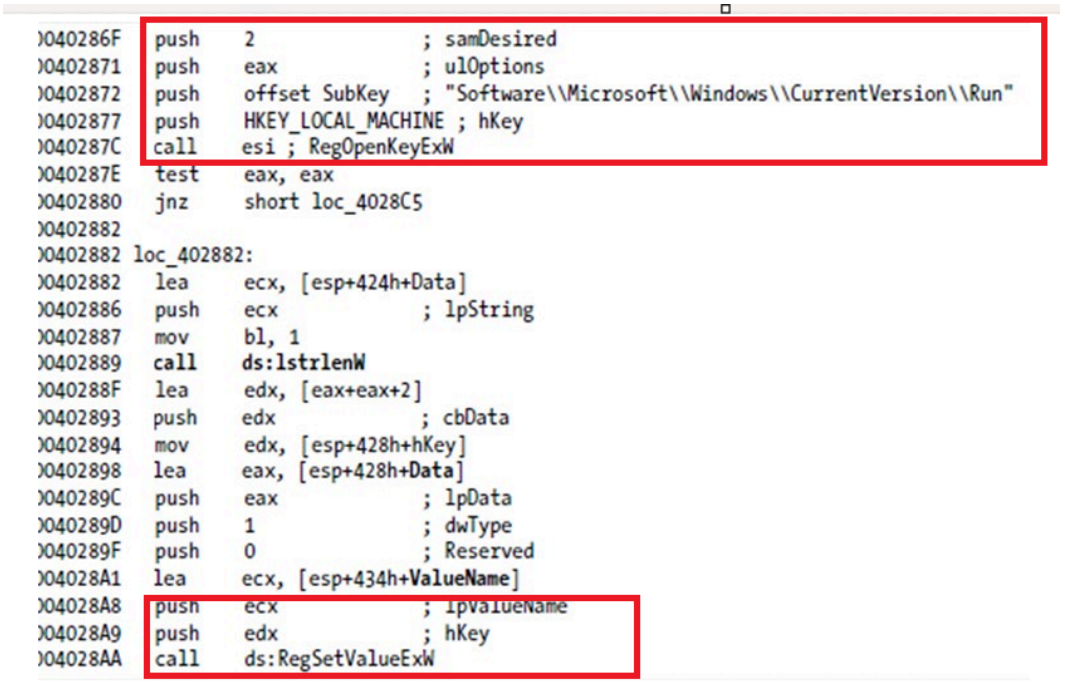
S11/L1

Traccia:
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




Per prima cosa notiamo la key: HKEY_LOCAL_MACHINE(HKLM): dove sono contenuti i record e le configurazioni della macchina.
La figura mostra il codice di un malware che apre una chiave di registro per aggiungere un valore in modo tale da ottenere persistenza.
Vediamo la chiamata alla funzione **RegOpenKeyEx** e 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. In sintesi questo permette al malware di avviarsi automaticamente ogni volta che si avvia il computer.
Vediamo poi la chiamata alla funzione **RegSetValueEx.** Anche in questo caso i valori sono passati sullo stack tramite le istruzioni «pushecx» e «pushedx». 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.

Una delle chiavi di registro che viene utilizzata dai malware per ottenere **persistenza** su un sistema operativo Windows:
Software\\Microsoft\\Windows\\CurrentVersion\\Run
che utilizza "push" per inserire un valore specifico nello stack della CPU. In questo caso, il valore è l'offset di una stringa che rappresenta il percorso del registro di sistema di Windows.. In particolare, sta probabilmente inserendo il percorso di una chiave di registro specifica (in questo caso, la chiave di registro per le applicazioni che si avviano all'avvio di Windows) nello stack per eseguire operazioni su di essa, come la modifica o la lettura dei suoi valori.

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

```
; DWORD __stdcall StartAddress(LPVOID)
StartAddress    proc near               ; DATA XREF: sub_401040+EC↑o
                push    esi
                push    edi
                push    0               ; dwFlags
                push    0               ; lpszProxyBypass
                push    0               ; lpszProxy
                push    1               ; dwAccessType
                push    offset szAgent  ; "Internet Explorer 8.0"
                call    ds:InternetOpenA
                mov     edi, ds:InternetOpenUrlA
                mov     esi, eax

loc_40116D:                             ; CODE XREF: StartAddress+30↓j
                push    0               ; dwContext
                push    80000000h       ; dwFlags
                push    0               ; dwHeadersLength
                push    0               ; lpszHeaders
                push    offset szUrl    ; "http://www.malware12COM
                push    esi             ; hInternet
                call    edi ; InternetOpenUrlA
                jmp     short loc_40116D
StartAddress    endp
```

InternetOpen: questa funzione viene utilizzata per inizializzare una connessione verso Internet, ed il client software utilizzato è Internet Explorer 8.0

• 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

```
                        ; CODE XREF: StartAddress+30↓j
  push    0             ; dwContext
  push    80000000h     ; dwFlags
  push    0             ; dwHeadersLength
  push    0             ; lpszHeaders
  push    offset szUrl  ; "http://www.malware12COM
  push    esi           ; hInternet
  call    edi ; InternetOpenUrlA
  jmp     short loc_40116D
  endp
```

Qui possiamo vedere l'inizializzazione del parametro dell'URL e il suo passaggio alla funzione InternetOpenUrl. Ciò consente l'utilizzo dell'handler che gestisce la connessione hInternet e l'apertura dell'URL.

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

In generale, “lea” è spesso utilizzato per calcolare gli indirizzi degli operandi per istruzioni future, come ad esempio quando si accede a un elemento in un array o quando si calcola l'indirizzo di una variabile. È una delle istruzioni più utili e potenti in assembly per quanto riguarda la manipolazione degli indirizzi di memoria.

> lea ecx, [esp+424+Data]: Qui, LEA calcola l'indirizzo effettivo dell'operando [esp+424+Data] e lo carica nel registro ECX.
> lea edx, [eax+eax+2]: LEA calcola l'indirizzo effettivo dell'operando [eax+eax+2] e lo carica nel registro EDX.
> lea eax, [esp+428h+Data]: LEA calcola l'indirizzo effettivo dell'operando [esp+428h+Data] e lo carica nel registro EAX.

Questi comandi LEA non stanno effettuando operazioni di caricamento di dati dalla memoria, ma stanno calcolando gli indirizzi effettivi degli operandi specificati. Gli indirizzi effettivi vengono utilizzati successivamente per riferirsi ai dati o alle istruzioni in memoria, come nel caso del push degli operandi su uno stack prima di chiamare una funzione. Quindi, in breve, il comando LEA in questo contesto viene utilizzato per calcolare gli indirizzi di memoria necessari per gli operandi di successive istruzioni.