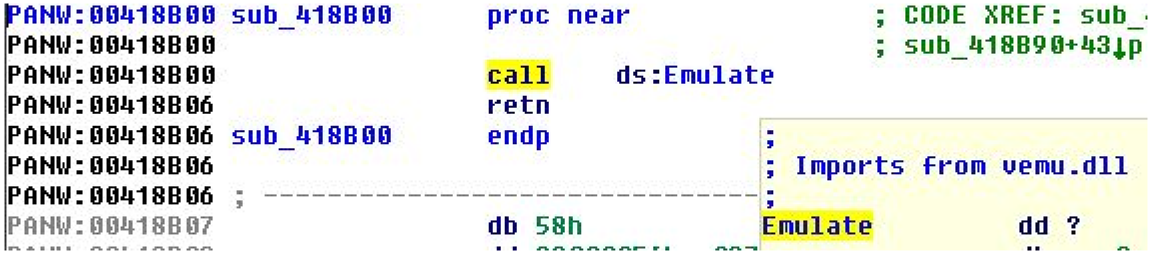
Module Name	Imports	OFTs	TimeDateStamp	ForwarderCha
0001AC64	N/A	0001AC3C	0001AC40	0001AC44
szAnsi	(nFunctions)	Dword	Dword	Dword
KERNEL32.dll	69	00023184	00000000	00000000
USER32.dll	2	0002329C	00000000	00000000
ADVAPI32.dll	1	0002317C	00000000	00000000
vemu.dll	1	0002607F	00000000	00000000

OFTs	FTs (IAT)	Hint	Name
0001AC7F	0001AC77	0001AC6D	0001AC6F
Dword	Dword	Word	szAnsi
0002606D	0002606D	0000	Emulate

Redirect Call



```
PANW:00418B00                 push    ebp
PANW:00418B01                 mov     ebp, esp
PANW:00418B03                 push    ecx
PANW:00418B04                 push    ebx
PANW:00418B05                 push    esi
PANW:00418B06                 mov     eax, 5658h
PANW:00418B0B                 mov     [ebp+var_4], ecx
PANW:00418B0E                 push    edi
PANW:00418B0F                 mov     dword ptr [ecx], 564D5868h
PANW:00418B15                 mov     [ecx+0Ch], ax
```



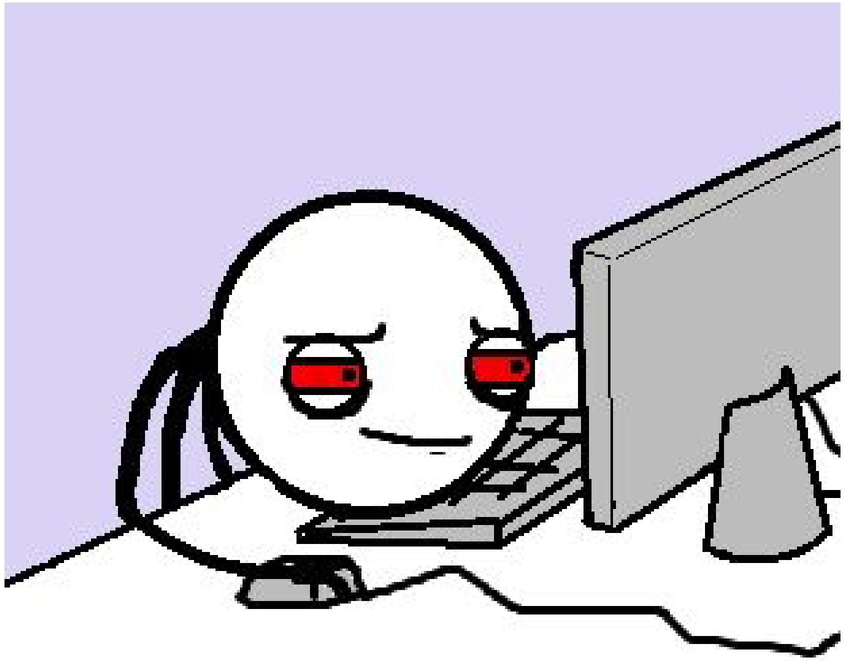
```
PANW:00418B00 sub_418B00      proc near               ; CODE XREF: sub_
PANW:00418B00                                         ; sub_418B90+43↓p
PANW:00418B00                 call    ds:Emulate
PANW:00418B06                 retn
PANW:00418B06 sub_418B00      endp
PANW:00418B06
PANW:00418B06 ; ------------------------------
PANW:00418B07                 db 58h
```

```
;
; Imports from vemu.dll
;
Emulate         dd ?
```

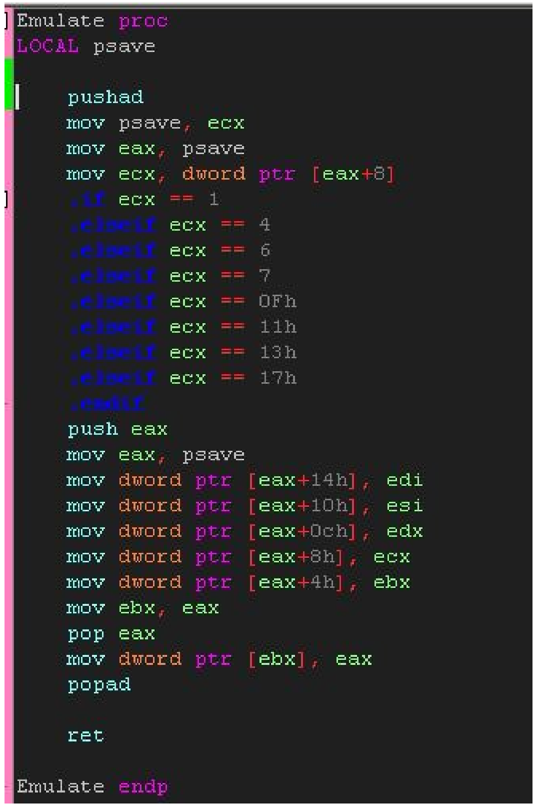
Debugging and Coding the DLL

- IDA Pro
- HexWorkshop
- Masm32 + RadASM

Debug Time!

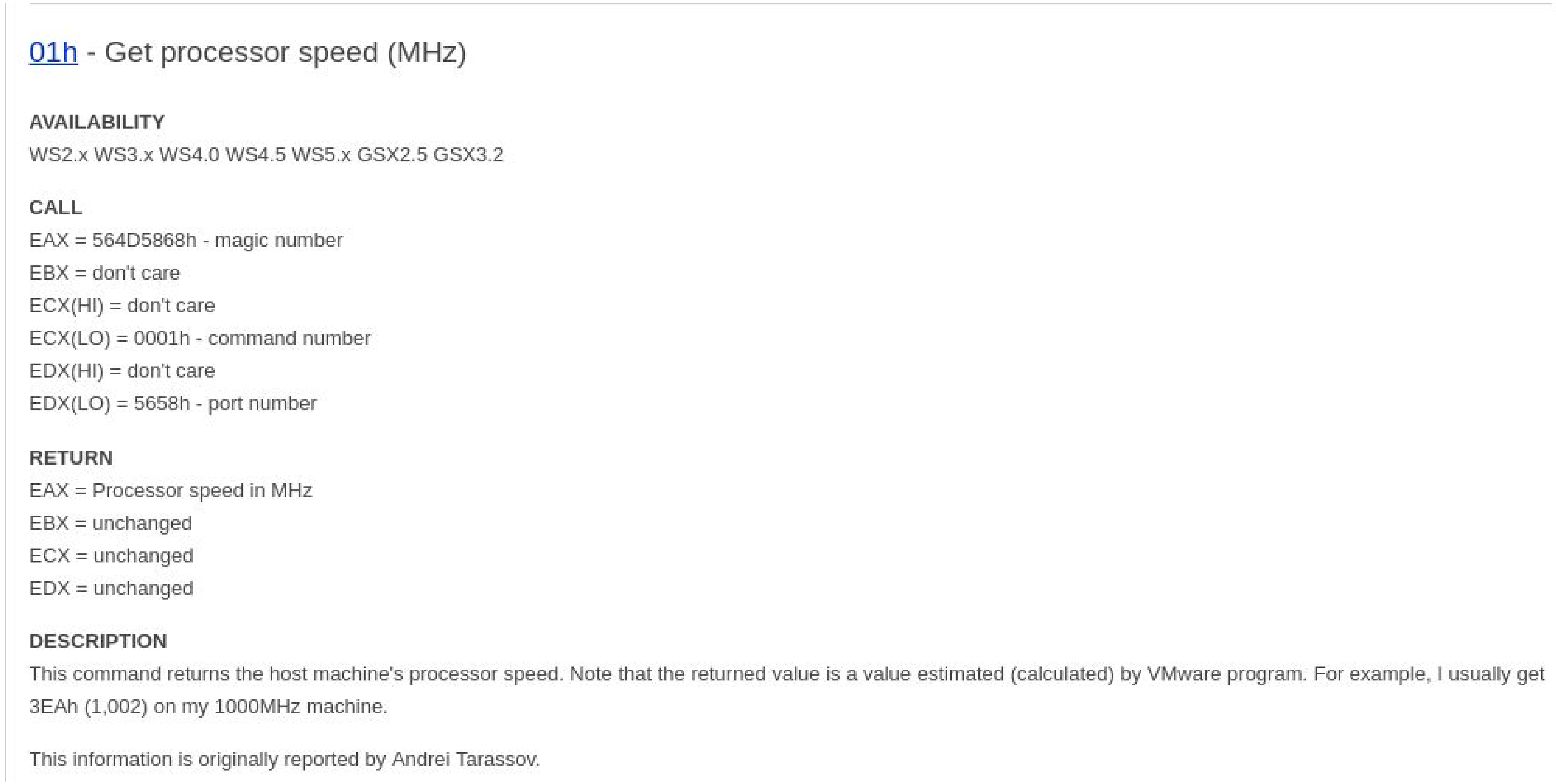


DLL Skeleton



```
Emulate proc
LOCAL psave

    pushad
    mov psave, ecx
    mov eax, psave
    mov ecx, dword ptr [eax+8]
    .if ecx == 1
    .elseif ecx == 4
    .elseif ecx == 6
    .elseif ecx == 7
    .elseif ecx == 0Fh
    .elseif ecx == 11h
    .elseif ecx == 13h
    .elseif ecx == 17h
    .endif
    push eax
    mov eax, psave
    mov dword ptr [eax+14h], edi
    mov dword ptr [eax+10h], esi
    mov dword ptr [eax+0ch], edx
    mov dword ptr [eax+8h], ecx
    mov dword ptr [eax+4h], ebx
    mov ebx, eax
    pop eax
    mov dword ptr [ebx], eax
    popad

    ret

Emulate endp
```

Function 01h



01h - Get processor speed (MHz)

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5 WS5.x GSX2.5 GSX3.2

CALL

EAX = 564D5868h - magic number
EBX = don't care
ECX(HI) = don't care
ECX(LO) = 0001h - command number
EDX(HI) = don't care
EDX(LO) = 5658h - port number

RETURN

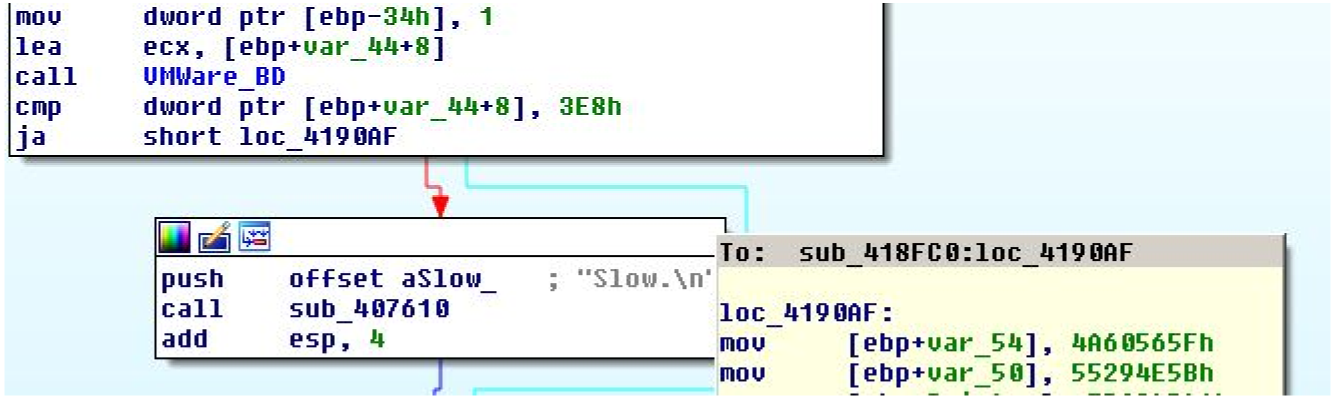
EAX = Processor speed in MHz
EBX = unchanged
ECX = unchanged
EDX = unchanged

DESCRIPTION

This command returns the host machine's processor speed. Note that the returned value is a value estimated (calculated) by VMware program. For example, I usually get 3EAh (1,002) on my 1000MHz machine.

This information is originally reported by Andrei Tarassov.

Function 01h



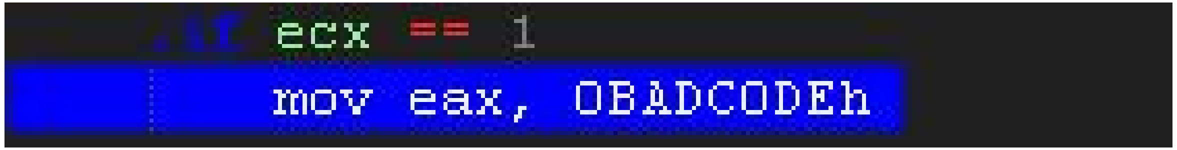
```
mov     dword ptr [ebp-34h], 1
lea     ecx, [ebp+var_44+8]
call    VMWare_BD
cmp     dword ptr [ebp+var_44+8], 3E8h
ja      short loc_4190AF
```

```
push    offset aSlow_    ; "Slow.\n"
call    sub_407610
add     esp, 4
```

```
To:  sub_418FC0:loc_4190AF

loc_4190AF:
mov     [ebp+var_54], 4A60565Fh
mov     [ebp+var_50], 55294E5Bh
```

3E8h = 1000d

Function 01h



```
.if ecx == 1
    mov eax, 0BADC0DEh
```

Function 11h

11h - Get virtual hardware version

AVAILABILITY

WS5.x

CALL

EAX = 564D5868h - magic number
EBX = don't care
ECX(HI) = don't care
ECX(LO) = 0011h - command number
EDX(HI) = don't care
EDX(LO) = 5658h - port number

RETURN

EAX = virtual hardware version
EBX = unchanged
ECX = unchanged
EDX = unchanged

DESCRIPTION

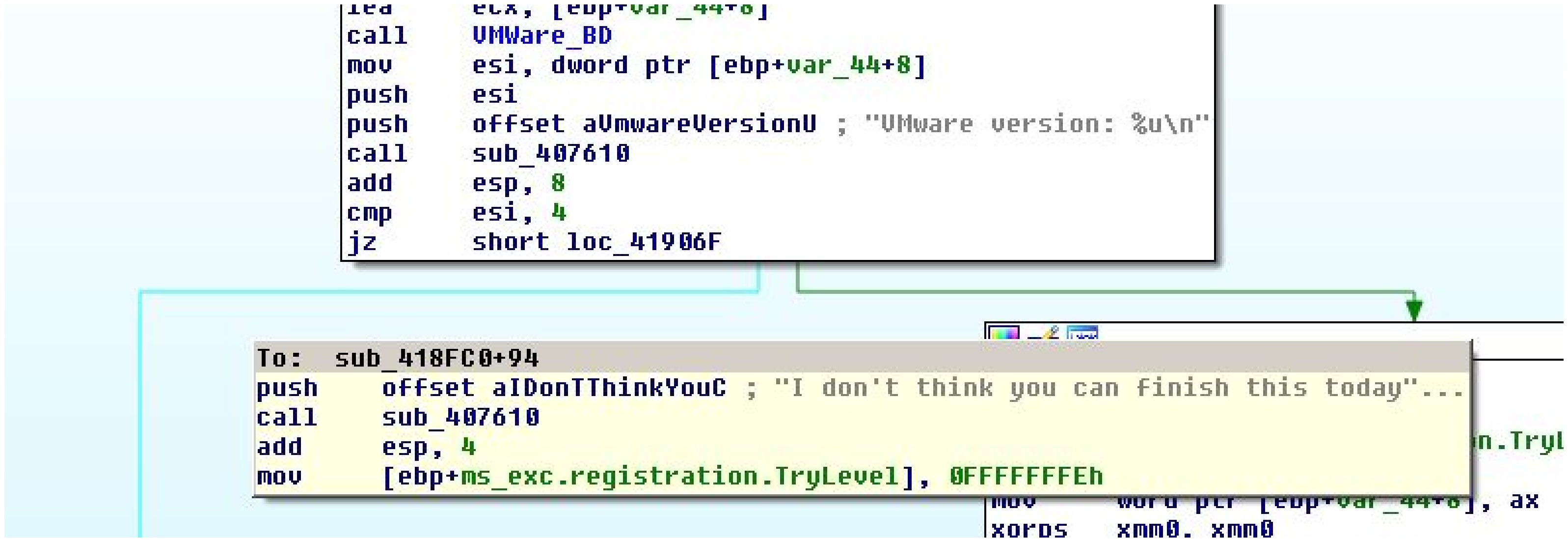
This command returns the virtual hardware version of the current virtual machine.
Possible version numbers are:

- 3: Virtual machines created with WS4.x, ESX2.x, GSX3.x, ACE1.x, and with WS5.x as a legacy VM
- 4: Virtual machines created with WS5.x as a new type VM

Although virtual machines created with WS3.x/GSX2.x also have a virtual hardware version (1 or 2), they can not run on WS5.x without first upgrading the virtual hardware and therefore this command never returns such values.

Note: Command 11h is also implemented in WS2.x but it seems to have a different function and I don't know what.

Function 11h



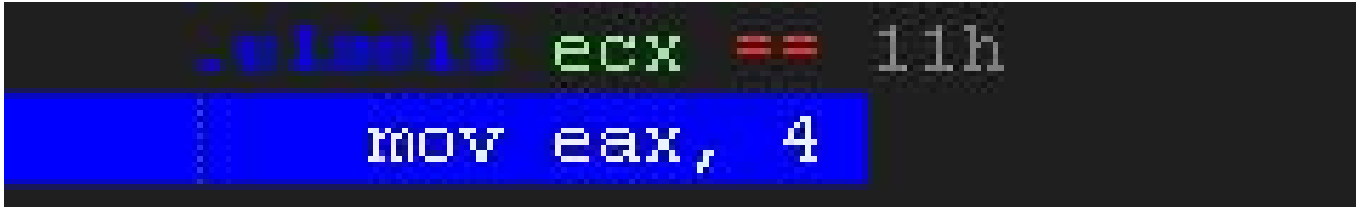
```
call    VMWare_BD
mov     esi, dword ptr [ebp+var_44+8]
push    esi
push    offset aVmwareVersionU ; "VMware version: %u\n"
call    sub_407610
add     esp, 8
cmp     esi, 4
jz      short loc_41906F
```

```
To:  sub_418FC0+94
push    offset aIDonTThinkYouC ; "I don't think you can finish this today"...
call    sub_407610
add     esp, 4
mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
```

```
xorps   xmm0, xmm0
```

Version == 4

Function 11h



```
.elseif ecx == 11h
        mov eax, 4
```

Function 0Fh

0Fh - Get host screen size

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5 WS5.x GSX2.5 GSX 3.2

CALL

EAX = 564D5868h - magic number

EBX = don't care

ECX(HI) = don't care

ECX(LO) = 000Fh - command number

EDX(HI) = don't care

EDX(LO) = 5658h - port number

RETURN

EAX(HI) = X resolution (pixels)

EAX(LO) = Y resolution (pixels)

EBX = unchanged

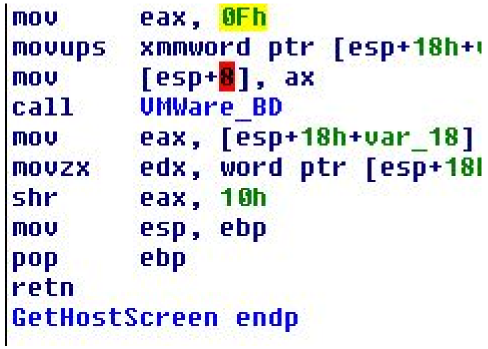
ECX = unchanged

EDX = unchanged

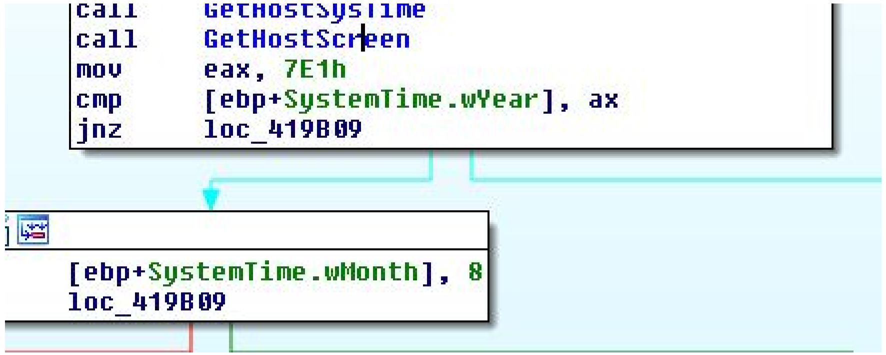
DESCRIPTION

This command returns the host's screen size.

Function 0Fh



```
mov     eax, 0Fh
movups  xmmword ptr [esp+18h+
mov     [esp+8], ax
call    VMWare_BD
mov     eax, [esp+18h+var_18]
movzx   edx, word ptr [esp+18
shr     eax, 10h
mov     esp, ebp
pop     ebp
retn
GetHostScreen endp
```

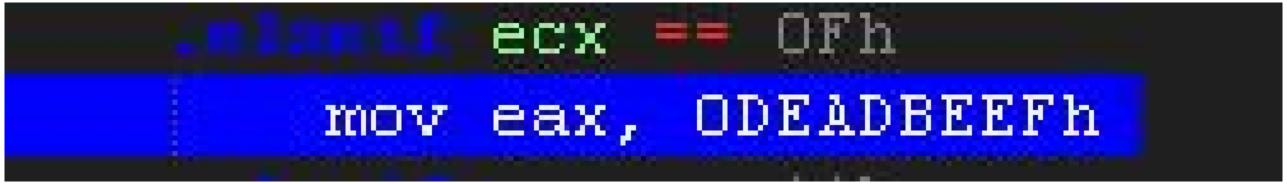


```
call    GetHostSysTime
call    GetHostScreen
mov     eax, 7E1h
cmp     [ebp+SystemTime.wYear], ax
jnz     loc_419B09
```

```
        [ebp+SystemTime.wMonth], 8
        loc_419B09
```

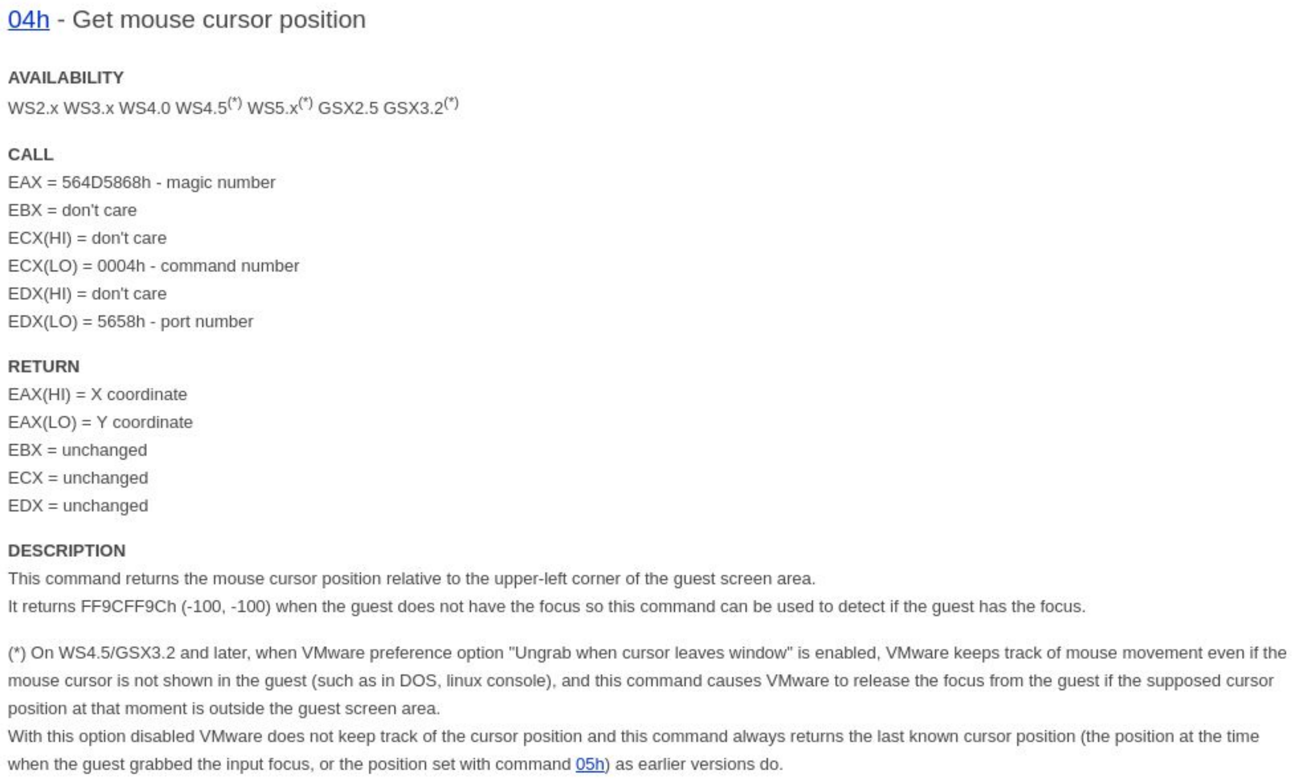
Not used??

Function 0Fh



```
.elseif ecx == 0Fh
        mov eax, 0DEADBEEFh
```

Function 04h



04h - Get mouse cursor position

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5$^{(*)}$ WS5.x$^{(*)}$ GSX2.5 GSX3.2$^{(*)}$

CALL

EAX = 564D5868h - magic number
EBX = don't care
ECX(HI) = don't care
ECX(LO) = 0004h - command number
EDX(HI) = don't care
EDX(LO) = 5658h - port number

RETURN

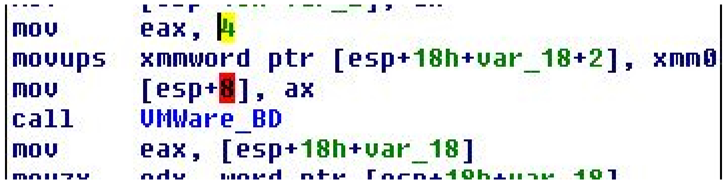
EAX(HI) = X coordinate
EAX(LO) = Y coordinate
EBX = unchanged
ECX = unchanged
EDX = unchanged

DESCRIPTION

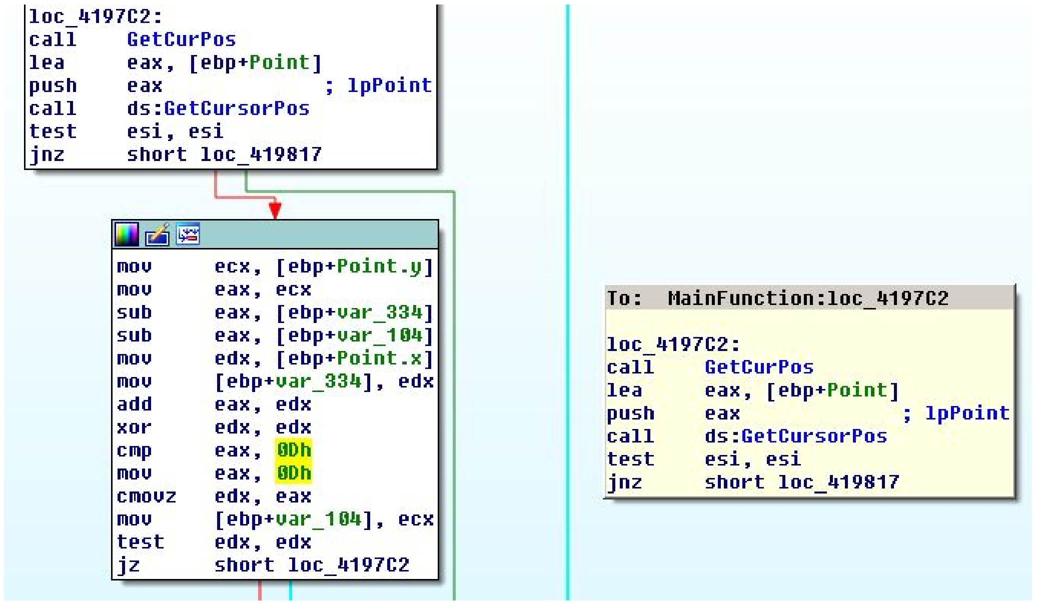
This command returns the mouse cursor position relative to the upper-left corner of the guest screen area.
It returns FF9CFF9Ch (-100, -100) when the guest does not have the focus so this command can be used to detect if the guest has the focus.

(*) On WS4.5/GSX3.2 and later, when VMware preference option "Ungrab when cursor leaves window" is enabled, VMware keeps track of mouse movement even if the mouse cursor is not shown in the guest (such as in DOS, linux console), and this command causes VMware to release the focus from the guest if the supposed cursor position at that moment is outside the guest screen area.
With this option disabled VMware does not keep track of the cursor position and this command always returns the last known cursor position (the position at the time when the guest grabbed the input focus, or the position set with command 05h) as earlier versions do.

Function 04h



```
mov      eax, 4
movups   xmmword ptr [esp+18h+var_18+2], xmm0
mov      [esp+8], ax
call     VMWare_BD
mov      eax, [esp+18h+var_18]
```

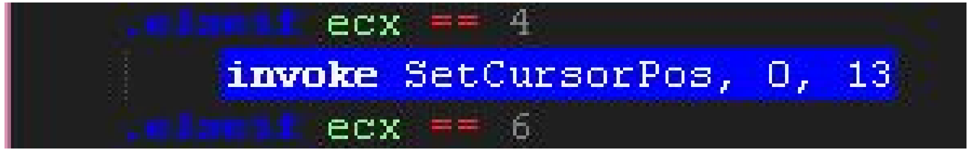


```
loc_4197C2:
call     GetCurPos
lea      eax, [ebp+Point]
push     eax                  ; lpPoint
call     ds:GetCursorPos
test     esi, esi
jnz      short loc_419817
```

```
mov      ecx, [ebp+Point.y]
mov      eax, ecx
sub      eax, [ebp+var_334]
sub      eax, [ebp+var_104]
mov      edx, [ebp+Point.x]
mov      [ebp+var_334], edx
add      eax, edx
xor      edx, edx
cmp      eax, 0Dh
mov      eax, 0Dh
cmovz    edx, eax
mov      [ebp+var_104], ecx
test     edx, edx
jz       short loc_4197C2
```

```
To:  MainFunction:loc_4197C2

loc_4197C2:
call     GetCurPos
lea      eax, [ebp+Point]
push     eax                  ; lpPoint
call     ds:GetCursorPos
test     esi, esi
jnz      short loc_419817
```

Function 04h



```
.elseif ecx == 4
    invoke SetCursorPos, 0, 13
.elseif ecx == 6
```

Function 17h

17h - Get host's system time (GMT)

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5 WS5.x GSX2.5 GSX3.2

CALL

EAX = 564D5868h - magic number
EBX = don't care
ECX(HI) = don't care
ECX(LO) = 0017h - command number
EDX(HI) = don't care
EDX(LO) = 5658h - port number

RETURN

EAX = host's system time (GMT, unix style 32 bit time value)
EBX = microsecond
ECX = ?
EDX = host's timezone (offset to GMT in minutes, 32 bit signed integer) on WS4.0/GSX2.5 and earlier / 0 on WS4.5/GSX3.2 and later

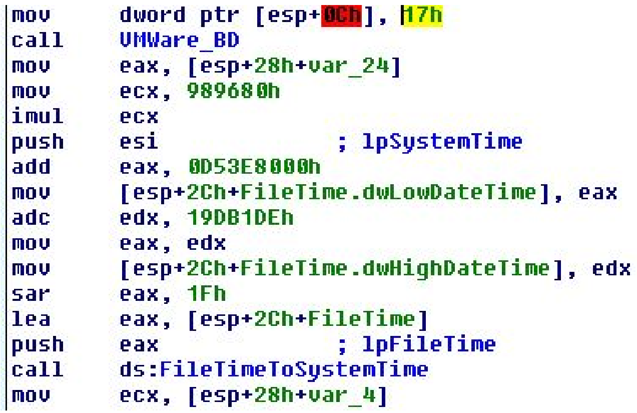
DESCRIPTION

This command returns the host's system time (GMT).
On WS4.0/GSX2.5 and earlier, you can get the host's local time by subtracting the offset (*EDX*) from the GMT time (*EAX*). Since the offset is returned in minutes rather than in seconds, the expression should be like:

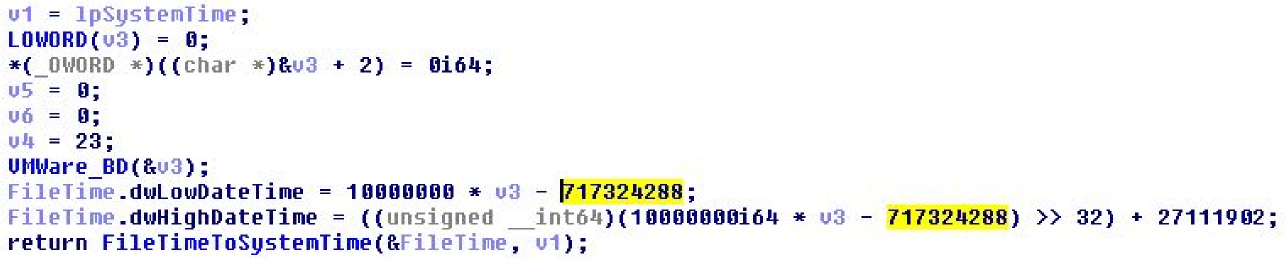
```
        localtime = EAX - (EDX * 60)
```

Another seemingly meaningful value is returned in *ECX* but I still don't know what it is. For the record, on my WS5.5 it is almost always 000F4240h (1,000,000) but somethimes changes between 1,000,000 and about 3,000,000. I'd imagine it has something to do with the clock precision. Maybe.

Function 17h

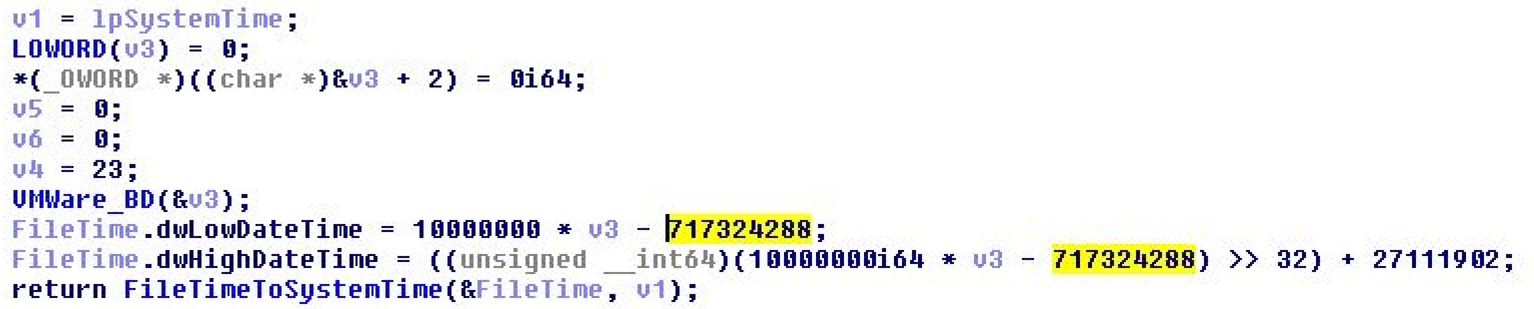


```
mov     dword ptr [esp+0Ch], 17h
call    VMWare_BD
mov     eax, [esp+28h+var_24]
mov     ecx, 989680h
imul    ecx
push    esi             ; lpSystemTime
add     eax, 0D53E8000h
mov     [esp+2Ch+FileTime.dwLowDateTime], eax
adc     edx, 19DB1DEh
mov     eax, edx
mov     [esp+2Ch+FileTime.dwHighDateTime], edx
sar     eax, 1Fh
lea     eax, [esp+2Ch+FileTime]
push    eax             ; lpFileTime
call    ds:FileTimeToSystemTime
mov     ecx, [esp+28h+var_4]
```

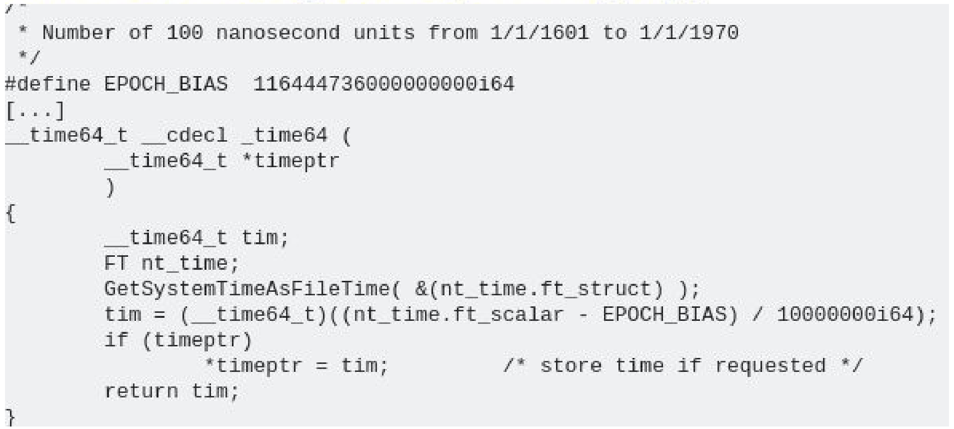


```
v1 = lpSystemTime;
LOWORD(v3) = 0;
*(_OWORD *)((char *)&v3 + 2) = 0i64;
v5 = 0;
v6 = 0;
v4 = 23;
VMWare_BD(&v3);
FileTime.dwLowDateTime = 10000000 * v3 - 717324288;
FileTime.dwHighDateTime = ((unsigned __int64)(10000000i64 * v3 - 717324288) >> 32) + 27111902;
return FileTimeToSystemTime(&FileTime, v1);
```

Function 17h

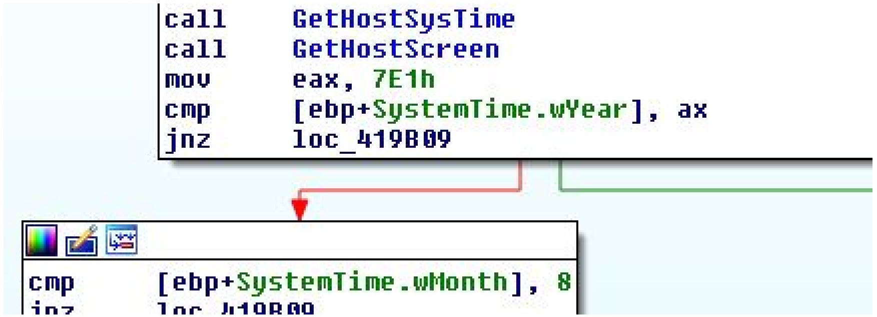


```
v1 = lpSystemTime;
LOWORD(v3) = 0;
*(_OWORD *)((char *)&v3 + 2) = 0i64;
v5 = 0;
v6 = 0;
v4 = 23;
VMWare_BD(&v3);
FileTime.dwLowDateTime = 10000000 * v3 - 717324288;
FileTime.dwHighDateTime = ((unsigned __int64)(10000000i64 * v3 - 717324288) >> 32) + 27111902;
return FileTimeToSystemTime(&FileTime, v1);
```



```
/*
 * Number of 100 nanosecond units from 1/1/1601 to 1/1/1970
 */
#define EPOCH_BIAS  116444736000000000i64
[...]
__time64_t __cdecl _time64 (
        __time64_t *timeptr
        )
{
        __time64_t tim;
        FT nt_time;
        GetSystemTimeAsFileTime( &(nt_time.ft_struct) );
        tim = (__time64_t)((nt_time.ft_scalar - EPOCH_BIAS) / 10000000i64);
        if (timeptr)
                *timeptr = tim;         /* store time if requested */
        return tim;
}
```

Function 17h



```
call    GetHostSysTime
call    GetHostScreen
mov     eax, 7E1h
cmp     [ebp+SystemTime.wYear], ax
jnz     loc_419B09
```

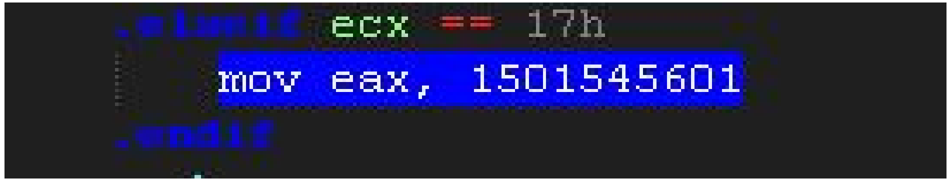
```
cmp     [ebp+SystemTime.wMonth], 8
jnz     loc_419B09
```

EPOCH of Year = 2017, Month = 08

2017-08-01 to 2017-08-31

1501545601 to 1504155599

Function 17h



```
.elseif ecx == 17h
    mov eax, 1501545601
.endif
```

Function 13h

13h - Get BIOS UUID

AVAILABILITY

WS4.0 WS4.5 WS5.x GSX2.5 GSX3.2

CALL

EAX = 564D5868h - magic number

EBX = don't care

ECX(HI) = don't care

ECX(LO) = 0013h - command number

EDX(HI) = don't care

EDX(LO) = 5658h - port number

RETURN

EAX = 1st 4 bytes of the UUID (first byte in LSB)

EBX = 2nd 4 bytes of the UUID (ditto)

ECX = 3rd 4 bytes of the UUID (ditto)

EDX = 4th 4 bytes of the UUID (ditto)

DESCRIPTION

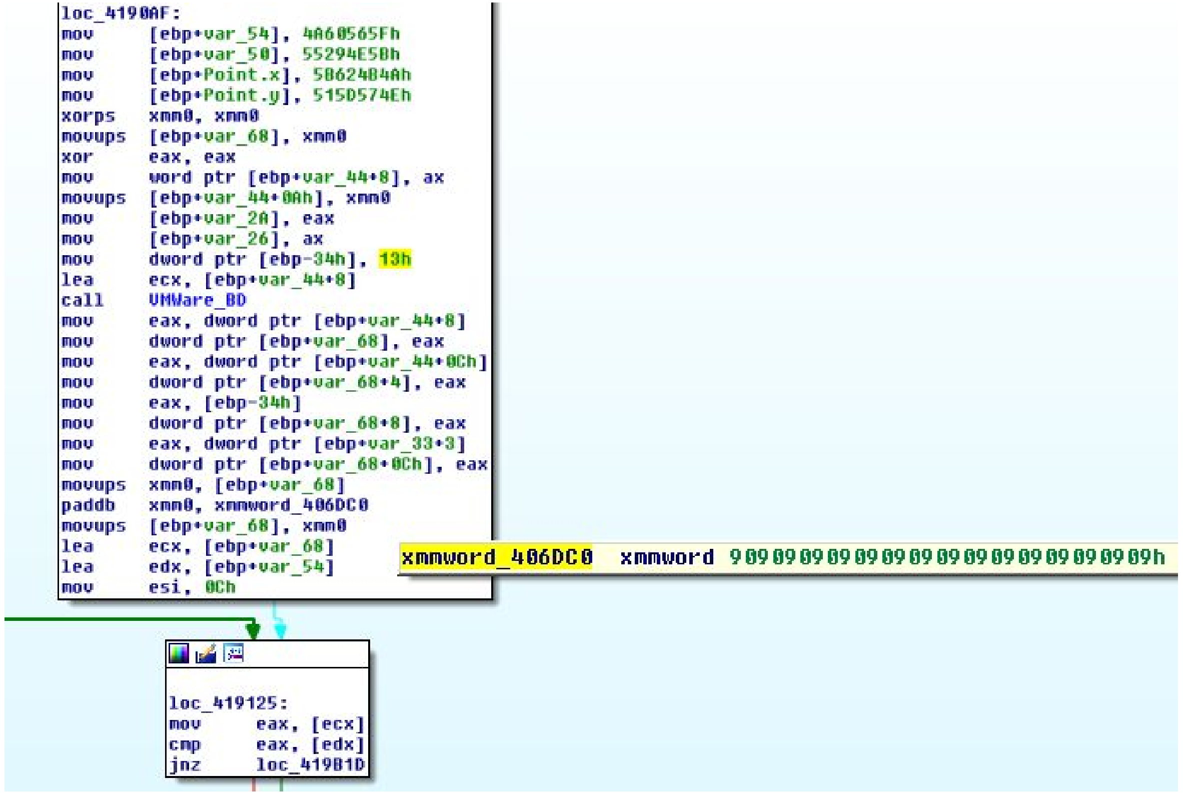
This command returns the BIOS UUID of the current virtual machine. BIOS UUID is stored in the config file in the following form:

```
uuid.bios = "56 4d 3e 7a 92 ee 4c 46-e8 0d 86 f3 68 a0 cb e7"
```

With this command, the UUID illustrated above is returned in each registor in the following form:

```
EAX: 7a3e4d56
EBX: 464cee92
ECX: f3860de8
EDX: e7cba068
```

Function 13h

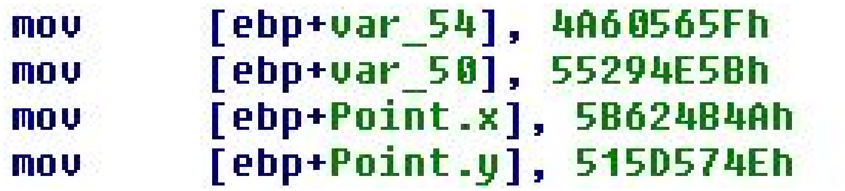


```
loc_4190AF:
mov     [ebp+var_54], 4A60565Fh
mov     [ebp+var_50], 55294E50h
mov     [ebp+Point.x], 5B62484Ah
mov     [ebp+Point.y], 515D574Eh
xorps   xmm0, xmm0
movups  [ebp+var_68], xmm0
xor     eax, eax
mov     word ptr [ebp+var_44+8], ax
movups  [ebp+var_44+0Ah], xmm0
mov     [ebp+var_2A], eax
mov     [ebp+var_26], ax
mov     dword ptr [ebp-34h], 13h
lea     ecx, [ebp+var_44+8]
call    VMWare_BD
mov     eax, dword ptr [ebp+var_44+8]
mov     dword ptr [ebp+var_68], eax
mov     eax, dword ptr [ebp+var_44+0Ch]
mov     dword ptr [ebp+var_68+4], eax
mov     eax, [ebp-34h]
mov     dword ptr [ebp+var_68+8], eax
mov     eax, dword ptr [ebp+var_33+3]
mov     dword ptr [ebp+var_68+0Ch], eax
movups  xmm0, [ebp+var_68]
paddb   xmm0, xmmword_406DC0
movups  [ebp+var_68], xmm0
lea     ecx, [ebp+var_68]
lea     edx, [ebp+var_54]
mov     esi, 0Ch
```

```
xmmword_406DC0   xmmword 9090909090909090909090909090909h
```

```
loc_419125:
mov     eax, [ecx]
cmp     eax, [edx]
jnz     loc_4191DD
```

Function 13h



```
mov     [ebp+var_54], 4A60565Fh
mov     [ebp+var_50], 55294E5Bh
mov     [ebp+Point.x], 5B624B4Ah
mov     [ebp+Point.y], 515D574Eh
```

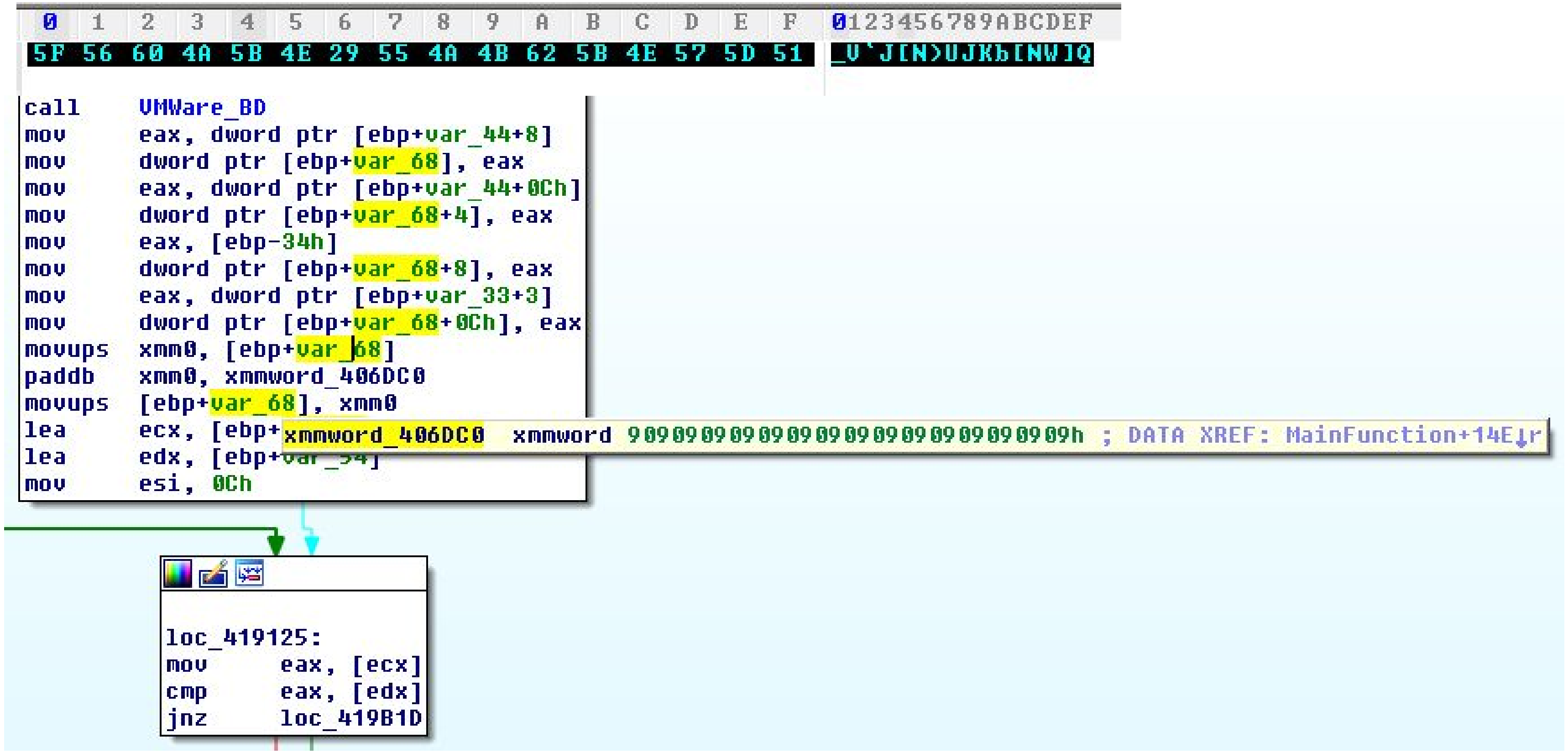
4A 60 56 5F ➪ 5F 56 60 4A

55 29 4E 5B ➪ 5B 4E 29 55

5B 62 4B 4A ➪ 4A 4B 62 5B

51 5D 57 4E ➪ 4E 57 5D 51

Function 13h

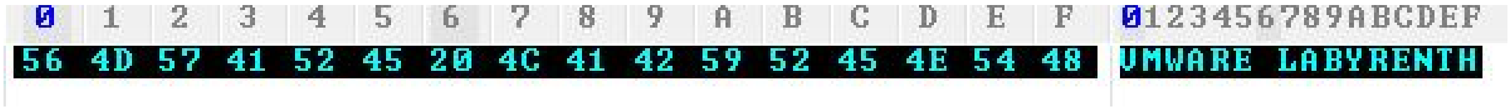


```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
5F 56 60 4A 5B 4E 29 55 4A 4B 62 5B 4E 57 5D 51   _V`J[N)UJKb[NW]Q
```

```
call    VMWare_BD
mov     eax, dword ptr [ebp+var_44+8]
mov     dword ptr [ebp+var_68], eax
mov     eax, dword ptr [ebp+var_44+0Ch]
mov     dword ptr [ebp+var_68+4], eax
mov     eax, [ebp-34h]
mov     dword ptr [ebp+var_68+8], eax
mov     eax, dword ptr [ebp+var_33+3]
mov     dword ptr [ebp+var_68+0Ch], eax
movups  xmm0, [ebp+var_68]
paddb   xmm0, xmmword_406DC0
movups  [ebp+var_68], xmm0
lea     ecx, [ebp+xmmword_406DC0  xmmword 9090909090909090909090909090909h ; DATA XREF: MainFunction+14E↓r
lea     edx, [ebp+var_54]
mov     esi, 0Ch
```

```
loc_419125:
mov     eax, [ecx]
cmp     eax, [edx]
jnz     loc_419B1D
```

Function 13h



```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
56 4D 57 41 52 45 20 4C 41 42 59 52 45 4E 54 48   VMWARE LABYRENTH
```

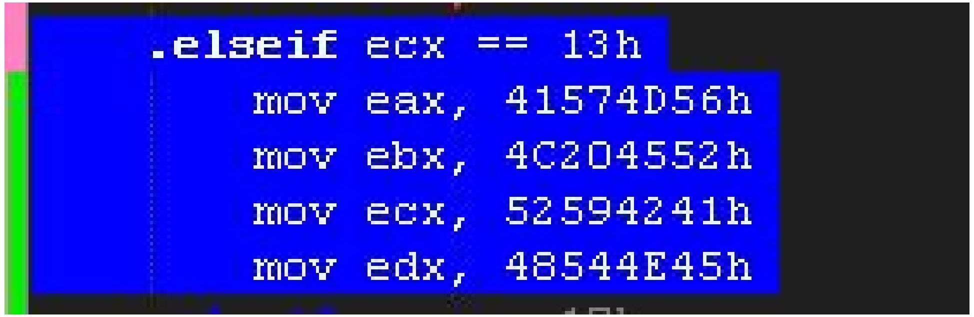
56 4D 57 41 ➪ 41 57 4D 56

52 45 20 4C ➪ 4C 20 45 52

41 42 59 52 ➪ 52 59 42 41

45 4E 54 48 ➪ 48 54 4E 45

Function 13h



```
.elseif ecx == 13h
    mov eax, 41574D56h
    mov ebx, 4C204552h
    mov ecx, 52594241h
    mov edx, 48544E45h
```

Function 6h & 7h

06h - Get text length from clipboard

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5(*) WS5.x(*) GSX2.5 GSX3.2(*)

CALL

EAX = 564D5868h - magic number

EBX = don't care

ECX(HI) = don't care

ECX(LO) = 0006h - command number

EDX(HI) = don't care

EDX(LO) = 5658h - port number

RETURN

EAX = text length

EBX = unchanged

ECX = unchanged

EDX = unchanged

DESCRIPTION

This command returns the length of text data available from the host's clipboard. Also this command resets the clipboard data transfer process. The first get text command (07h) issued after this command returns the very beginning of the text data.

Function 6h & 7h

07h - Get text from clipboard

AVAILABILITY

WS2.x WS3.x WS4.0 WS4.5 WS5.x GSX2.5 GSX3.2

CALL

EAX = 564D5868h - magic number
EBX = don't care
ECX(HI) = don't care
ECX(LO) = 0007h - command number
EDX(HI) = don't care
EDX(LO) = 5658h - port number

RETURN

EAX = 4 bytes of text from clipboard (first byte in LSB)
EBX = unchanged
ECX = unchanged
EDX = unchanged

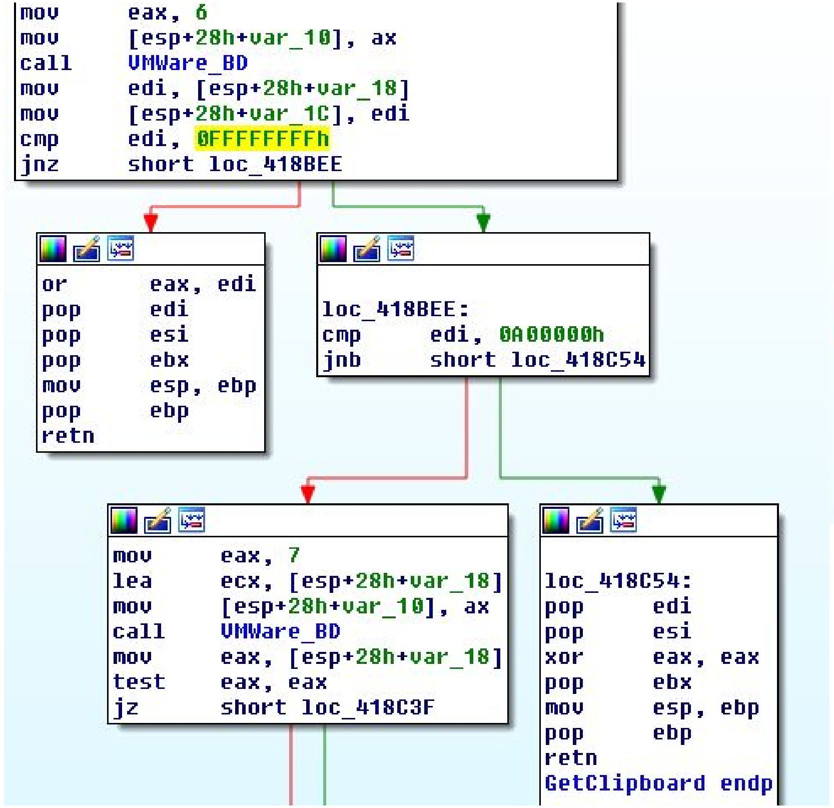
DESCRIPTION

This command return a portion of text in the clipboard. The get text length command (06h) should be called prior to this command. The first call after a command 06h call returns the first 4 bytes, and the next call returns the next 4 bytes, and so on.
If no more data is available in the clipboard, 00000000h is returned.
This command does not pass carrage return characters (0Dh), so if the guest OS requires them (e.g. DOS), you have to supply them as line feed characters (0Ah) appear in returned text data.

On WS4.0/GSX2.5 and earlier the data length value returned by the get length command (06h) is often slightly larger than actual text length so you should search for a terminating null character in returned text data to know the actual end of the text.

See notes on WS4.5/GSX3.2 and later in command 06h above.

Function 6h & 7h



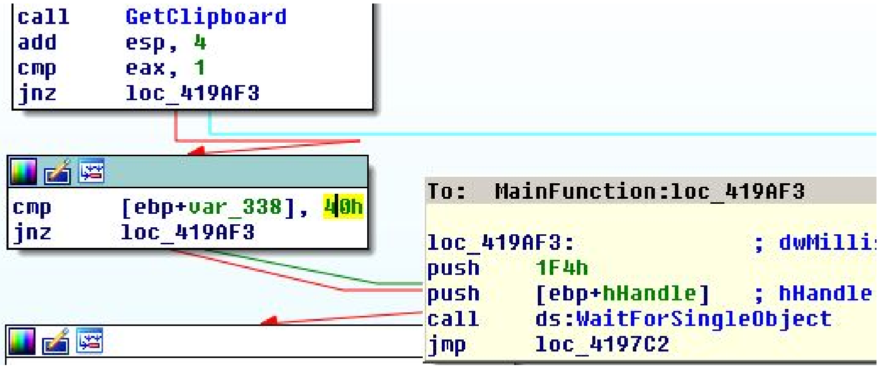
```
mov     eax, 6
mov     [esp+28h+var_10], ax
call    VMWare_BD
mov     edi, [esp+28h+var_18]
mov     [esp+28h+var_1C], edi
cmp     edi, 0FFFFFFFFh
jnz     short loc_418BEE
```

```
or      eax, edi
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

```
loc_418BEE:
cmp     edi, 0A00000h
jnb     short loc_418C54
```

```
mov     eax, 7
lea     ecx, [esp+28h+var_18]
mov     [esp+28h+var_10], ax
call    VMWare_BD
mov     eax, [esp+28h+var_18]
test    eax, eax
jz      short loc_418C3F
```

```
loc_418C54:
pop     edi
pop     esi
xor     eax, eax
pop     ebx
mov     esp, ebp
pop     ebp
retn
GetClipboard endp
```

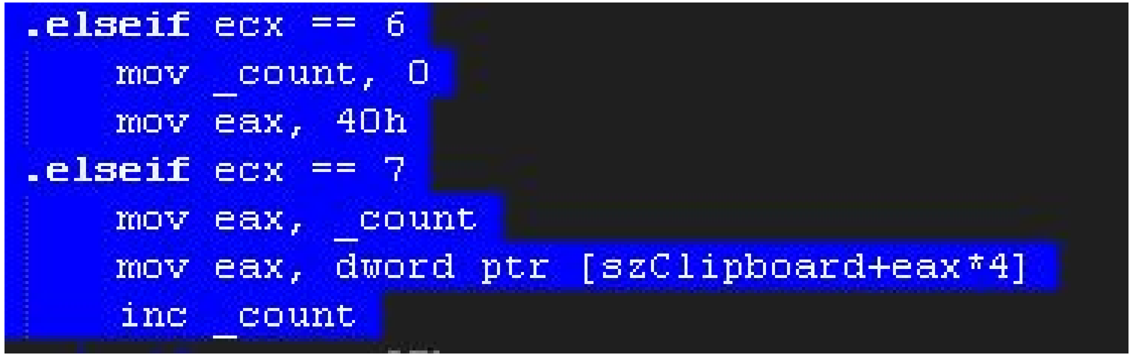


```
call    GetClipboard
add     esp, 4
cmp     eax, 1
jnz     loc_419AF3
```

```
cmp     [ebp+var_338], 40h
jnz     loc_419AF3
```

```
To:  MainFunction:loc_419AF3

loc_419AF3:                 ; dwMilli
push    1F4h
push    [ebp+hHandle]       ; hHandle
call    ds:WaitForSingleObject
jmp     loc_4197C2
```

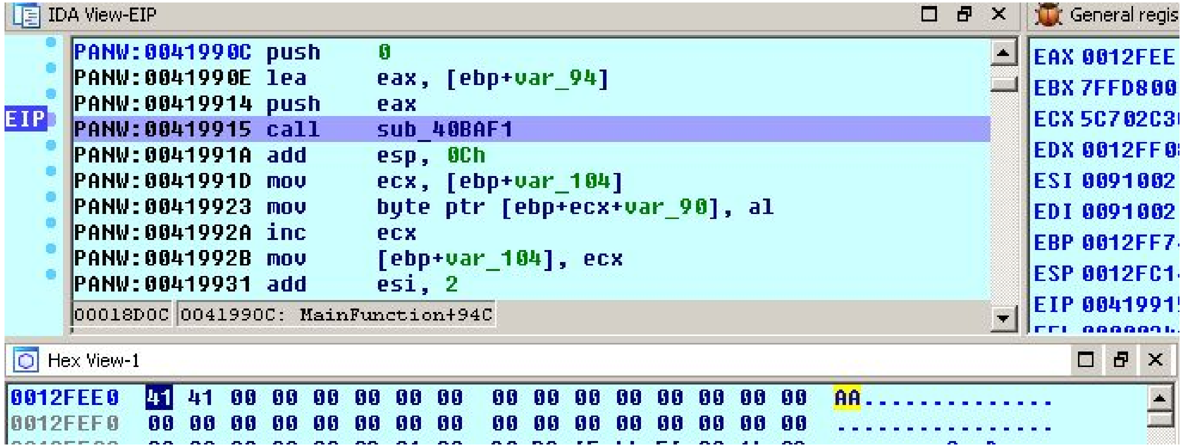
Function 6h & 7h



```
.elseif ecx == 6
    mov _count, 0
    mov eax, 40h
.elseif ecx == 7
    mov eax, _count
    mov eax, dword ptr [szClipboard+eax*4]
    inc _count
```

Function 6h & 7h

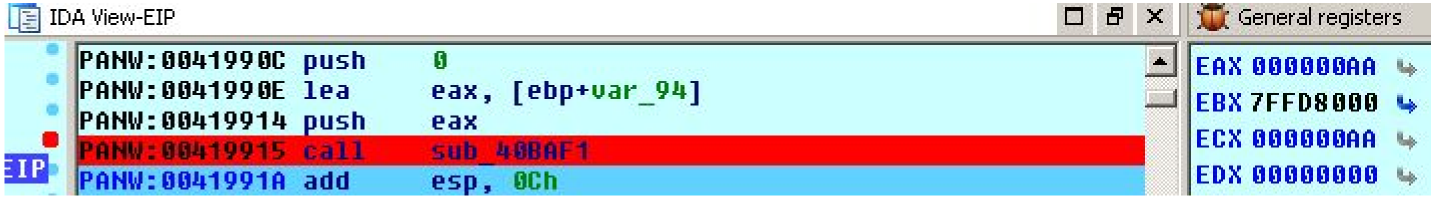
Buffer = ‘A’ * 40h (AAAAAAA…..)



```
PANW:0041990C push    0
PANW:0041990E lea     eax, [ebp+var_94]
PANW:00419914 push    eax
PANW:00419915 call    sub_40BAF1
PANW:0041991A add     esp, 0Ch
PANW:0041991D mov     ecx, [ebp+var_104]
PANW:00419923 mov     byte ptr [ebp+ecx+var_90], al
PANW:0041992A inc     ecx
PANW:0041992B mov     [ebp+var_104], ecx
PANW:00419931 add     esi, 2
00018D0C 0041990C: MainFunction+94C
```

```
EAX 0012FEE
EBX 7FFD800
ECX 5C702C3
EDX 0012FF0
ESI 0091002
EDI 0091002
EBP 0012FF7
ESP 0012FC1
EIP 0041991
```

```
0012FEE0  41 41 00 00 00 00 00 00  00 00 00 00 00 00 00 00  AA..............
0012FEF0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

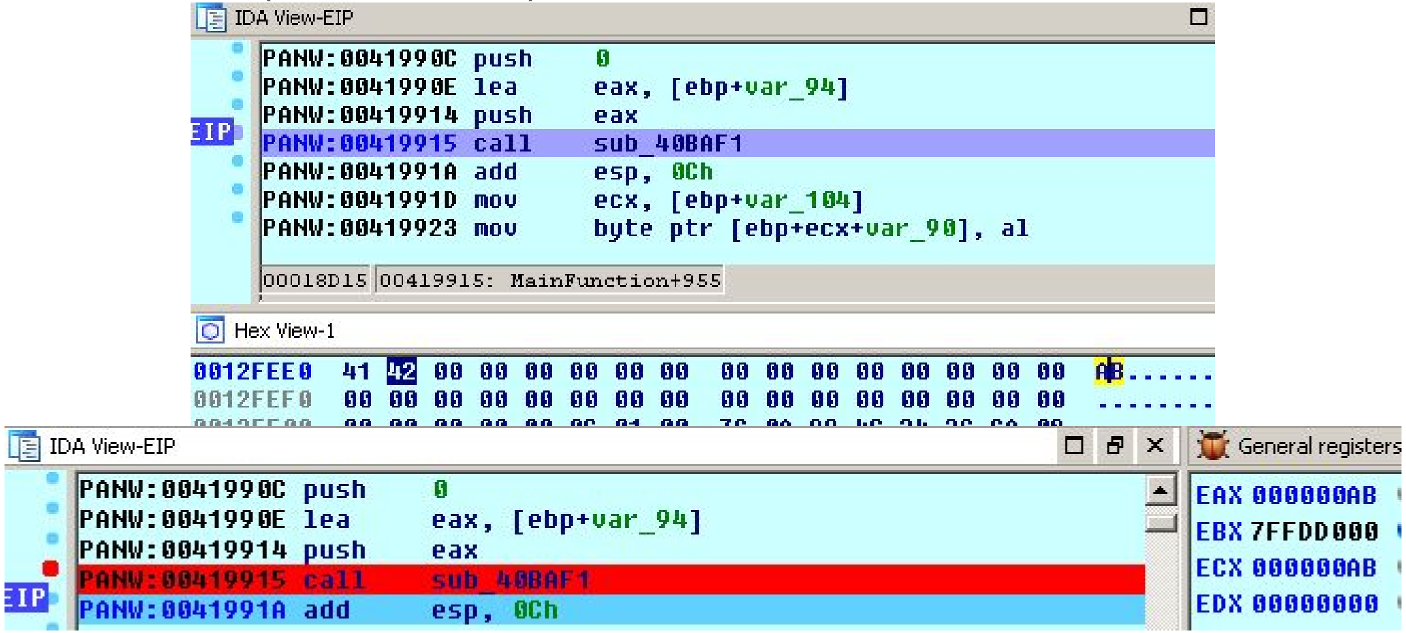


```
PANW:0041990C push    0
PANW:0041990E lea     eax, [ebp+var_94]
PANW:00419914 push    eax
PANW:00419915 call    sub_40BAF1
PANW:0041991A add     esp, 0Ch
```

```
EAX 000000AA
EBX 7FFD8000
ECX 000000AA
EDX 00000000
```

Function 6h & 7h

Buffer = ‘AB’ * 20h (ABABAB….)



IDA View-EIP

```
PANW:0041990C push    0
PANW:0041990E lea     eax, [ebp+var_94]
PANW:00419914 push    eax
PANW:00419915 call    sub_40BAF1
PANW:0041991A add     esp, 0Ch
PANW:0041991D mov     ecx, [ebp+var_104]
PANW:00419923 mov     byte ptr [ebp+ecx+var_90], al

00018D15 00419915: MainFunction+955
```

Hex View-1

```
0012FEE0  41 42 00 00 00 00 00 00  00 00 00 00 00 00 00 00  AB......
0012FEF0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ........
```

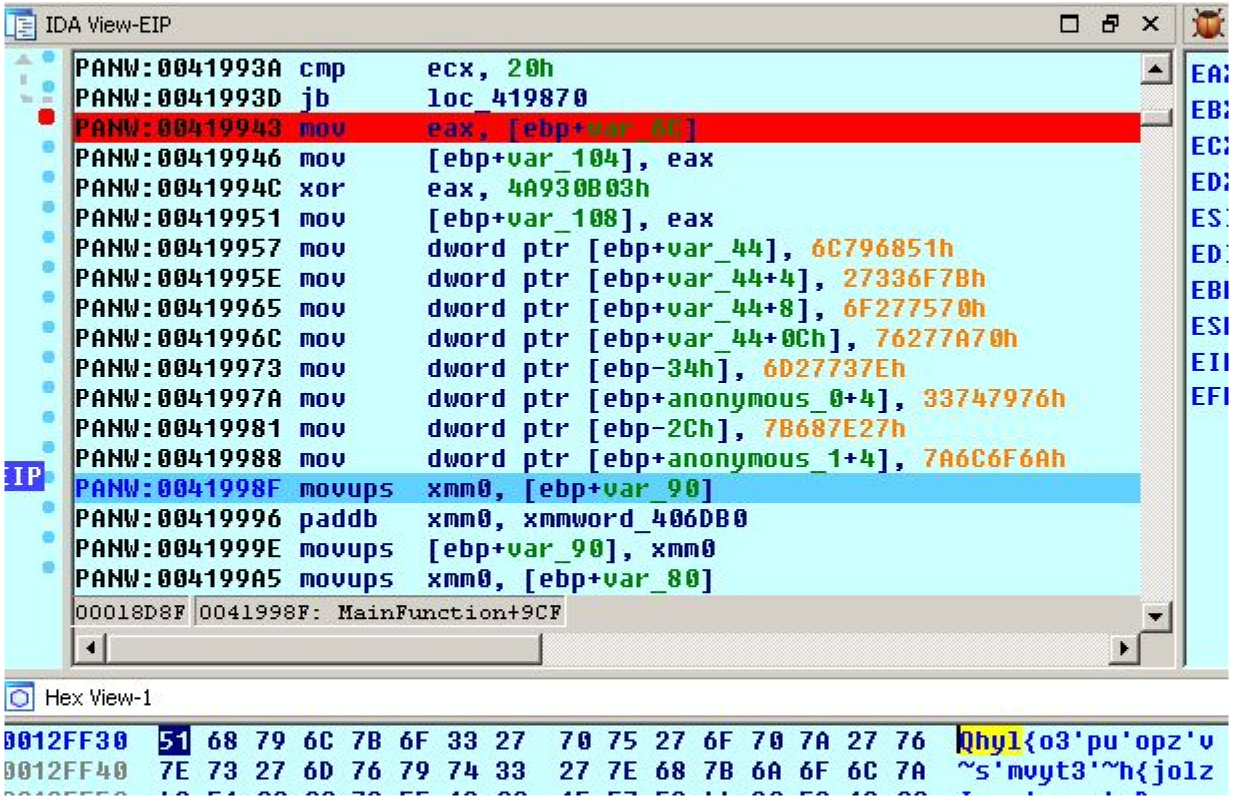
IDA View-EIP

```
PANW:0041990C push    0
PANW:0041990E lea     eax, [ebp+var_94]
PANW:00419914 push    eax
PANW:00419915 call    sub_40BAF1
PANW:0041991A add     esp, 0Ch
```

General registers

```
EAX 000000AB
EBX 7FFDD000
ECX 000000AB
EDX 00000000
```

Function 6h & 7h

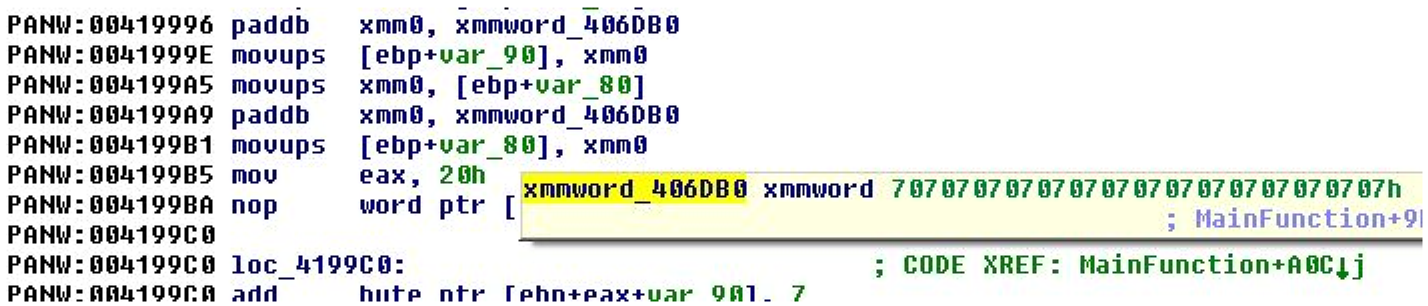


```
PANW:0041993A cmp     ecx, 20h
PANW:0041993D jb      loc_419870
PANW:00419943 mov     eax, [ebp+var_6C]
PANW:00419946 mov     [ebp+var_104], eax
PANW:0041994C xor     eax, 4A930B03h
PANW:00419951 mov     [ebp+var_108], eax
PANW:00419957 mov     dword ptr [ebp+var_44], 6C796851h
PANW:0041995E mov     dword ptr [ebp+var_44+4], 27336F7Bh
PANW:00419965 mov     dword ptr [ebp+var_44+8], 6F277570h
PANW:0041996C mov     dword ptr [ebp+var_44+0Ch], 76277A70h
PANW:00419973 mov     dword ptr [ebp-34h], 6D27737Eh
PANW:0041997A mov     dword ptr [ebp+anonymous_0+4], 33747976h
PANW:00419981 mov     dword ptr [ebp-2Ch], 7B687E27h
PANW:00419988 mov     dword ptr [ebp+anonymous_1+4], 7A6C6F6Ah
PANW:0041998F movups  xmm0, [ebp+var_90]
PANW:00419996 paddb   xmm0, xmmword_406DB0
PANW:0041999E movups  [ebp+var_90], xmm0
PANW:004199A5 movups  xmm0, [ebp+var_80]
00018D8F 0041998F: MainFunction+9CF
```

```
0012FF30  51 68 79 6C 7B 6F 33 27  70 75 27 6F 70 7A 27 76  Qhyl{o3'pu'opz'v
0012FF40  7E 73 27 6D 76 79 74 33  27 7E 68 7B 6A 6F 6C 7A  ~s'mvyt3'~h{jolz
```

EA
EB
EC
ED
ES
ED
EB
ES
EI
EF

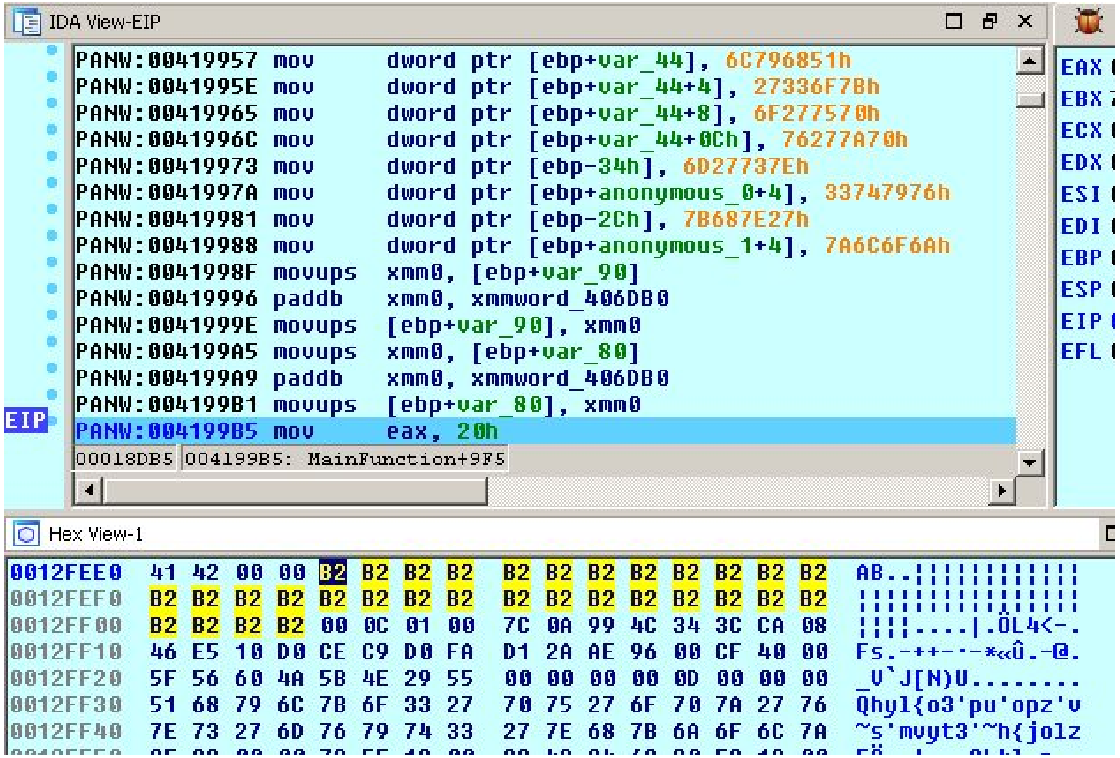
Function 6h & 7h



```
PANW:00419996 paddb   xmm0, xmmword_406DB0
PANW:0041999E movups  [ebp+var_90], xmm0
PANW:004199A5 movups  xmm0, [ebp+var_80]
PANW:004199A9 paddb   xmm0, xmmword_406DB0
PANW:004199B1 movups  [ebp+var_80], xmm0
PANW:004199B5 mov     eax, 20h
PANW:004199BA nop     word ptr [
PANW:004199C0
PANW:004199C0 loc_4199C0:                             ; CODE XREF: MainFunction+A0C↓j
PANW:004199CA add     byte ptr [ebp+eax+var_90], 7
```

```
xmmword_406DB0 xmmword 7070707070707070707070707070707h
                                          ; MainFunction+9
```

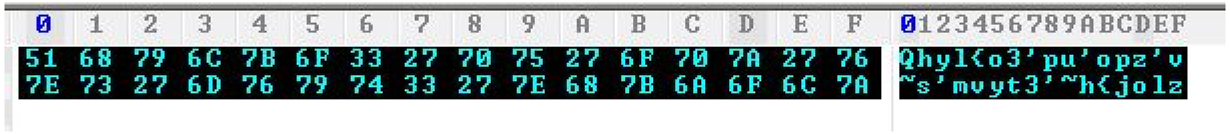
Function 6h & 7h



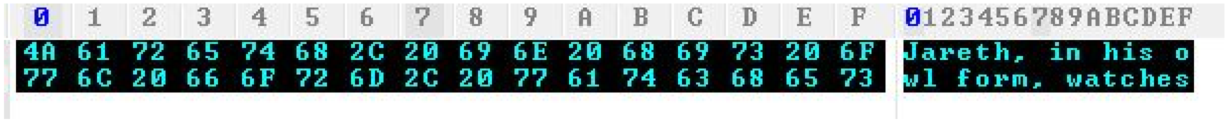
```
PANW:00419957 mov     dword ptr [ebp+var_44], 6C796851h
PANW:0041995E mov     dword ptr [ebp+var_44+4], 27336F7Bh
PANW:00419965 mov     dword ptr [ebp+var_44+8], 6F277570h
PANW:0041996C mov     dword ptr [ebp+var_44+0Ch], 76277A70h
PANW:00419973 mov     dword ptr [ebp-34h], 6D27737Eh
PANW:0041997A mov     dword ptr [ebp+anonymous_0+4], 33747976h
PANW:00419981 mov     dword ptr [ebp-2Ch], 7B687E27h
PANW:00419988 mov     dword ptr [ebp+anonymous_1+4], 7A6C6F6Ah
PANW:0041998F movups  xmm0, [ebp+var_90]
PANW:00419996 paddb   xmm0, xmmword_406DB0
PANW:0041999E movups  [ebp+var_90], xmm0
PANW:004199A5 movups  xmm0, [ebp+var_80]
PANW:004199A9 paddb   xmm0, xmmword_406DB0
PANW:004199B1 movups  [ebp+var_80], xmm0
PANW:004199B5 mov     eax, 20h
00018DB5 004199B5: MainFunction+9F5
```

```
0012FEE0  41 42 00 00 B2 B2 B2 B2  B2 B2 B2 B2 B2 B2 B2 B2  AB..||||||||||||
0012FEF0  B2 B2 B2 B2 B2 B2 B2 B2  B2 B2 B2 B2 B2 B2 B2 B2  ||||||||||||||||
0012FF00  B2 B2 B2 B2 00 0C 01 00  7C 0A 99 4C 34 3C CA 08  ||||....|.ÖL4<-.
0012FF10  46 E5 10 D0 CE C9 D0 FA  D1 2A AE 96 00 CF 40 00  Fs.-++-·-*«û.-@.
0012FF20  5F 56 60 4A 5B 4E 29 55  00 00 00 00 0D 00 00 00  _V`J[N)U........
0012FF30  51 68 79 6C 7B 6F 33 27  70 75 27 6F 70 7A 27 76  Qhyl{o3'pu'opz'v
0012FF40  7E 73 27 6D 76 79 74 33  27 7E 68 7B 6A 6F 6C 7A  ~s'mvyt3'~h{jolz
```

Function 6h & 7h

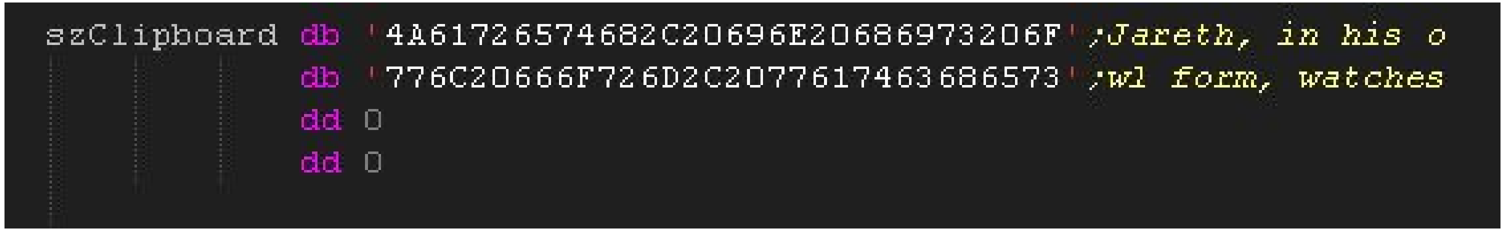


```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
51 68 79 6C 7B 6F 33 27 70 75 27 6F 70 7A 27 76  Qhyl{o3'pu'opz'v
7E 73 27 6D 76 79 74 33 27 7E 68 7B 6A 6F 6C 7A  ~s'mvyt3'~h{jolz
```



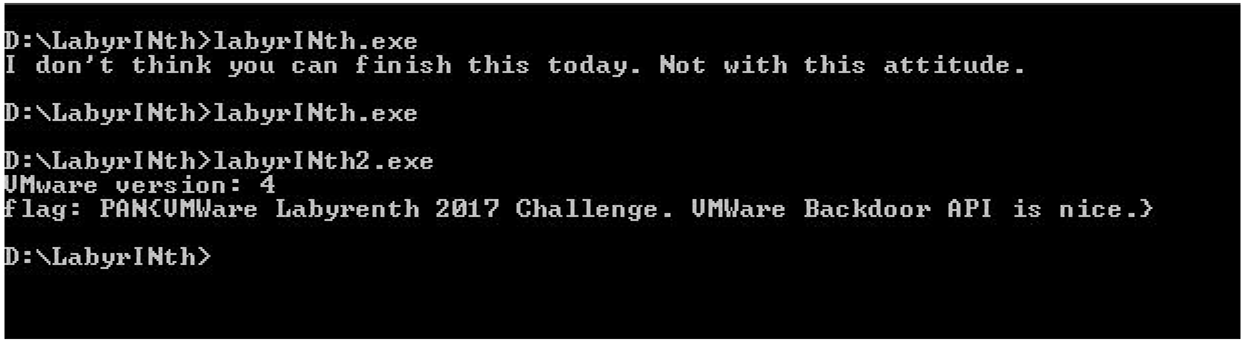
```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
4A 61 72 65 74 68 2C 20 69 6E 20 68 69 73 20 6F  Jareth, in his o
77 6C 20 66 6F 72 6D 2C 20 77 61 74 63 68 65 73  wl form, watches
```

Function 6h & 7h



```
szClipboard db '4A6172657468 2C20696E20686973206F' ;Jareth, in his o
            db '776C20666F726D2C207761746368 6573' ;wl form, watches
            dd 0
            dd 0
```

Function 6h & 7h



```
D:\LabyrINth>labyrINth.exe
I don't think you can finish this today. Not with this attitude.

D:\LabyrINth>labyrINth.exe

D:\LabyrINth>labyrINth2.exe
VMware version: 4
flag: PAN{VMWare Labyrenth 2017 Challenge. VMWare Backdoor API is nice.}

D:\LabyrINth>
```

Q/A

Questions?