

**S11 / L2**

**ESERCITAZIONE**

# TRACCIA

01

Individuare l'indirizzo della funzione DLLMain (così com'è, in esadecimale)

02

Dalla scheda «imports» individuare la funzione «gethostbyname ». Qual è l'indirizzo dell'import? Cosa fa la funzione?

03

Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?

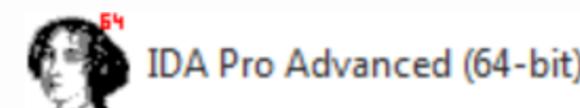
04

Quanti sono, invece, i parametri della funzione sopra?

05

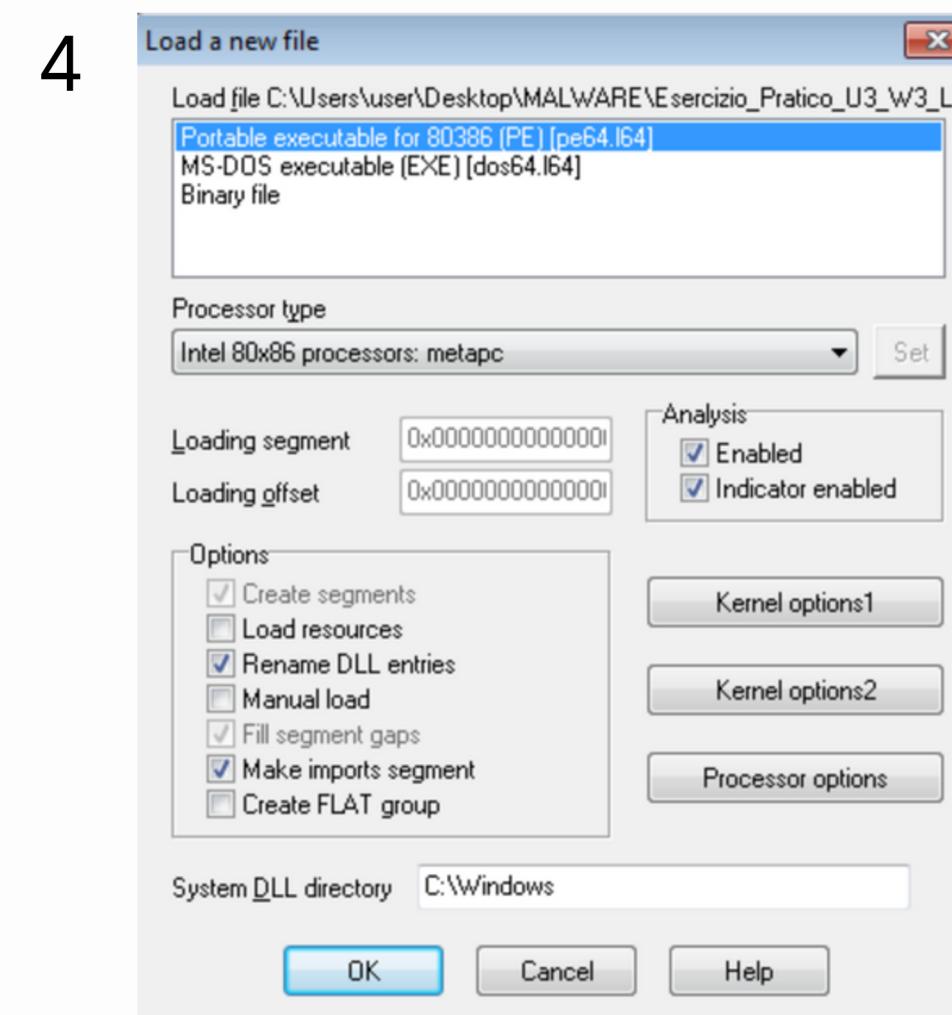
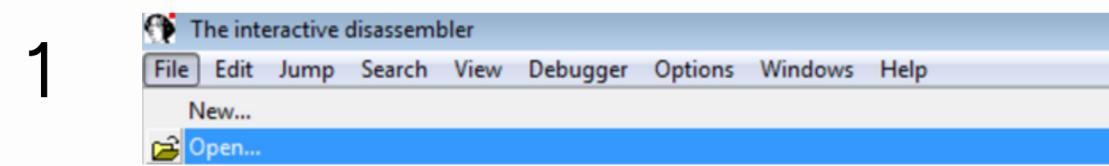
Inserire altre considerazioni macro livello sul malware (comportamento)

# AVVIO IDA

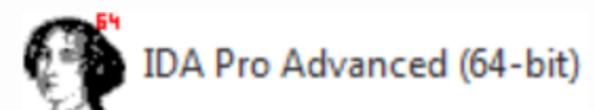


Il software che andremo ad utilizzare è **IDA** ( Interactive Disassembler )  
uno strumento che traduce il codice macchina in codice assembly leggibile dall'uomo.

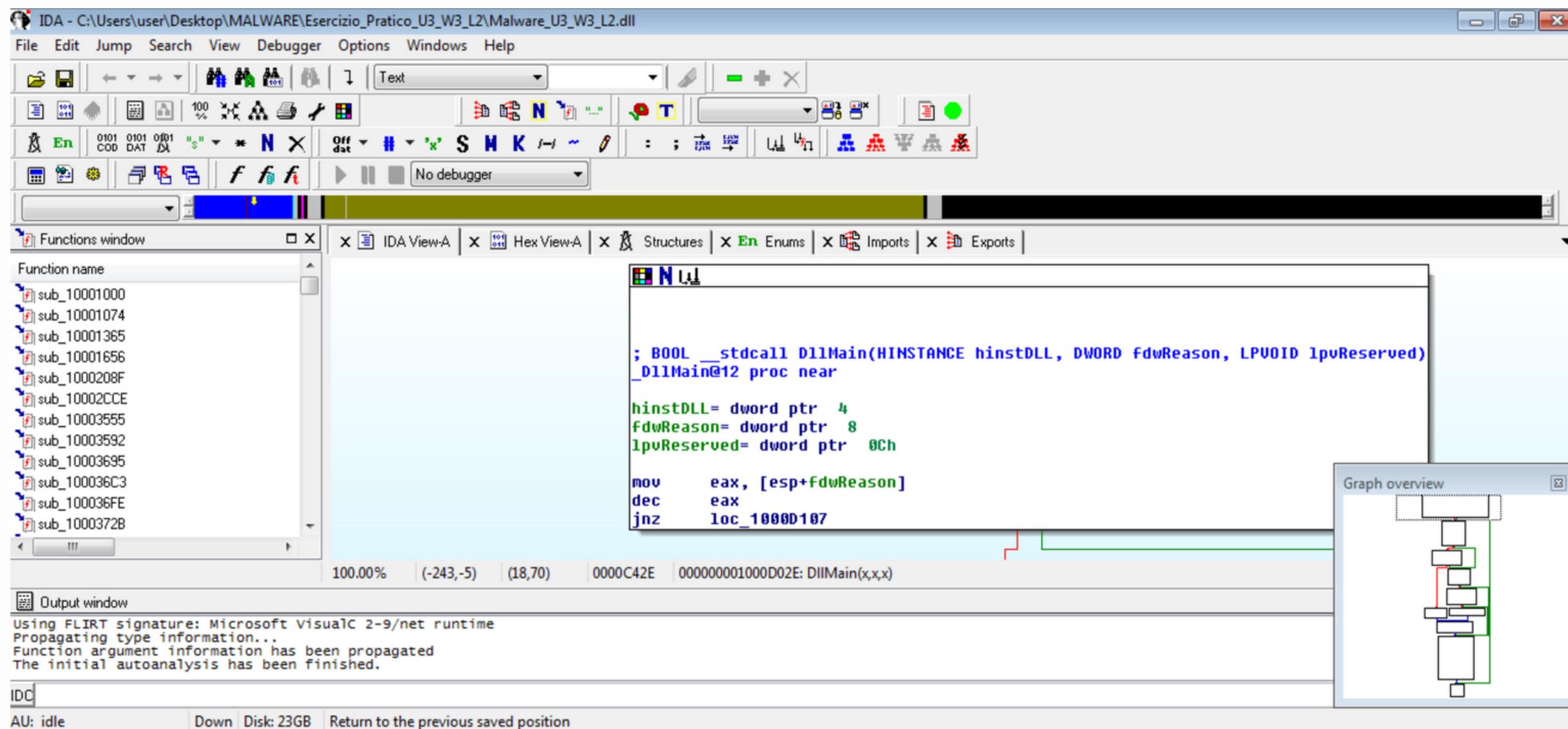
Eseguito il tool andiamo ad aprire all'interno il malware in questione: **Malware\_U3\_W3\_L2**



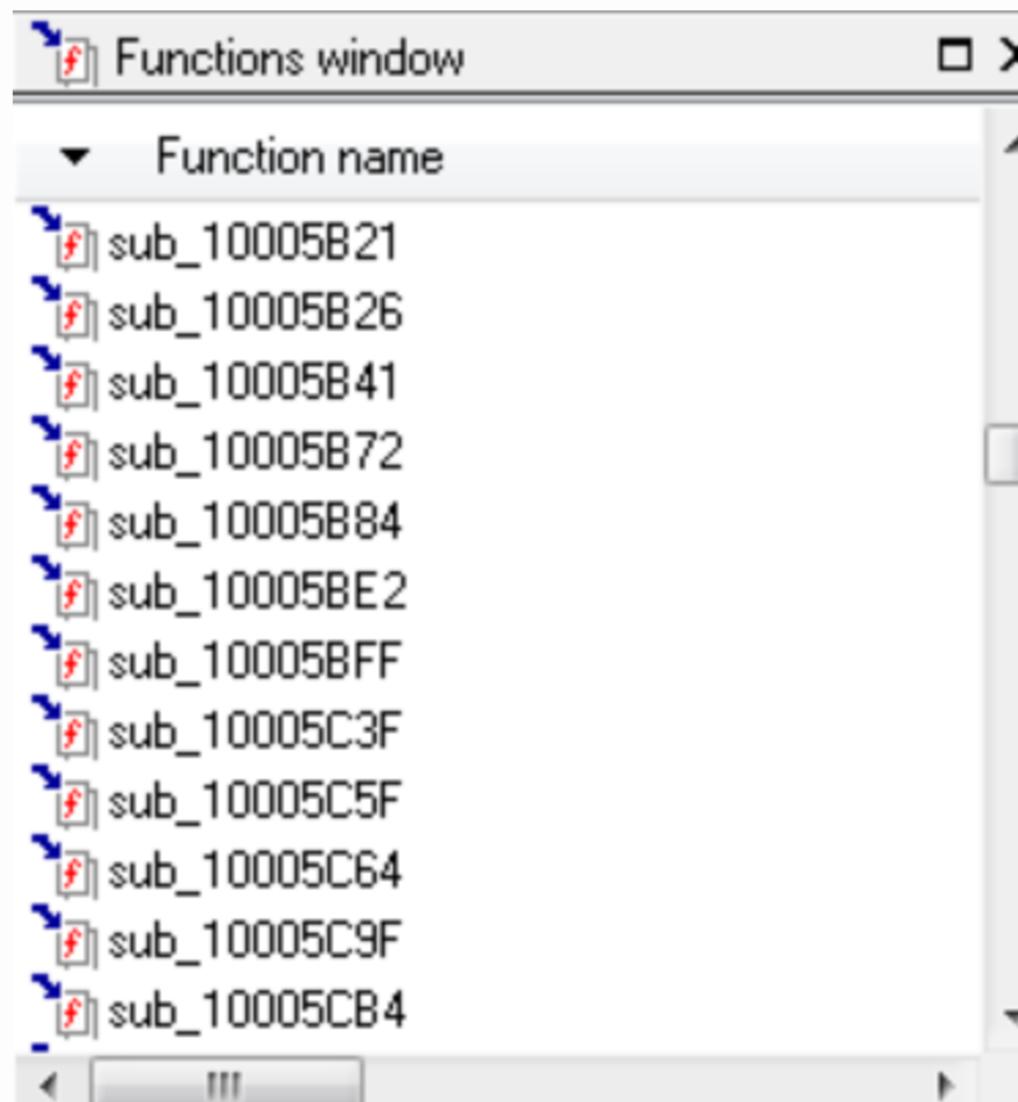
# AVVIO IDA



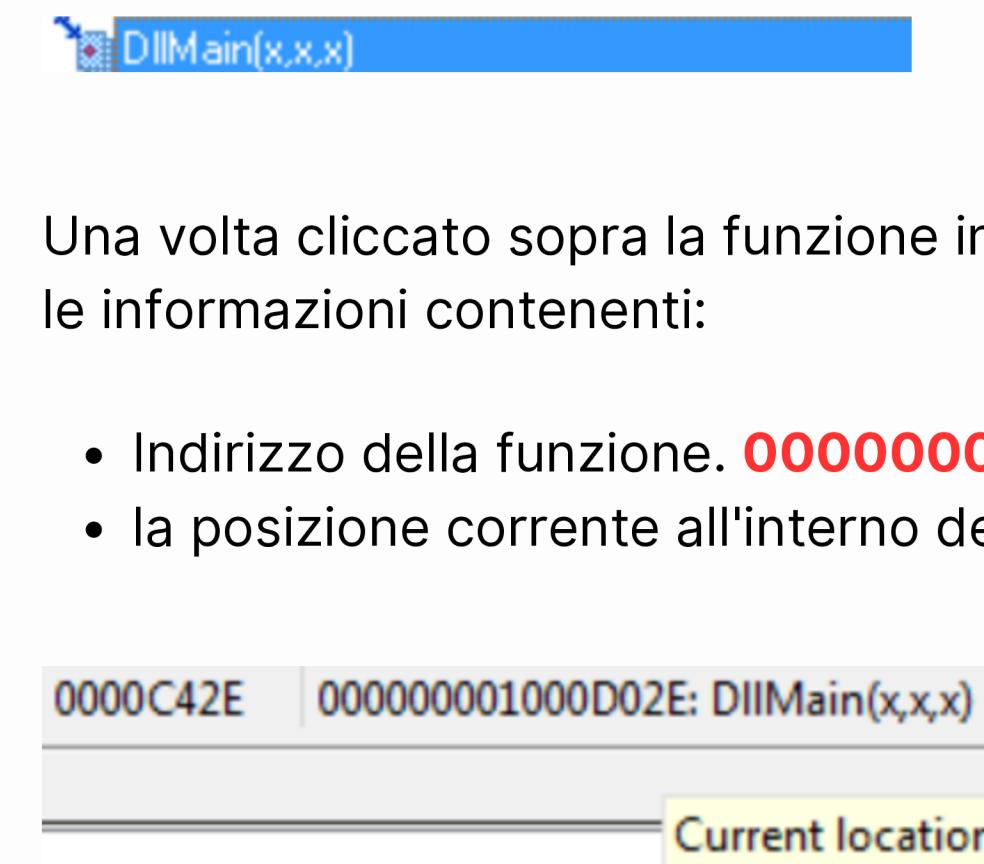
Da qui si aprirà una finestra, dove possiamo iniziare ad analizzare l'eseguibile.



# INDIVIDUARE L'INDIRIZZO DELLA FUNZIONE DLLMAIN (COSÌ COM'È, IN ESADECIMALE)



Per andare a controllare l'indirizzo della funzione DLL Main andiamo a far una ricerca nella sezione Funzioni situati nella colonna a sinistra.



Una volta cliccato sopra la funzione in questione appariranno le informazioni contenenti:

- Indirizzo della funzione. **000000001000D02E**
- la posizione corrente all'interno del file **0000C42E**

# DALLA SCHEDA «IMPORTS» INDIVIDUARE LA FUNZIONE «GETHOSTBYNAME». QUAL È L'INDIRIZZO DELL'IMPORT? COSA FA LA FUNZIONE?

## DEFINIZIONE

La funzione gethostbyname restituisce un puntatore a una struttura hostent, una struttura allocata da Windows Sockets. Esso converte il nome dei domini in indirizzi ipv4

Possiamo andare a cercare la funzione Gethostbyname nella sezione Imports.

000000...	fwrite	MSVCRT
000000...	52 gethostbyname	WS2_32
000000...	9 htons	WS2_32

doppio click e ci aprirà una finestra con le diverse informazioni sulla funzione

000000...	SystemParametersInfoA	USER32
000000...	SystemTimeToFileTime	KERNEL32
000000...	TerminateProcess	KERNEL32
000000...	TerminateThread	KERNEL32
000000...	Thread32First	KERNEL32
000000...	Thread32Next	KERNEL32
000000...	VariantClear	OLEAUT32
9 000000...	VirtualAllocEx	KERNEL32
000000...	VirtualQuery	KERNEL32

```
.idata:100163CC ; struct hostent * __stdcall gethostbyname(const char *name)
idata:100163CC      extrn gethostbyname:dword
idata:100163CC      ; CODE XREF: sub_10001074:loc_
idata:100163CC      ; sub_10001074+1D3↑D ...
```

Indirizzo funzione: **1000163CC**

# QUANTE SONO LE VARIABILI LOCALI DELLA FUNZIONE ALLA LOCAZIONE DI MEMORIA 0X10001656?

Per ricercare la locazione di memoria interessata andiamo nella sezione filtro e inseriamo **0X10001656**

Address ▼ 0x10001656 ▼

Diamo **OK**

e avremo la rappresentazione in assembly della Funzione.

```
var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= duord ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Source= byte ptr -63Dh
Data= byte ptr -638h
var_637= byte ptr -637h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
Buf2= byte ptr -4FCh
readFds= fd_set ptr -48Ch
phkResult= byte ptr -388h
var_3B0= dword ptr -380h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
arg_0= duord ptr -4
```

Le variabili individuate per la funzione sono:

**24**

## QUANTI SONO, INVECE, I PARAMETRI DELLA FUNZIONE SOPRA?

L'unico parametro individuato in questa funzione è

**arg\_0 dword ptr 4**

```
var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= duord ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Source= byte ptr -63Dh
Data= byte ptr -638h
var_637= byte ptr -637h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
Buf2= byte ptr -4FCCh
readFds= fd_set ptr -48Ch
phkResult= byte ptr -388h
var_380= dword ptr -380h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSADATA ptr -190h
arg_0= dword ptr 4
```

## INSERIRE ALTRE CONSIDERAZIONI MACRO LIVELLO SUL MALWARE (COMPORTAMENTO)

### sub 1000355

con la chiamata **LoadLibraryA**  
carica librerie con diverse funzioni

```
sub_10003555 proc near
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    ecx
push    ecx
push    offset aUser32_dll_0 ; "user32.dll"
call    ds:LoadLibraryA
push    offset aGetlastinputin ; "GetLastInputInfo"
push    eax      ; hModule
call    ds:GetProcAddress
lea     ecx, [ebp+var_8]
mov     [ebp+var_8], 8
push    ecx
call    eax
call    ds:GetTickCount
sub    eax, [ebp+var_4]
xor    edx, edx
mov     ecx, 3E8h
div    ecx
leave
retn
```

### sub 10003695

con la chiamata **GetVersionExA**  
La funzione recupera le informazioni sulla versione del sistema operativo.



```
; Attributes: bp-based frame
sub_10003695 proc near
VersionInformation= _OSVERSIONINFOA ptr -94h
push    ebp
mov     ebp, esp
sub    esp, 94h
lea     eax, [ebp+VersionInformation]
mov     [ebp+VersionInformation.dwOSVersionInfoSize], 94h
push    eax      ; lpVersionInformation
call    ds:GetVersionExA
xor    eax, eax
cmp    [ebp+VersionInformation.dwPlatformId], 2
setz    al
leave
retn
sub_10003695 endp
```

## INSERIRE ALTRE CONSIDERAZIONI MACRO LIVELLO SUL MALWARE (COMPORTAMENTO)

In base alle analisi generali con il tool IDA possiamo aver notato che il Malware ha caricato diverse librerie e attraverso vari processi ha utilizzato diverse funzioni:

- **RegOpenkeyexA**
- **RegQueryValue**
- **GetVersionExA**
- **GetProcAddress**
- **CreateProcessAsUserA**

Funzioni che possono essere utilizzati per andare a ottenere informazioni sul sistema infettato.

