

# SII

TEAM CYBERSECURITY

# Traccia Giorno 1

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

**1** Descrivere come il malware ottiene la persistenza , evidenziando il codice assembly dove le relative istruzioni e chiamate di funzioni vengono eseguite

**2** Identificare il client software utilizzato dal malware per la connessione ad Internet

**3** 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

**4** BONUS: qual è il significato e il funzionamento del comando assembly "lea"

## **Descrivere come il malware ottiene la persistenza , evidenziando il codice assembly dove le relative istruzioni e chiamate di funzioni vengono eseguite**

**I malware utilizzano molto spesso il registro per ottenere quella che viene chiamata «persistenza». Ovvero, il malware 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.**

### **I Software \ \Microsoft \ \Windows \ \CurrentVersion \ \Run:**

Una cosa importante da notare, è che abbiamo visto qual è una delle chiavi di registro che viene utilizzata dai malware per ottenere persistenza su un sistema operativo Windows.

#### **RegOpenKeyEx:**

questa funzione permette di aprire una chiave di registro al fine di modificarla. Essa accetta come parametri, tra gli altri, la chiave da aprire.

#### **RegSetValueEx:**

questa funzione permette invece di aggiungere un nuovo valore all'interno del registro e di settare i rispettivi dati. Accetta come parametri la chiave, la sottochiave e il dato da inserire.

# 1 — Descrivere come il malware ottiene la persistenza , evidenziando il codice assembly dove le relative istruzioni e chiamate di funzioni vengono eseguite

Il malware ottiene la persistenza inserendo un nuovo valore all'interno della chiave di registro

**Software \ Microsoft \ Windows \ CurrentVersion \ Run**, che include tutti i programmi che sono avviati all'avvio del sistema operativo.

Le funzioni utilizzate sono:

**RegOpenKey**, che permette di aprire la chiave selezionata. I parametri sono passati sullo stack tramite le istruzioni «push» che precedono la chiamata di funzione

**RegSetValueEx**, che permette al malware di inserire un nuovo valore all'interno della chiave di registro appena aperta

Approfondimento:

La funzione **lstrlenW** è una funzione di Windows che misura la lunghezza di una stringa terminata da carattere nullo (null-terminated string). È specifica per stringhe a 16 bit, cioè per stringhe in formato Unicode (UTF-16).

Traccia:

```
040286F push 2          ; samDesired
0402871 push eax        ; ulOptions
0402872 push offset SubKey ; "Software\Microsoft\Windows\CurrentVersion\Run"
0402877 push HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
040287C call esi; RegOpenKeyExW
040287E test eax, eax
0402880 jnz short loc_402882
0402882 loc_402882:
0402882 lea ecx, [esp+424h+Data]
0402886 push ecx        ; lpString
0402887 mov bl, 1
0402889 call ds:lstrlenW
040288F lea edx, [eax+eax+2]
0402893 push edx        ; cbData
0402894 mov edx, [esp+428h+hKey]
0402898 lea eax, [esp+428h+Data]
040289C push eax        ; lpData
040289D push 1          ; dwType
040289F push 0          ; Reserved
04028A1 lea ecx, [esp+434h+ValueName]
04028A8 push ecx        ; lpValueName
04028A9 push edx        ; hkey
04028AA call ds:RegSetValueExW
```

## Identificare il client software utilizzato dal malware per la connessione ad Internet

Per la gestione del networking a più ampio raggio, esistono altre API messe a disposizione da Windows che prendono il nome di «WinINet» APIs. Le APIs WinINet sono incluse nella libreria Wininet.dll.

Le funzioni di questa libreria includono funzioni per l'implementazione di protocolli di rete come HTTP ed FTP. Tra le più utilizzate dai malware, troviamo:

- **InternetOpen**: questa funzione viene utilizzata per inizializzare una connessione verso Internet
- **InternetOpenUrl**: viene utilizzata invece per la connessione ad un determinato URL. Accetta, tra gli altri parametri, un oggetto «handler» ad una connessione inizializzata con InternetOpen, e l'URL per la connessione

È molto probabile che un malware che importa la libreria Wininet.dll utilizzi una di queste funzioni per la connessione ad un determinato URL via HTTP, FTP

## Identificare il client software utilizzato dal malware per la connessione ad Internet

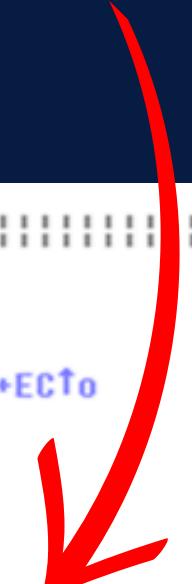
Il malware cerca di connettersi all'URL [www.malware12.com](http://www.malware12.com). La chiamata di funzione che consente al malware la connessione verso un URL è «InternetOpenURL».

L'URL è passato come parametro di questa funzione sullo stack, tramite l'istruzione push.

Il client utilizzato dal malware per connettersi ad internet è Internet Explorer, più precisamente la versione 8.

Traccia:

```
.text:00401150 ; ::::::::::::::: S U B R O U T I N E :::::::::::::::  
.text:00401150  
.text:00401150  
.text:00401150 ; DWORD __stdcall StartAddress(LPUUID)  
.text:00401150 StartAddress proc near ; DATA XREF: sub_401040+ECTo  
.text:00401150     push    esi  
.text:00401151     push    edi  
.text:00401152     push    0  
.text:00401154     push    0  
.text:00401156     push    0  
.text:00401158     push    1  
.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
```



## 3

## 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

Il malware tenta di connettersi all'URL www.malware12.com.

Per stabilire questa connessione, utilizza la funzione InternetOpenURL. L'URL viene passato come parametro a questa funzione tramite lo stack, utilizzando l'istruzione push.

```
.text:0040116D loc_40116D:  
.text:0040116D          push    0  
.text:0040116F          push    80000000h ; dwContext  
.text:00401174          push    0  
.text:00401176          push    0  
.text:00401178          push    offset szUrl ; "http://www.malware12COM  
.text:0040117D          push    esi    ; hInternet  
.text:0040117E          call    edi    ; InternetOpenUrlA  
.text:00401180          jmp    short loc_40116D  
.text:00401180 StartAddress  
.text:00401180          endp
```

; CODE XREF: StartAddress+30↓j  
; dwFlags  
; dwHeadersLength  
; lpszHeaders  
; "http://www.malware12COM  
; hInternet



## BONUS: qual è il significato e il funzionamento del comando assembly "lea"

**Il comando assembly lea** (Load Effective Address) è utilizzato per caricare un indirizzo efficace in un registro. A differenza di altri comandi di caricamento che caricano il valore contenuto in una determinata memoria, lea carica l'indirizzo di memoria calcolato in base a una particolare espressione.

### Significato di lea

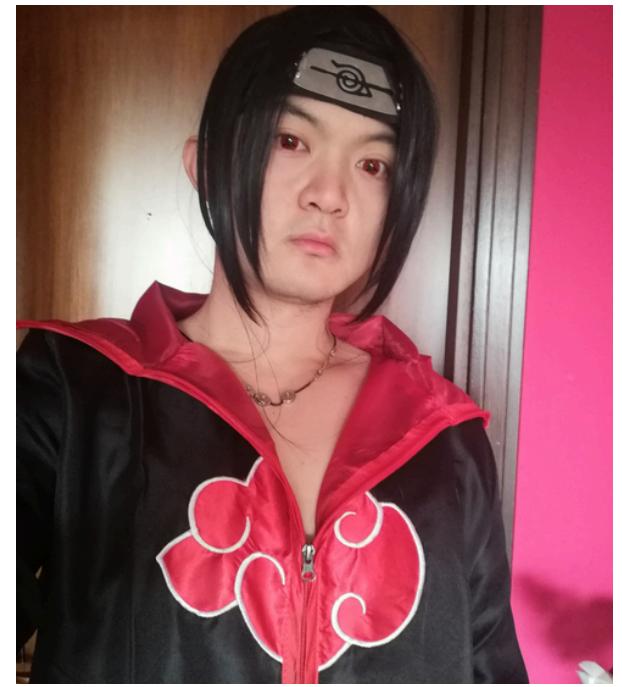
Il comando lea sta per "Load Effective Address" e viene utilizzato principalmente per calcolare l'indirizzo di un'operazione di memoria e caricarlo in un registro senza accedere effettivamente alla memoria. Questo può essere utile in vari scenari, come il calcolo di indirizzi o offset senza influire sui dati contenuti in quelle posizioni di memoria.

### Utilità di lea

- Calcolo degli indirizzi: lea è molto utile per calcolare gli indirizzi senza fare effettivamente accesso alla memoria, che può essere più veloce.
- Efficienza: Questo comando è spesso utilizzato per evitare istruzioni aritmetiche addizionali. Ad esempio, lea eax, [ebx+ecx\*4] è equivalente a mov eax, ebx seguito da add eax, ecx\*4, ma in una sola istruzione.
- Puntatori e Array: lea è comunemente usato per manipolare puntatori e array in modo efficiente.

### Conclusione

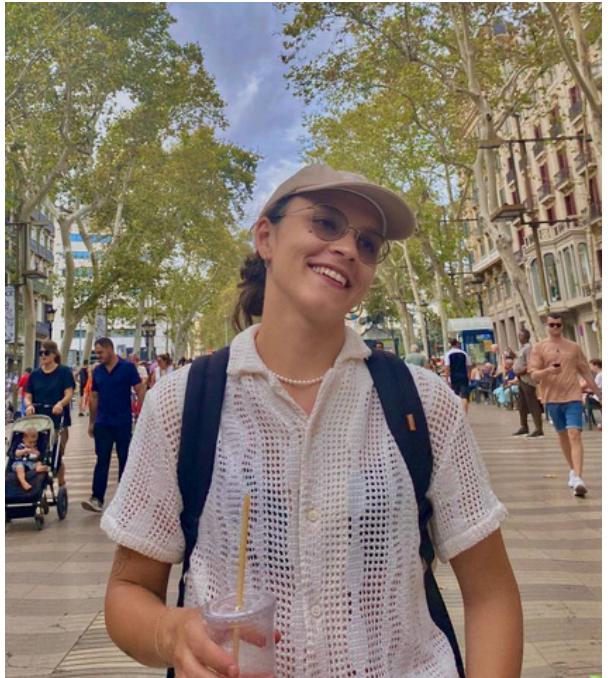
Il comando lea è uno strumento potente nell'assembly per manipolare indirizzi di memoria in modo efficiente senza accedere effettivamente alla memoria stessa. Questo lo rende utile in vari scenari di programmazione a basso livello, specialmente quando si lavora con puntatori, strutture dati complesse o si cerca di ottimizzare le prestazioni del codice.



**ZhongShi Liu**



**Mario Marsicano**



**Mara Dello Russo**



**André**

# Our Team

**Thank you!**

