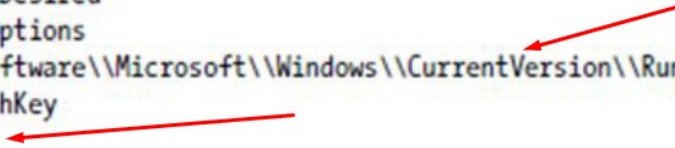


Report Analisi Statica Avanzata

Come richiesto dall'esercizio, andiamo ad individuare nel codice Assembly come il Malware ottiene la persistenza.

Il primo punto sta nel passare i parametri alla funzione **RegOpenKeyEx** sullo stack tramite il comando *push* per poi chiamarla con il comando *call*:

```
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
```



Il secondo punto sta nella chiamata alla funzione **RegSetValueEx** ed anche in questo caso, i valori vengono passati allo stack tramite il comando *push ecx* e *push edx* ed il Malware in questione modifica il valore del registro ed aggiungere una nuova entry in modo tale da avere accesso persistente all'avvio del sistema operativo:

```
004028A1  lea     ecx, [esp+434h+ValueName]
004028A8  push    ecx              ; lpValueName
004028A9  push    edx              ; hKey
004028AA  call    ds:RegSetValueExW
```

Possiamo infine dire che nel registro `Software\\Microsoft\\Windows\\CurrentVersion\\Run` il Malware si insidia per andare a modificare i valori ed ottenere, infine, accesso persistente.

Adesso andiamo ad identificare nella slide 3, nella subroutine data, qual è il client software e l'URL a cui cerca di connettersi.

```

-----
.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+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 Client Software:
.text:00401165 mov edi, ds:InternetOpenUrlA Internet Explore 8.0
.text:00401168 mov esi, eax
.text:0040116D loc_40116D: ; CODE XREF: StartAddress+30↓j
.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.malware12COM
.text:0040117D push esi ; hInternet
.text:0040117E call edi ; InternetOpenUrlA
.text:00401180 jmp short loc_40116D URL: http://www.malware12.com
.text:00401180 StartAddress endp
.text:00401180
-----

```

Come abbiamo visto nei precedenti screen, viene usato un comando chiamato lea, ma cos'è effettivamente? LEA sta per Load Effective Address ed in breve, LEA carica un puntatore all'elemento che si sta indirizzando, a differenza di MOV che carica il valore effettivo a quell'indirizzo. Lo scopo di LEA è quello di consentire l'esecuzione di un calcolo di indirizzo non banale e la memorizzazione del risultato.