MALWARE u3W2 L5

1,2

librerie importate

Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
00006664	N/A	000064F0	000064F4	000064F8	000064FC	00006500
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006000
WININET.dll	5	000065CC	00000000	00000000	00006664	000060B4

CFF Explorer

Malware_U3_W2_L5.exe

Name	Virtual Size	Virtual Address	Raw Size	Raw Address	Reloc Address	Linenumbers	Relocations ...	Linenumber...	Characteristics
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

Kernel.32.dl libreria piuttosto comune che contiene le funzioni principali per interagire con il sistema operativo, ad esempio: manipolazione file; gestione memoria.

Wininet.dll libreria che contiene le funzioni per l'implementazione di alcuni protocolli di rete http,ftp,ntp.

.text: la sezione «text» contiene le istruzioni (le righe di codice) che la CPU eseguirà una volta che il software sarà avviato. Generalmente questa è l'unica sezione di un file eseguibile che viene eseguita dalla CPU, in quanto tutte le altre sezioni contengono dati o informazioni a supporto.

.data: la sezione «data» contiene tipicamente i dati / le variabili globali del programma eseguibile, che devono essere disponibili da qualsiasi parte del programma. Ricordate che una variabile si dice globale quando non è definita all'interno di un contesto di una funzione, ma bensì è globalmente dichiarata ed è di conseguenza accessibile da qualsiasi funzione all'interno dell'eseguibile.

.rdata: la sezione «rdata» include generalmente le informazioni circa le librerie e le funzioni importate ed esportate dall'eseguibile, informazione che come abbiamo visto possiamo ricavare con CFF Explorer.

Ida

```
sub_401040 proc near

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

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

Il codice che hai fornito sembra essere un frammento di un programma scritto in linguaggio assembly. Si tratta di un'approssimazione di un'implementazione della funzione "sub_401040". Ecco una spiegazione punto per punto del codice:

1. Vengono dichiarate alcune variabili locali nella subroutine:
    - "Buffer" è un puntatore a dword situato a -210h rispetto all'EBP (puntatore alla base del frame dell'attuale subroutine).
    - "var_20C" è un puntatore a byte situato a -20Ch rispetto all'EBP.
    - "InFile" è un puntatore a dword situato a -10h rispetto all'EBP.
    - "hInternet" è un puntatore a dword situato a -0Ch rispetto all'EBP.
    - "dwNumberOfBytesRead" è un puntatore a dword situato a -8 rispetto all'EBP.
    - "var_4" è un puntatore a dword situato a -4 rispetto all'EBP.
2. Viene salvato il valore corrente del puntatore della base del frame (EBP) nello stack e viene poi impostato EBP come il valore attuale dello stack. Questo viene fatto per stabilire un nuovo frame per la subroutine.
3. Viene effettuata una sottrazione all'ESP (puntatore dello stack), riservando 210h (528 in decimale) byte per l'allocazione delle variabili locali nello stack.
4. Vengono eseguite diverse istruzioni "push" per mettere valori nello stack in preparazione alla chiamata della funzione "InternetOpenA". I valori zero sono passati come parametri per i primi quattro argomenti della funzione.
5. Viene caricato l'indirizzo della stringa "szAgent" in un registro (presumibilmente EAX).
6. Viene chiamata la funzione "InternetOpenA" attraverso la chiamata di un'etichetta di segmento (ds) per ottenere il puntatore a una connessione Internet aperta. Il valore di ritorno della funzione viene memorizzato nella variabile "hInternet" (situata a [ebp+hInternet]).
7. Vengono eseguite ulteriori istruzioni "push" per preparare gli argomenti per la chiamata alla funzione "InternetOpenUrlA".

8. Viene caricato l'indirizzo della stringa "szUr1" in un registro (presumibilmente EAX).
9. Viene chiamata la funzione "InternetOpenUrlA" attraverso la chiamata di un'etichetta di segmento (ds) per aprire un URL specifico. Il valore di ritorno della funzione viene memorizzato nella variabile "hFile" (situata a [ebp+hFile]).
10. Viene effettuato un confronto tra il valore di "hFile" e zero.
11. Se il valore di "hFile" non è zero, viene saltato a "loc_40109D" (un'etichetta di destinazione), altrimenti viene continuato l'esecuzione del codice successivo.

Questa è solo una spiegazione generale del codice fornito. Senza il contesto completo del programma, non è possibile fornire una spiegazione più dettagliata o comprenderne appieno lo scopo complessivo.

3,4

```
push    ebp
mov     ebp, esp
```

creazione dello stack

```
push    ecx
push    0                       ; dwReserved
push    0                       ; lpdwFlags
call    ds:InternetGetConnectedState
```

I parametri sono passati sullo stack tramite le istruzioni pusch

```
cmp     [ebp+var_4], 0
jz      short loc_40102B
```

Ciclo if, in caso lo zf sia impostato su 1 allora avvera il salto

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

In questo caso se zf è uguale a 0 significa che la conessione è attiva

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

In questo caso che zf è 1, vuol dire che la connessione è disattivata

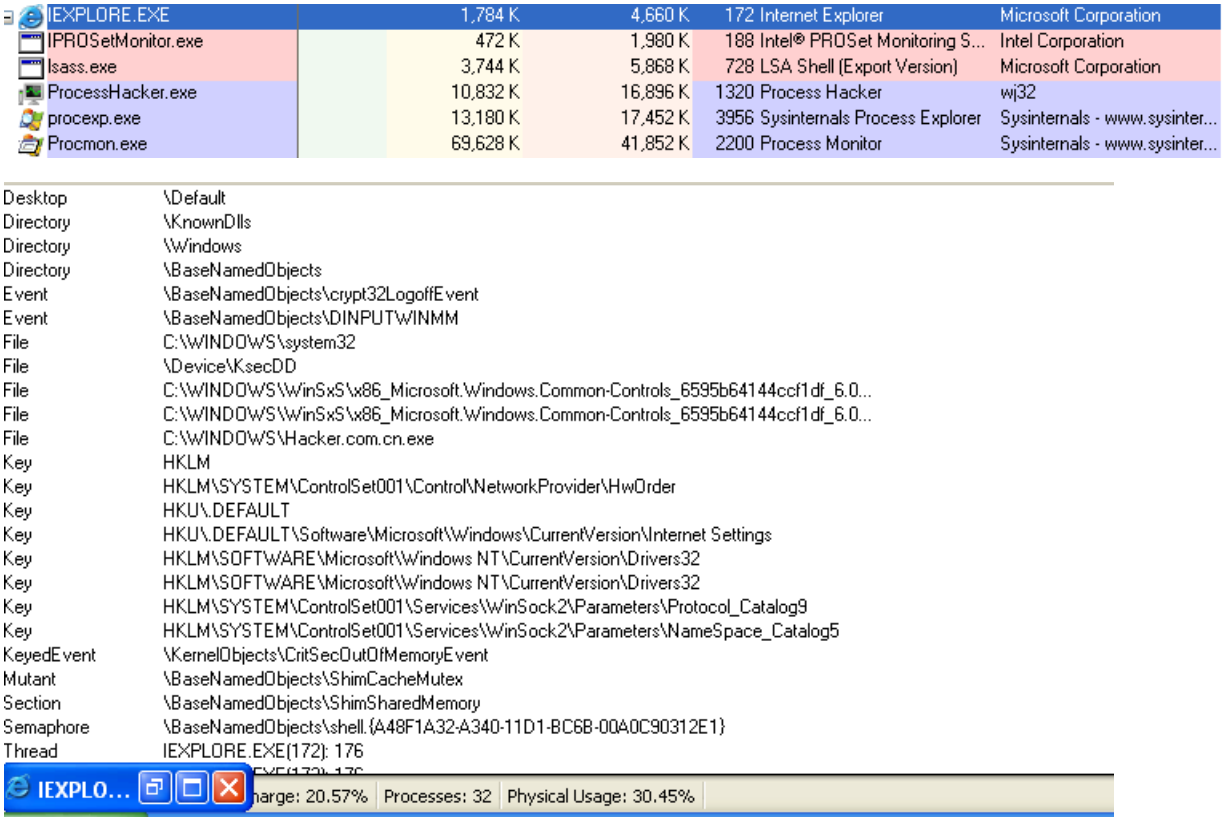
```
loc_40103A:
mov     esp, ebp
pop     ebp
retn
sub_401000 endp
```

Rimozione e pulizia dello stack.

Analisi malware si può affermare che stia cercando di verificare lo stato della connessione. Perciò si può ipotizzare che l'obiettivo del malware è di essere connesso ad internet e che quindi una delle sue altre funzionalità potrebbe essere attivare la connessione in caso fosse disattivata.

5

con **Procces Explorer** gli mostro i processi che si attivano con quel codice.



IEXPLORE.EXE		1,784 K	4,660 K	172 Internet Explorer	Microsoft Corporation
IPROSetMonitor.exe		472 K	1,980 K	188 Intel® PROSet Monitoring S...	Intel Corporation
lsass.exe		3,744 K	5,868 K	728 LSA Shell (Export Version)	Microsoft Corporation
ProcessHacker.exe		10,832 K	16,896 K	1320 Process Hacker	wj32
procexp.exe		13,180 K	17,452 K	3956 Sysinternals Process Explorer	Sysinternals - www.sysinter...
Procmon.exe		69,628 K	41,852 K	2200 Process Monitor	Sysinternals - www.sysinter...

Desktop	\Default
Directory	\KnownDlls
Directory	\Windows
Directory	\BaseNamedObjects
Event	\BaseNamedObjects\crypt32LogoffEvent
Event	\BaseNamedObjects\DINPUTWINMM
File	C:\WINDOWS\system32
File	\Device\KsecDD
File	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0...
File	C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0...
File	C:\WINDOWS\Hacker.com.cn.exe
Key	HKLM
Key	HKLM\SYSTEM\ControlSet001\Control\NetworkProvider\HwOrder
Key	HKU\.DEFAULT
Key	HKU\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32
Key	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Drivers32
Key	HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\Protocol_Catalog9
Key	HKLM\SYSTEM\ControlSet001\Services\WinSock2\Parameters\NameSpace_Catalog5
KeyedEvent	\KernelObjects\CritSecOutOfMemoryEvent
Mutant	\BaseNamedObjects\ShimCacheMutex
Section	\BaseNamedObjects\ShimSharedMemory
Semaphore	\BaseNamedObjects\shell.{A48F1A32-A340-11D1-BC6B-00A0C90312E1}
Thread	IEXPLORE.EXE(172): 176