CNIT 58100 CFM: CYBERFORENSICS OF MALWARE – LAB 6

Ibrahim Waziri Jr

PhD in Information Security (CERIAS)

Lab 6

Due on: September 24th, 2014

Instructor: **Associate Prof Sam Liles**

Purdue University

2014

Questions

Lab 6-1

In this lab, you will analyze the malware found in the file Lab06-01.exe

1. What is the major code construct found in the only subroutine called by main?
2. What is the subroutine located at 0x40105F?
3. What is the purpose of this program?

Lab 6-2

Analyze the malware found in the file Lab06-02.exe

1. What operation does the first subroutine called by main perform?
2. What is the subroutine located at 0x40117F?
3. What does the second subroutine called main do?
4. What type of code construct is used in this subroutine?
5. Are there any network-based indicators for this program?
6. What is the purpose of this malware?

Lab 6-3

In this lab, we'll analyze the malware found in the file Lab06-03.exe

1. Compare the calls in main to Lab6-2's main method. What is the new function called from main?
2. What parameters does this new function take?
3. What major code construct does this function contain?
4. What can this function do?
5. Are there any host-based indicators for this malware?
6. What is the purpose of this malware?

Lab 6-4

In this lab we will analyze the malware found in the file Lab06-04.exe

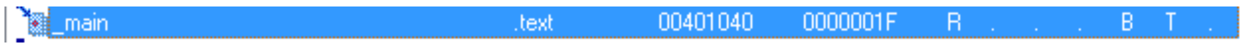
1. What is the difference between the calls made from the main method in Lab6-3 and 6-4?
2. What new construct has been added to main?
3. What is the difference between this lap's parse HTML function and those of the previous labs?
4. How long will this program run? (Assume that it is connected to the internet.)
5. Are there any new network-based indicators for this malware?
6. What is the purpose of this malware?

Answers

Lab 6-1

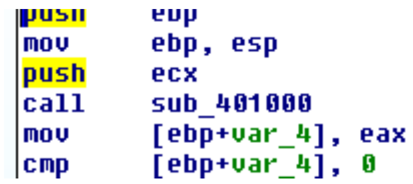
1. The main major code construct found in the only subroutine called by main is a call to *InternetGetConnectedState*

   We start by loading the Lab06-01.exe file into IDAPro and disassembling it. Next we navigate to the **function** tab and locating the *_main* function as shown in Fig 6-1A below:

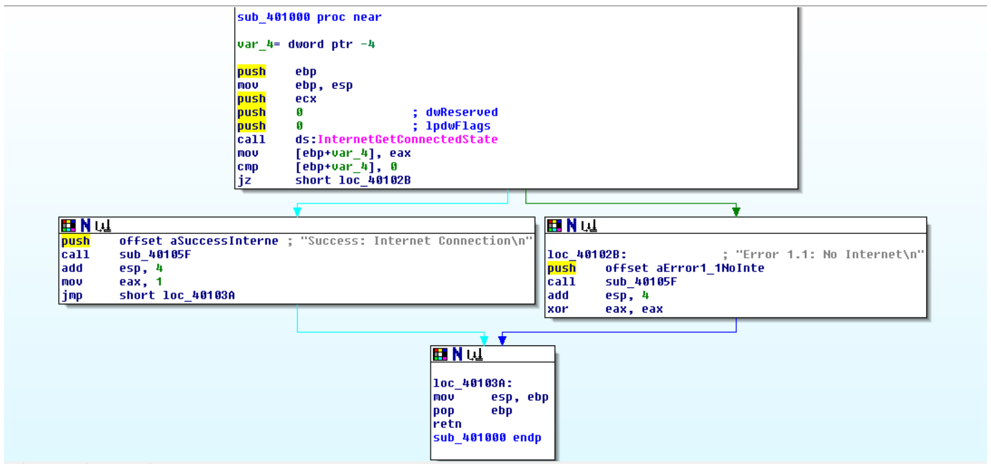


```
_main                    .text          00401040      0000001F      R  .  .  .  B  T  .
```

   Double clicking on that function shows that the *main* function calls the function at *sub_401000* as shown in Fig 6-1B below: Double clicking on the *call sub_401000* shown in Fig 6-1B below, we see two different calls, one is *InternetGetConnectedState* and the other is *sub_40105F.*



```
push    ebp
mov     ebp, esp
push    ecx
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
```



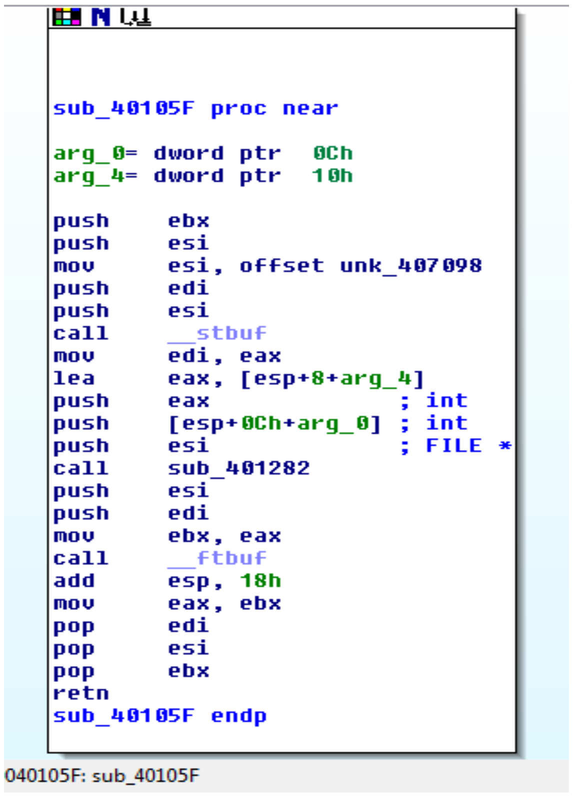
```
sub_401000 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40105F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:             ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40105F
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

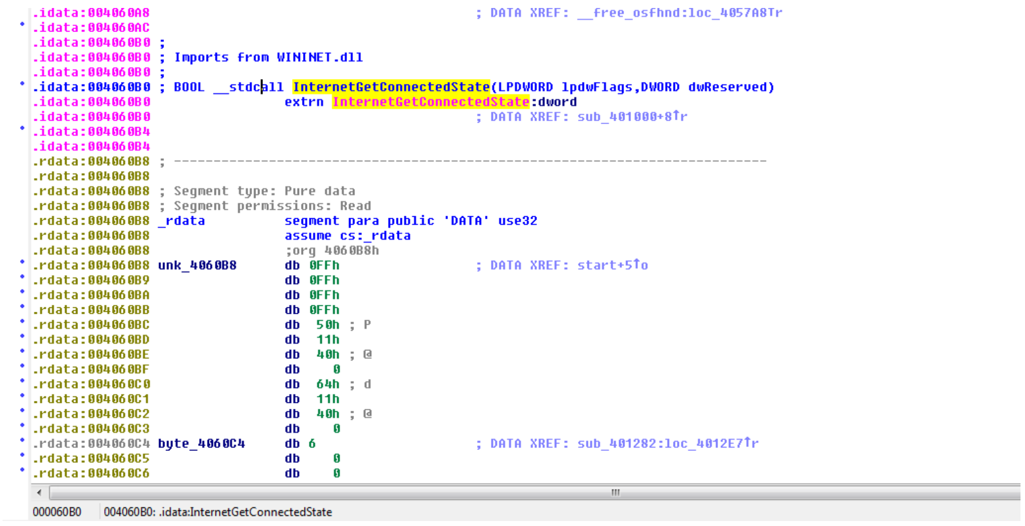
2. With reference to Figure 6-1-1C above. The subroutine located at 0x40105F is shown below:



```
sub_40105F proc near

arg_0= dword ptr  0Ch
arg_4= dword ptr  10h

push    ebx
push    esi
mov     esi, offset unk_407098
push    edi
push    esi
call    __stbuf
mov     edi, eax
lea     eax, [esp+8+arg_4]
push    eax                 ; int
push    [esp+0Ch+arg_0]     ; int
push    esi                 ; FILE *
call    sub_401282
push    esi
push    edi
mov     ebx, eax
call    __ftbuf
add     esp, 18h
mov     eax, ebx
pop     edi
pop     esi
pop     ebx
retn
sub_40105F endp
```

040105F: sub_40105F

3. The purpose of this program is to check for Internet Connectivity.
   This answer is concluded with respect to Fig 6-1-1C and carefully analyzing Fig 6-1-3 below.



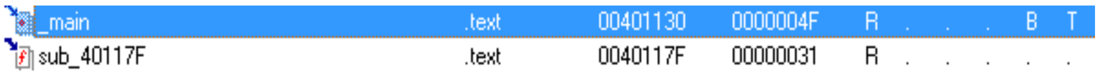
```
.idata:004060A8                                         ; DATA XREF: __free_osfhnd:loc_4057A8↑r
.idata:004060AC
.idata:004060B0 ;
.idata:004060B0 ; Imports from WININET.dll
.idata:004060B0 ;
.idata:004060B0 ; BOOL __stdcall InternetGetConnectedState(LPDWORD lpdwFlags,DWORD dwReserved)
.idata:004060B0                 extrn InternetGetConnectedState:dword
.idata:004060B0                                         ; DATA XREF: sub_401000+8↑r
.idata:004060B4
.idata:004060B4
.rdata:004060B8 ; ---------------------------------------------------------------------------
.rdata:004060B8
.rdata:004060B8 ; Segment type: Pure data
.rdata:004060B8 ; Segment permissions: Read
.rdata:004060B8 _rdata          segment para public 'DATA' use32
.rdata:004060B8                 assume cs:_rdata
.rdata:004060B8                 ;org 4060B8h
.rdata:004060B8 unk_4060B8      db 0FFh                 ; DATA XREF: start+5↑o
.rdata:004060B9                 db 0FFh
.rdata:004060BA                 db 0FFh
.rdata:004060BB                 db 0FFh
.rdata:004060BC                 db  50h ; P
.rdata:004060BD                 db  11h
.rdata:004060BE                 db  40h ; @
.rdata:004060BF                 db    0
.rdata:004060C0                 db  64h ; d
.rdata:004060C1                 db  11h
.rdata:004060C2                 db  40h ; @
.rdata:004060C3                 db    0
.rdata:004060C4 byte_4060C4     db 6                    ; DATA XREF: sub_401282:loc_4012E7↑r
.rdata:004060C5                 db    0
.rdata:004060C6                 db    0
```

000060B0 004060B0: .idata:InternetGetConnectedState

Lab 6-2

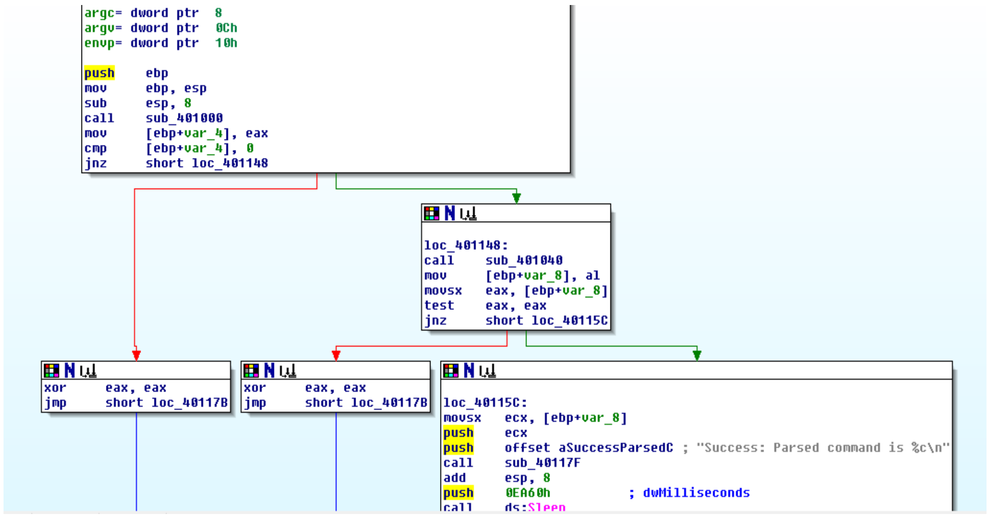
1. Like Lab6-1 the first subroutine called by *main* function is a call to *IntenetGetConnectedState*

   Loading and disassembling the Lab06-02.exe like we did with Lab06-01.exe. Navigating to functions tab and locating the _main function as shown in Fig 6-2-1A below:



```
_main         .text   00401130   0000004F   R . . . B T
sub_40117F    .text   0040117F   00000031   R . . . . .
```

   Double clicking on that function shows that the main function calls the same method at *sub_401000*. And also calls two other methods *sub_401040* and *su_40117F* as shown in Fig 6-2-1B below:



```
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 8
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401148
```

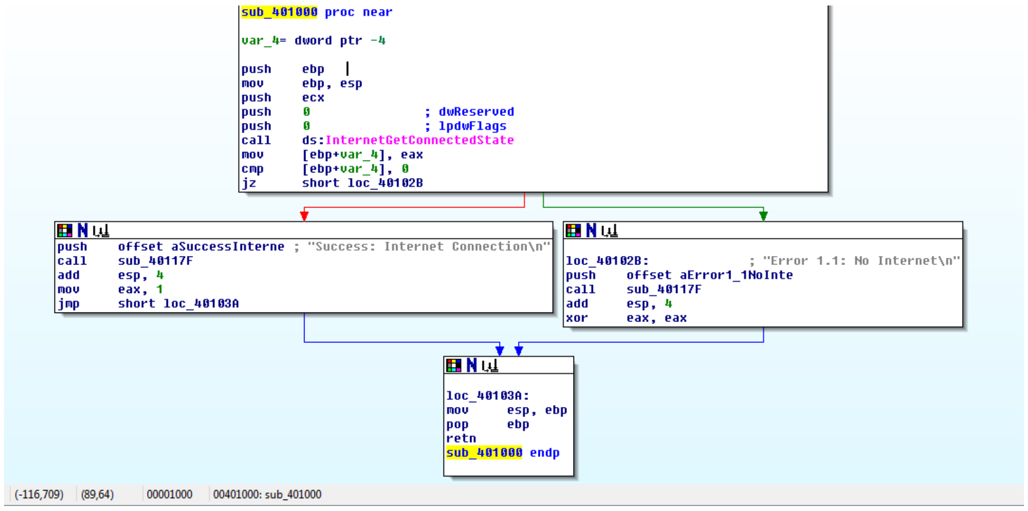
```
loc_401148:
call    sub_401040
mov     [ebp+var_8], al
movsx   eax, [ebp+var_8]
test    eax, eax
jnz     short loc_40115C
```

```
xor     eax, eax
jmp     short loc_40117B
```

```
xor     eax, eax
jmp     short loc_40117B
```

```
loc_40115C:
movsx   ecx, [ebp+var_8]
push    ecx
push    offset aSuccessParsedC ; "Success: Parsed command is %c\n"
call    sub_40117F
add     esp, 8
push    0EA60h          ; dwMilliseconds
call    ds:Sleep
```

   Double clicking on the *sub_401000* function shown in Fig 6-2-1B above shows the figure in Fig 6-2-1C. Clearly we can see that the first subroutine called by the main function is a call to *InternetGetConnectedState*



```
sub_401000 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

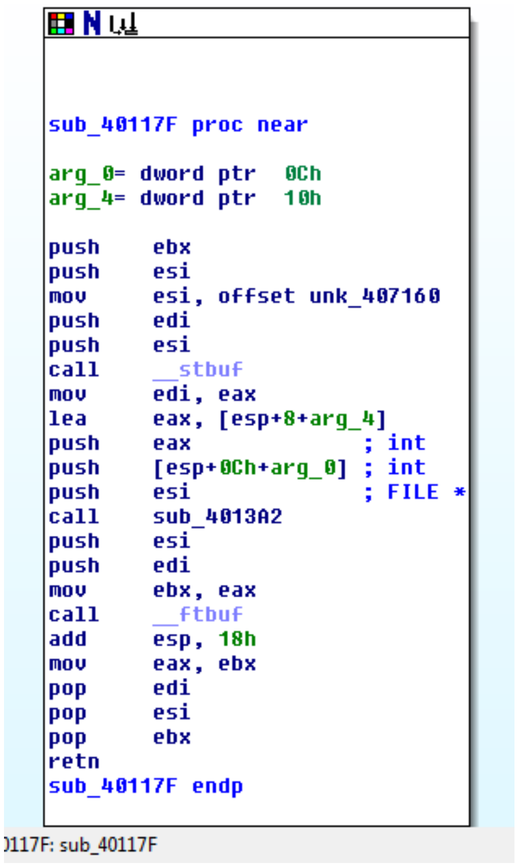
```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_40117F
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:             ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_40117F
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

```
(-116,709)  (89,64)  00001000  00401000: sub_401000
```

2. With reference to Fig 6-2-1C above, the subroutine located at 0x40117F is shown below:

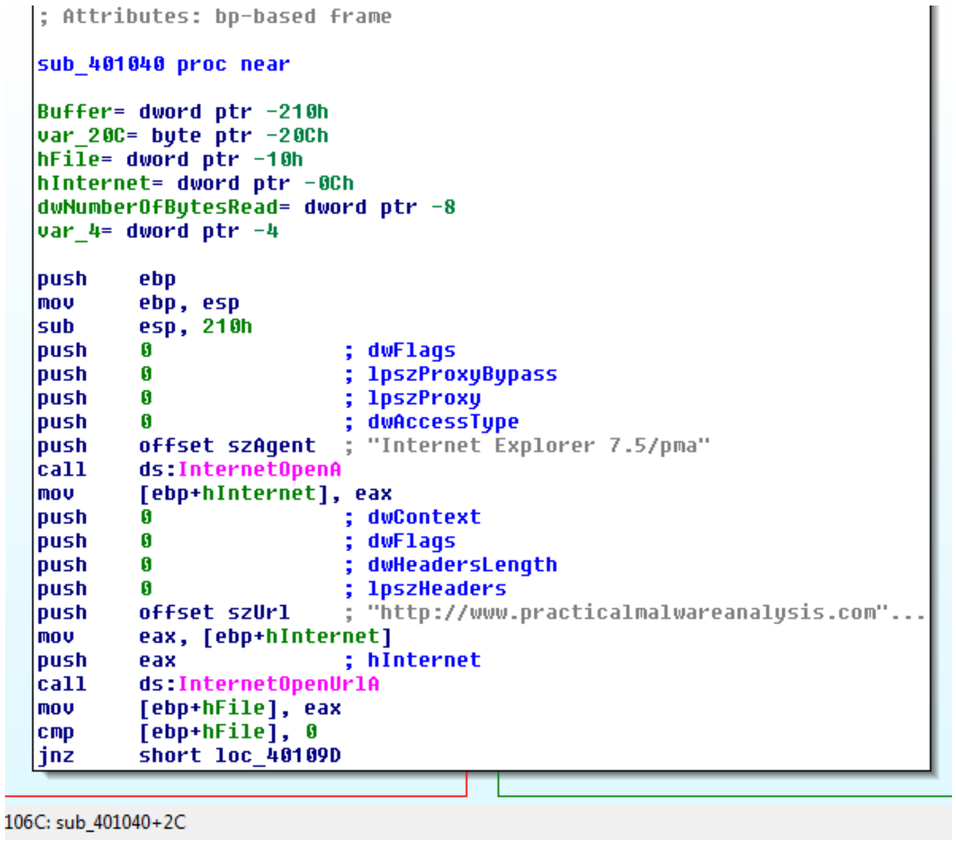


```
sub_40117F proc near

arg_0= dword ptr  0Ch
arg_4= dword ptr  10h

push    ebx
push    esi
mov     esi, offset unk_407160
push    edi
push    esi
call    __stbuf
mov     edi, eax
lea     eax, [esp+8+arg_4]
push    eax             ; int
push    [esp+0Ch+arg_0] ; int
push    esi             ; FILE *
call    sub_4013A2
push    esi
push    edi
mov     ebx, eax
call    __ftbuf
add     esp, 18h
mov     eax, ebx
pop     edi
pop     esi
pop     ebx
retn
sub_40117F endp
```

```
)117F: sub_40117F
```

3. From Figure 6-2-1C above, we can see that the second subroutine called by *main* is located at 0x401040. And what it does is download a HTML page located at

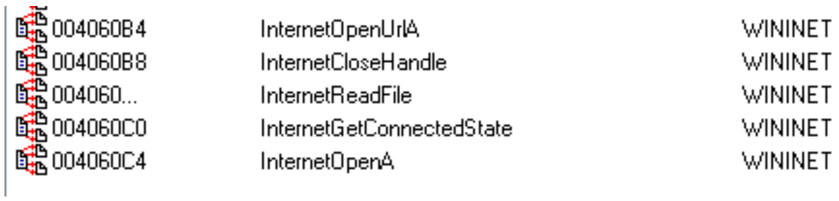
http://www.practicalmalwareanalysis.com/cc.htm . The second subroutine called by main is shown in Figure 6-2-3 below:



```
; Attributes: bp-based frame

sub_401040 proc near

Buffer= dword ptr -210h
var_20C= byte ptr -20Ch
hFile= dword ptr -10h
hInternet= dword ptr -0Ch
dwNumberOfBytesRead= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 210h
push    0               ; dwFlags
push    0               ; lpszProxyBypass
push    0               ; lpszProxy
push    0               ; dwAccessType
push    offset szAgent  ; "Internet Explorer 7.5/pma"
call    ds:InternetOpenA
mov     [ebp+hInternet], eax
push    0               ; dwContext
push    0               ; dwFlags
push    0               ; dwHeadersLength
push    0               ; lpszHeaders
push    offset szUrl    ; "http://www.practicalmalwareanalysis.com"...
mov     eax, [ebp+hInternet]
push    eax             ; hInternet
call    ds:InternetOpenUrlA
mov     [ebp+hFile], eax
cmp     [ebp+hFile], 0
jnz     short loc_40109D
```

106C: sub_401040+2C

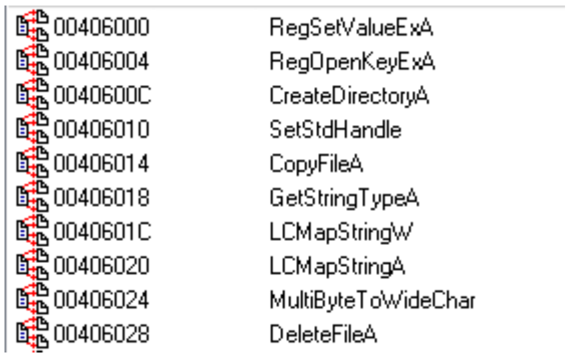
4. The code constructs used in this subroutine are call functions for networking. These functions can be seen by viewing the imports of the subroutine.



004060B4	InternetOpenUrlA	WININET
004060B8	InternetCloseHandle	WININET
004060...	InternetReadFile	WININET
004060C0	InternetGetConnectedState	WININET
004060C4	InternetOpenA	WININET

As shown in Figure 6-2-4 above. To view the imports we can click on the imports tab, are view the imports that are a function part of WININET. These imports are simple API for using HTTP over a network. These imports are as follows:

- InternetOpenUrlA
- InternetCloseHandle
- InternetReadFile
- InternetGetConnectedState
- InternetOpenA



```
00406000    RegSetValueExA
00406004    RegOpenKeyExA
0040600C    CreateDirectoryA
00406010    SetStdHandle
00406014    CopyFileA
00406018    GetStringTypeA
0040601C    LCMapStringW
00406020    LCMapStringA
00406024    MultiByteToWideChar
00406028    DeleteFileA
```

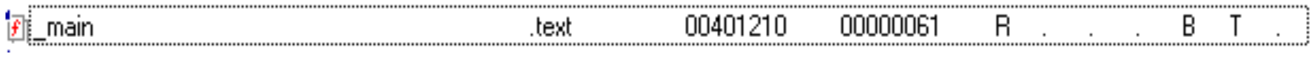
5. Yes, the HTML web page at http://www.practicalmalwareanalysis.com/cc.htm form Figure 6-2-1C above. This page can be used as a network-based indicator.

6. From the imports shown in Figure 6-2-4 above. One can conclude that the program checks for an active Internet connection.

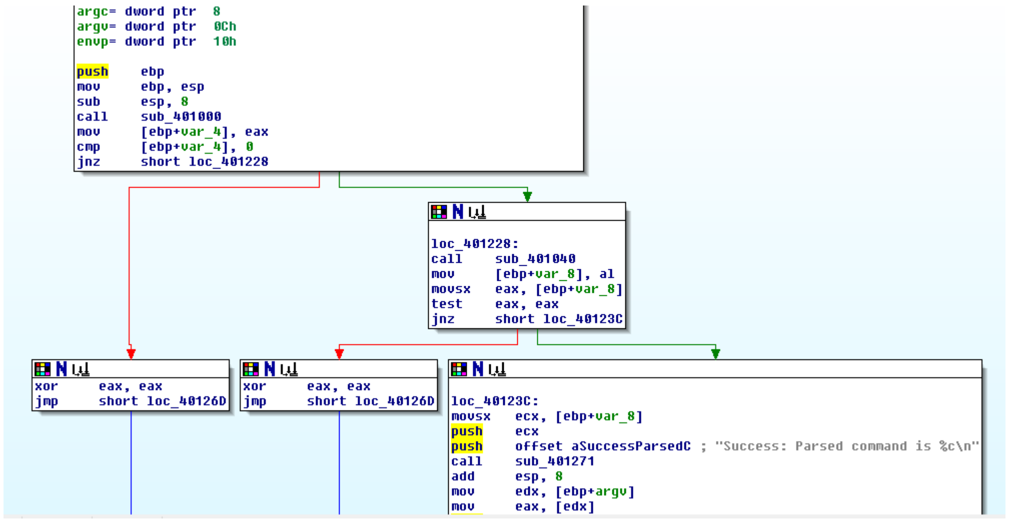
Lab 6-3

1. Beside from the fact that the main in Lab6-2 is 0x401130 and that of this Lab is 0x401210. Everything is the same as shown in Figure below

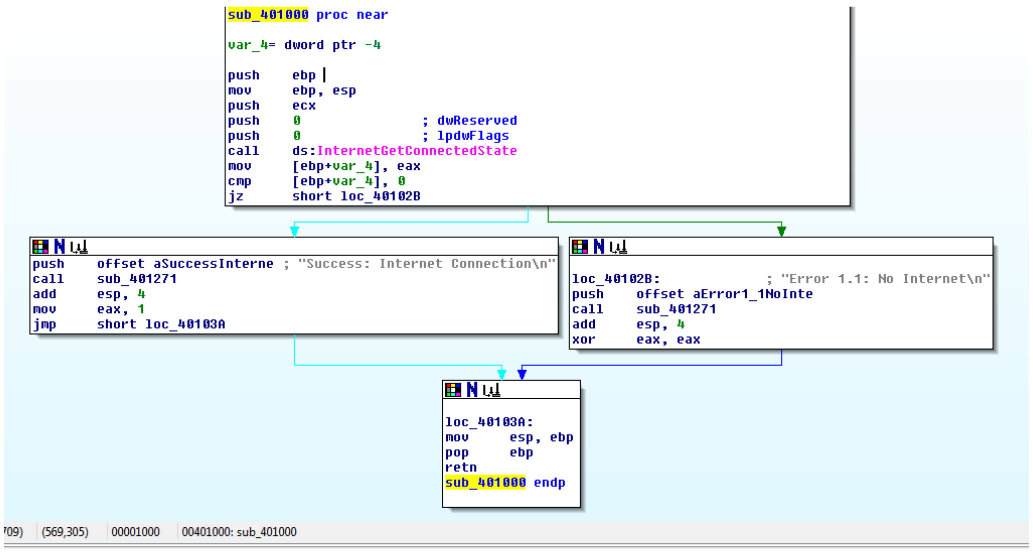
   From Figure 6-3-1D below, we can see that there is an extra call to 0x401130. That is the new function called from main.



```
_main    .text    00401210    00000061    R . . . B T .
```



```
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 8
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401228

loc_401228:
call    sub_401040
mov     [ebp+var_8], al
movsx   eax, [ebp+var_8]
test    eax, eax
jnz     short loc_40123C

xor     eax, eax
jmp     short loc_40126D

xor     eax, eax
jmp     short loc_40126D

loc_40123C:
movsx   ecx, [ebp+var_8]
push    ecx
push    offset aSuccessParsedC ; "Success: Parsed command is %c\n"
call    sub_401271
add     esp, 8
mov     edx, [ebp+argv]
mov     eax, [edx]
```



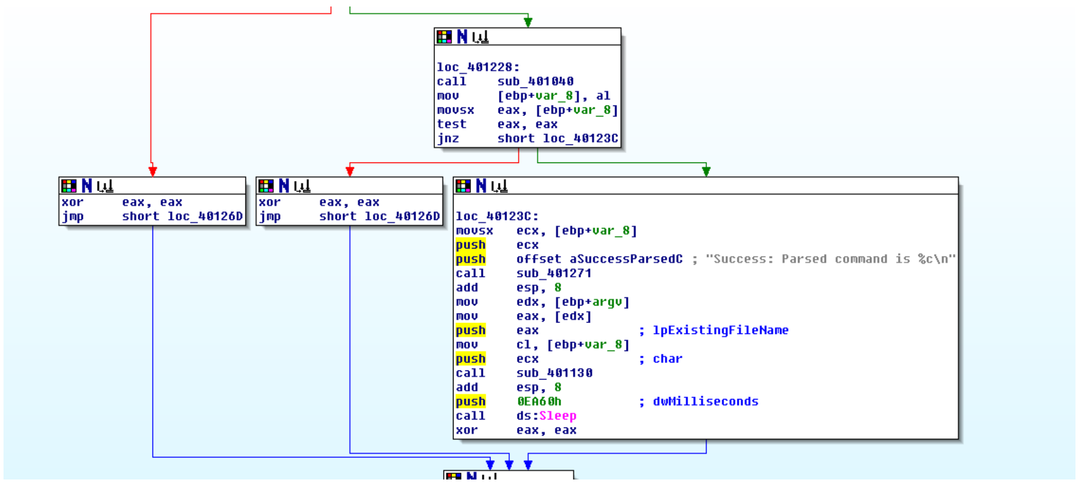
```
sub_401000 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    0               ; dwReserved
push    0               ; lpdwFlags
call    ds:InternetGetConnectedState
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

```
push    offset aSuccessInterne ; "Success: Internet Connection\n"
call    sub_401271
add     esp, 4
mov     eax, 1
jmp     short loc_40103A
```

```
loc_40102B:             ; "Error 1.1: No Internet\n"
push    offset aError1_1NoInte
call    sub_401271
add     esp, 4
xor     eax, eax
```

```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

```
709)  (569,305)  00001000  00401000: sub_401000
```



```
loc_401228:
call    sub_401040
mov     [ebp+var_8], al
movsx   eax, [ebp+var_8]
test    eax, eax
jnz     short loc_40123C
```

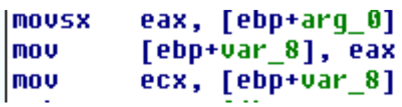
```
xor     eax, eax
jmp     short loc_40126D
```

```
xor     eax, eax
jmp     short loc_40126D
```

```
loc_40123C:
movsx   ecx, [ebp+var_8]
push    ecx
push    offset aSuccessParsedC ; "Success: Parsed command is %c\n"
call    sub_401271
add     esp, 8
mov     edx, [ebp+argv]
mov     eax, [edx]
push    eax             ; lpExistingFileName
mov     cl, [ebp+var_8]
push    ecx             ; char
call    sub_401130
add     esp, 8
push    0EA60h          ; dwMilliseconds
call    ds:Sleep
xor     eax, eax
```

2. Examining the parameters passed to 0x401130 which is the new function shown in Figure 6-3-1D, we can see that the new function takes the movsx arg 0 and mov +var 8 parameters. As shown in Figure 6-3-2 below:

The figure below can be viewed by double-clicking the calls 0x401130 as shown in Fig 6-3-1D above.



```
movsx   eax, [ebp+arg_0]
mov     [ebp+var_8], eax
mov     ecx, [ebp+var_8]
```