

# Malware analysis – OllyDBG

## Sommario

Traccia S11-L3 .....	1
Step 1.....	1
Step 2.....	1
Step 3.....	3

## Traccia S11-L3

Fate riferimento al malware: Malware\_U3\_W3\_L3, presente all'interno della cartella  
Esercizio\_Pratico\_U3\_W3\_L3 sul desktop della macchina virtuale dedicata all'analisi dei malware.  
Rispondete ai seguenti quesiti utilizzando OllyDBG.

- Step 1: all'indirizzo 0040106E il Malware effettua una chiamata di funzione alla funzione «CreateProcess». Qual è il valore del parametro «CommandLine» che viene passato sullo stack? (1)
- Step 2: inserite un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX? (2) Eseguite a questo punto uno «step-into». Indicate qual è ora il valore del registro EDX (3) motivando la risposta (4). Che istruzione è stata eseguita? (5)
- Step 3: inserite un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX? (6) Eseguite un step-into. Qual è ora il valore di ECX? (7) Spiegate quale istruzione è stata eseguita (8).



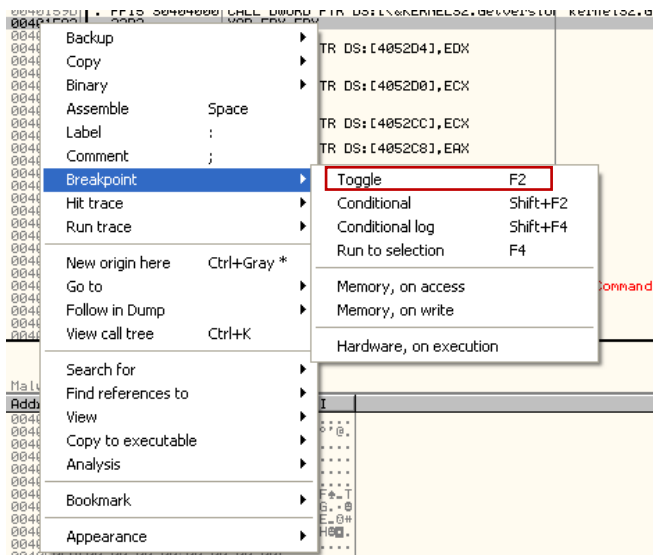
## Step 1

00401055	. 8055 F0	LEA EDX, DWORD PTR SS:[EBP-10]	pProcessInfo
00401056	. 52	PUSH EDX	pStartupInfo
00401057	. 8045 A8	LEA EAX, DWORD PTR SS:[EBP-58]	CurrentDir = NULL
0040105A	. 50	PUSH EAX	pEnvironment = NULL
0040105B	. 6A 00	PUSH 0	CreationFlags = 0
0040105D	. 6A 00	PUSH 0	InheritHandles = TRUE
0040105F	. 6A 00	PUSH 0	pThreadSecurity = NULL
00401061	. 6A 01	PUSH 1	pProcessSecurity = NULL
00401063	. 6A 00	PUSH 0	CommandLine = "cmd"
00401065	. 6A 00	PUSH 0	ModuleFileName = NULL
00401067	. 68 30504000	PUSH Malware_.00405030	
0040106C	. 6A 00	PUSH 0	
0040106E	. FF15 04404000	CALL DWORD PTR DS:[<&KERNEL32.CreatePro	CreateProcessA

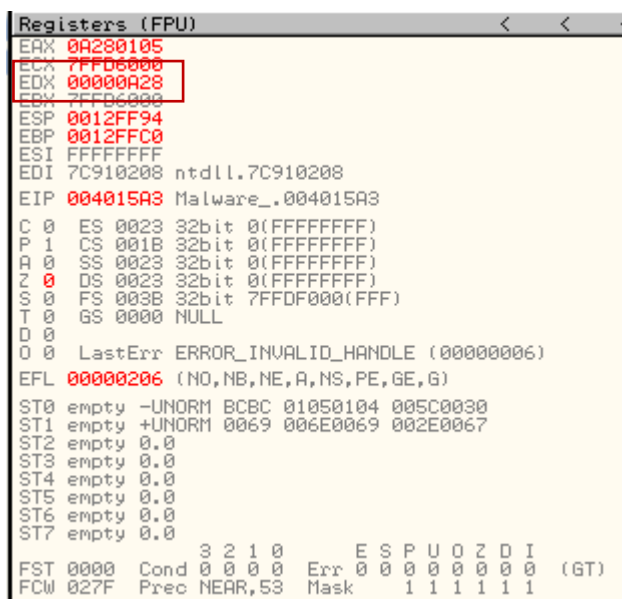
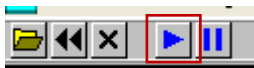
Il valore del parametro «CommandLine» passato sullo stack è "cmd"

## Step 2

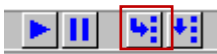
Inseriamo il breakpoint cliccando col tasto destro sull'indirizzo indicato e successivamente su breakpoint e toggle.



Avviamo il debugging del codice col tasto play e possiamo vedere che nella finestra Registers (FPU) che il valore del registro EDX è 00000A28.



Ora eseguiamo lo step-into cliccando sul tasto nel riquadro rosso. Lo utilizziamo per esaminare righe di codice e, a fronte di una chiamata di funzione, per accedere alla sua implementazione



```

Registers (FPU)
EAX 0A280105
ECX 7FFD6000
EDX 00000000
EBX 7FFD6000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015A5 Malware_.004015A5
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_INVALID_HANDLE (00000006)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty -UNORM BCBC 01050104 005C0030
ST1 empty +UNORM 0069 006E0069 002E0067
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,S3 Mask 1 1 1 1 1 1

```

Il valore del registro EDX ora è 0, perché la funzione XOR contenuta inizializza il registro a 0, siccome il risultato di uno XOR tra due operandi uguali è sempre 0.

00401599	: 57	PUSH EDI	
0040159A	: 8965 E8	MOV DWORD PTR SS:[EBP-18],ESP	
0040159D	: FF15 30404000	CALL DWORD PTR DS:[<&KERNEL32.GetVersion	kernel32.GetVersion
004015A3	: 33D2	XOR EDX,EDX	
004015A5	: 8AD4	MOV DL,AH	
004015A7	: 8915 04524000	MOV DWORD PTR DS:[4052D4],EDX	

### Step 3

Inseriamo un secondo breakpoint all'indirizzo di memoria 004015AF. Il valore del registro ECX è 0A280105.

```

Registers (FPU)
EAX 0A280105
ECX 0A280105
EDX 00000001
EBX 7FFD6000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015AF Malware_.004015AF
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_INVALID_HANDLE (00000006)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty -UNORM BCBC 01050104 005C0030
ST1 empty +UNORM 0069 006E0069 002E0067
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,S3 Mask 1 1 1 1 1 1

```

Effettuiamo nuovamente anche lo step-in e possiamo vedere che il registro ECX ha ora il valore di 5.

```

Registers (FPU)
EAX 0A280105
ECX 00000005
EDX 00000001
EBX 7FFD5000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015B5 Malware_.004015B5
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_INVALID_HANDLE (00000006)
EFL 00000206 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty -UNORM BCBC 01050104 005C0030
ST1 empty +UNORM 0069 006E0069 002E0067
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

Questo perché in questa posizione l'istruzione **and ECX, OFF** esegue un'operazione AND bit a bit tra il valore nel registro ECX e Off.

004015A7	8915	D4524000	MOV DWORD PTR DS:[4052D4],EDX
004015AD	8BC8		MOV ECX,EAX
004015AF	81E1	FF000000	AND ECX,0FF
004015B5	890D	D0524000	MOV DWORD PTR DS:[4052D0],ECX
004015BB	C1E1	08	SHL ECX,8
004015BE	03CA		ADD ECX,EDX

ECX: 00001010 00101000 00000001 00000101 (0A280105 in binario)

OFF: 00000000 00000000 00000000 11111111 (000000FF in binario)

-----

Risultato: 00000000 00000000 00000000 00000101 (00000005 in binario)