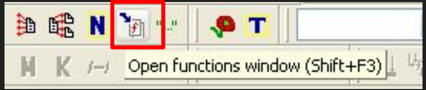
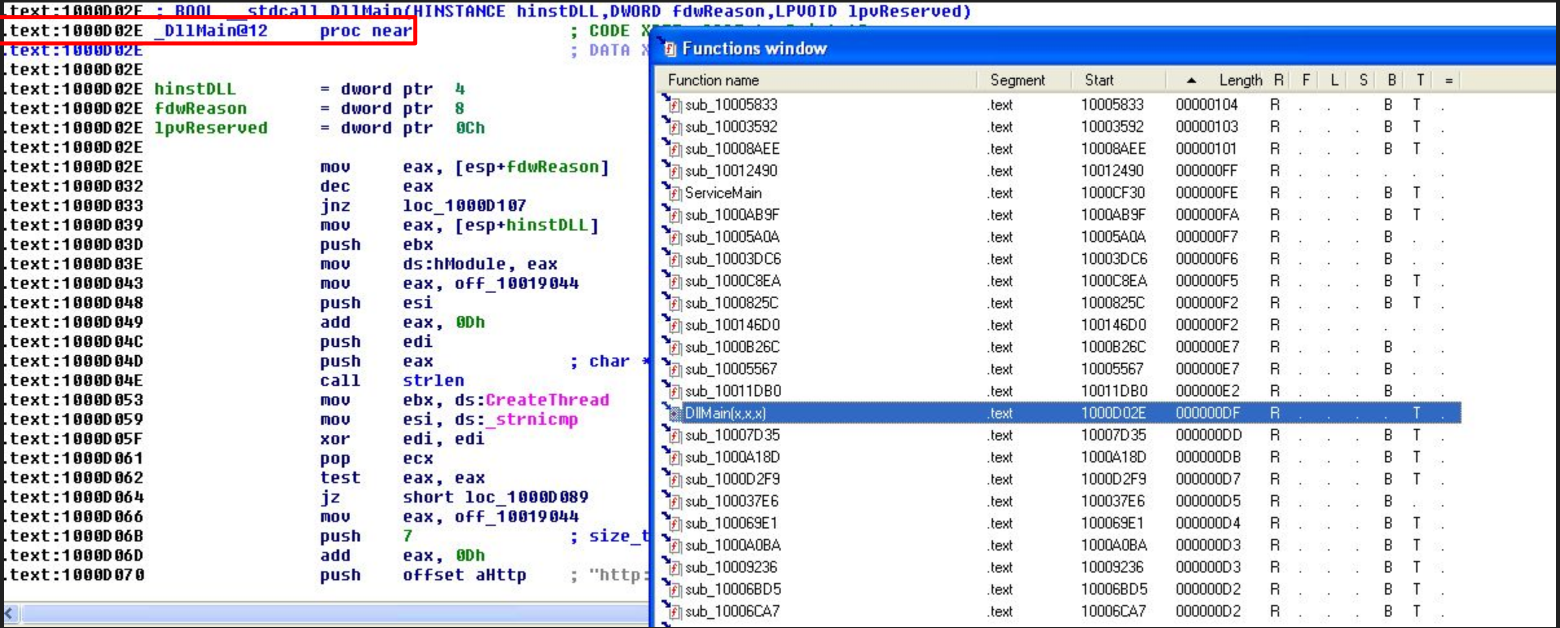
STATIC MALWARE ANALYSIS: IDA PRO

S11 - L2

Static Malware Analysis: IDA PRO
S11 - L2





```
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpReserved)
.text:1000D02E _DllMain@12     proc near               ; CODE X
.text:1000D02E                                         ; DATA X
.text:1000D02E
.text:1000D02E hinstDLL        = dword ptr  4
.text:1000D02E fdwReason       = dword ptr  8
.text:1000D02E lpvReserved     = dword ptr  0Ch
.text:1000D02E
.text:1000D02E                 mov     eax, [esp+fdwReason]
.text:1000D032                 dec     eax
.text:1000D033                 jnz     loc_1000D107
.text:1000D039                 mov     eax, [esp+hinstDLL]
.text:1000D03D                 push    ebx
.text:1000D03E                 mov     ds:hModule, eax
.text:1000D043                 mov     eax, off_10019044
.text:1000D048                 push    esi
.text:1000D049                 add     eax, 0Dh
.text:1000D04C                 push    edi
.text:1000D04D                 push    eax             ; char *
.text:1000D04E                 call    strlen
.text:1000D053                 mov     ebx, ds:CreateThread
.text:1000D059                 mov     esi, ds:_strnicmp
.text:1000D05F                 xor     edi, edi
.text:1000D061                 pop     ecx
.text:1000D062                 test    eax, eax
.text:1000D064                 jz      short loc_1000D089
.text:1000D066                 mov     eax, off_10019044
.text:1000D06B                 push    7               ; size_t
.text:1000D06D                 add     eax, 0Dh
.text:1000D070                 push    offset aHttp    ; "http:
```

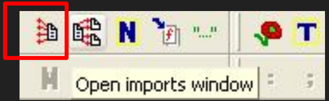
Functions window

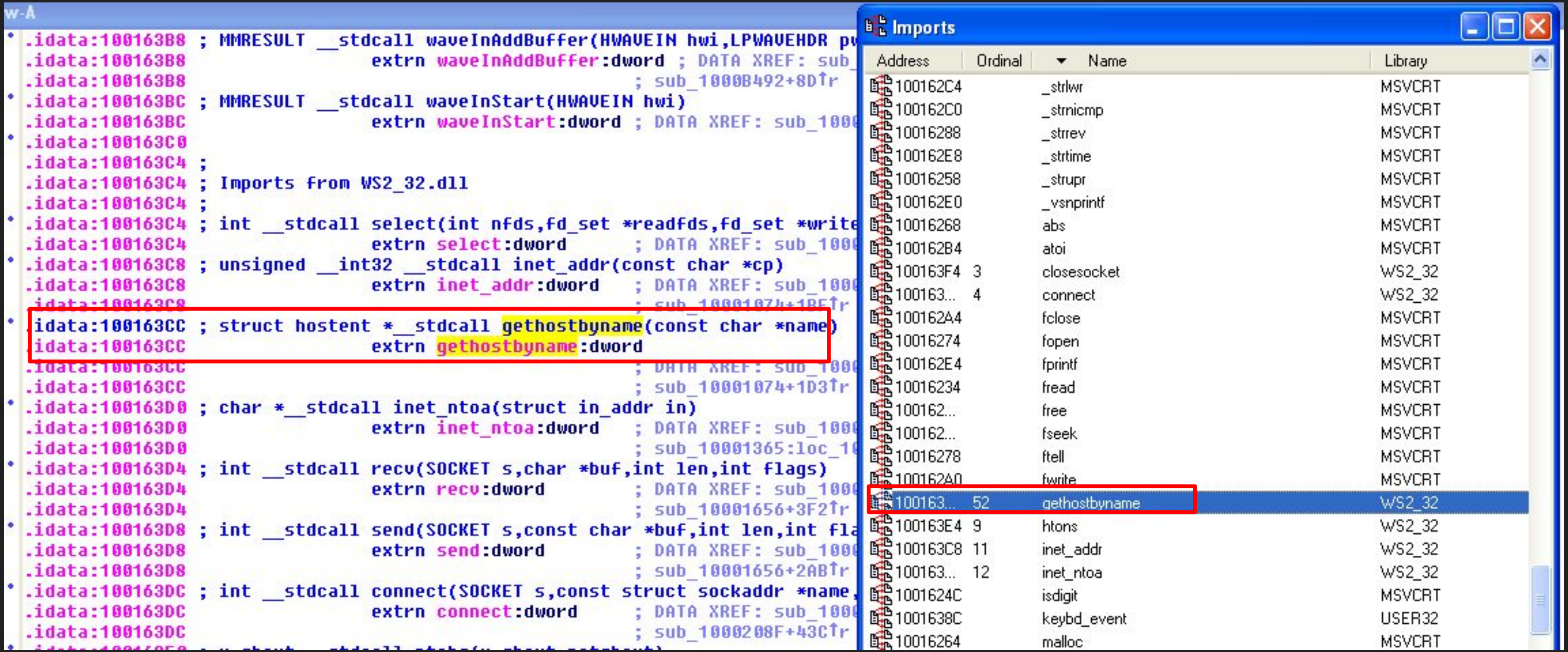
Function name	Segment	Start	Length	R	F	L	S	B	T	=
sub_10005833	.text	10005833	00000104	R	.	.	.	B	T	.
sub_10003592	.text	10003592	00000103	R	.	.	.	B	T	.
sub_10008AEE	.text	10008AEE	00000101	R	.	.	.	B	T	.
sub_10012490	.text	10012490	000000FF	R	.	.	.	.	.	.
ServiceMain	.text	1000CF30	000000FE	R	.	.	.	B	T	.
sub_1000AB9F	.text	1000AB9F	000000FA	R	.	.	.	B	T	.
sub_10005A0A	.text	10005A0A	000000F7	R	.	.	.	B	.	.
sub_10003DC6	.text	10003DC6	000000F6	R	.	.	.	B	.	.
sub_1000C8EA	.text	1000C8EA	000000F5	R	.	.	.	B	T	.
sub_1000825C	.text	1000825C	000000F2	R	.	.	.	B	T	.
sub_100146D0	.text	100146D0	000000F2	R	.	.	.	.	.	.
sub_1000B26C	.text	1000B26C	000000E7	R	.	.	.	B	.	.
sub_10005567	.text	10005567	000000E7	R	.	.	.	B	.	.
sub_10011DB0	.text	10011DB0	000000E2	R	.	.	.	B	.	.
DllMain(x,x,x)	.text	1000D02E	000000DF	R	.	.	.	.	T	.
sub_10007D35	.text	10007D35	000000DD	R	.	.	.	B	T	.
sub_1000A18D	.text	1000A18D	000000DB	R	.	.	.	B	T	.
sub_1000D2F9	.text	1000D2F9	000000D7	R	.	.	.	B	T	.
sub_100037E6	.text	100037E6	000000D5	R	.	.	.	B	.	.
sub_100069E1	.text	100069E1	000000D4	R	.	.	.	B	T	.
sub_1000A0BA	.text	1000A0BA	000000D3	R	.	.	.	B	T	.
sub_10009236	.text	10009236	000000D3	R	.	.	.	B	T	.
sub_10006BD5	.text	10006BD5	000000D2	R	.	.	.	B	T	.
sub_10006CA7	.text	10006CA7	000000D2	R	.	.	.	B	T	.

Attraverso la GUI possiamo aprire la "Functions Window" che contiene tutte le funzioni utilizzate nel codice. Cercandolo possiamo trovare la funzione DLLMain all'indirizzo 1000D0E.

Static Malware Analysis: IDA PRO

S11 - L2





```
.idata:100163B8 ; MMRESULT __stdcall waveInAddBuffer(HWAVEIN hwi,LPWAVEHDR pw
.idata:100163B8                 extrn waveInAddBuffer:dword ; DATA XREF: sub_
.idata:100163B8                                         ; sub_1000B492+8D↑r
.idata:100163BC ; MMRESULT __stdcall waveInStart(HWAVEIN hwi)
.idata:100163BC                 extrn waveInStart:dword ; DATA XREF: sub_1000
.idata:100163C0
.idata:100163C4 ;
.idata:100163C4 ; Imports from WS2_32.dll
.idata:100163C4 ;
.idata:100163C4 ; int __stdcall select(int nfds,fd_set *readfds,fd_set *write
.idata:100163C4                 extrn select:dword      ; DATA XREF: sub_1000
.idata:100163C8 ; unsigned __int32 __stdcall inet_addr(const char *cp)
.idata:100163C8                 extrn inet_addr:dword   ; DATA XREF: sub_1000
.idata:100163C8                                         ; sub_10001074+1BF↑r
.idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC                 extrn gethostbyname:dword
.idata:100163CC                                         ; DATA XREF: sub_1000
.idata:100163CC                                         ; sub_10001074+1D3↑r
.idata:100163D0 ; char *__stdcall inet_ntoa(struct in_addr in)
.idata:100163D0                 extrn inet_ntoa:dword   ; DATA XREF: sub_1000
.idata:100163D0                                         ; sub_10001365:loc_1
.idata:100163D4 ; int __stdcall recv(SOCKET s,char *buf,int len,int flags)
.idata:100163D4                 extrn recv:dword        ; DATA XREF: sub_1000
.idata:100163D4                                         ; sub_10001656+3F2↑r
.idata:100163D8 ; int __stdcall send(SOCKET s,const char *buf,int len,int fla
.idata:100163D8                 extrn send:dword        ; DATA XREF: sub_1000
.idata:100163D8                                         ; sub_10001656+2AB↑r
.idata:100163DC ; int __stdcall connect(SOCKET s,const struct sockaddr *name,
.idata:100163DC                 extrn connect:dword     ; DATA XREF: sub_1000
.idata:100163DC                                         ; sub_1000208F+43C↑r
```

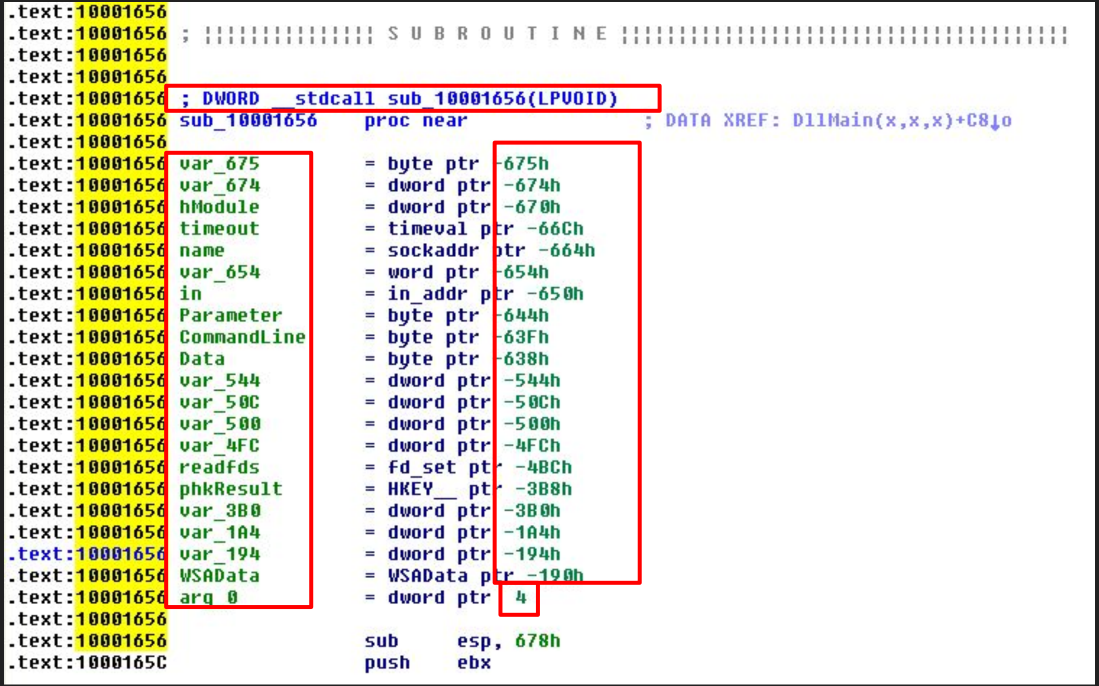
Imports

Address	Ordinal	Name	Library
100162C4		_strlwr	MSVCRT
100162C0		_strnicmp	MSVCRT
10016288		_strrev	MSVCRT
100162E8		_strtime	MSVCRT
10016258		_strupr	MSVCRT
100162E0		_vsnprintf	MSVCRT
10016268		abs	MSVCRT
100162B4		atoi	MSVCRT
100163F4	3	closesocket	WS2_32
100163...	4	connect	WS2_32
100162A4		fclose	MSVCRT
10016274		fopen	MSVCRT
100162E4		fprintf	MSVCRT
10016234		fread	MSVCRT
100162...		free	MSVCRT
100162...		fseek	MSVCRT
10016278		ftell	MSVCRT
100162A0		fwrite	MSVCRT
100163...	52	gethostbyname	WS2_32
100163E4	9	htons	WS2_32
100163C8	11	inet_addr	WS2_32
100163...	12	inet_ntoa	WS2_32
1001624C		isdigit	MSVCRT
1001638C		keybd_event	USER32
10016264		malloc	MSVCRT

Attraverso la GUI possiamo aprire la "Imports Window" che contiene tutte le funzioni importate nel codice. Cercandola possiamo trovare la funzione gethostbyname all'indirizzo 100163CC.

Static Malware Analysis: IDA PRO

S11 - L2



```
.text:10001656
.text:10001656 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.text:10001656
.text:10001656
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656    proc near               ; DATA XREF: DllMain(x,x,x)+C8↓o
.text:10001656
.text:10001656 var_675         = byte ptr -675h
.text:10001656 var_674         = dword ptr -674h
.text:10001656 hModule         = dword ptr -670h
.text:10001656 timeout         = timeval ptr -66Ch
.text:10001656 name            = sockaddr ptr -664h
.text:10001656 var_654         = word ptr -654h
.text:10001656 in              = in_addr ptr -650h
.text:10001656 Parameter       = byte ptr -644h
.text:10001656 CommandLine     = byte ptr -63Fh
.text:10001656 Data            = byte ptr -638h
.text:10001656 var_544         = dword ptr -544h
.text:10001656 var_50C         = dword ptr -50Ch
.text:10001656 var_500         = dword ptr -500h
.text:10001656 var_4FC         = dword ptr -4FCh
.text:10001656 readfds         = fd_set ptr -4BCh
.text:10001656 phkResult       = HKEY__ ptr -3B8h
.text:10001656 var_3B0         = dword ptr -3B0h
.text:10001656 var_1A4         = dword ptr -1A4h
.text:10001656 var_194         = dword ptr -194h
.text:10001656 WSAData         = WSAData ptr -190h
.text:10001656 arg_0           = dword ptr  4
.text:10001656
.text:10001656                 sub     esp, 678h
.text:1000165C                 push    ebx
```

All'indirizzo 10001656 possiamo trovare una funzione che contiene variabili e parametri. Per differenziare le funzioni dai parametri basta vedere come si riferiscono al puntatore base EBP, le funzioni sono indicate con valori negativi, mentre i parametri sono indicati con valori positivi, questo significa che ci sono 20 funzioni, e una solo 1 argomento.