


Epicode Unit 3 Week 3

Debug Analysis

 **EPICODE**

Esercizio
OllyDBG

Traccia:

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.

- 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)
- 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)
- 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).
- BONUS: spiegare a grandi linee il funzionamento del malware

3

Una volta avviato il tool oggetto di studio, procedo all'analisi necessaria alla risoluzione dei quesiti in traccia.

- 1) Andando ad analizzare lo specifico indirizzo di memoria, si può notare che il valore della funzione è “cmd”, prompt dei comandi di Windows.

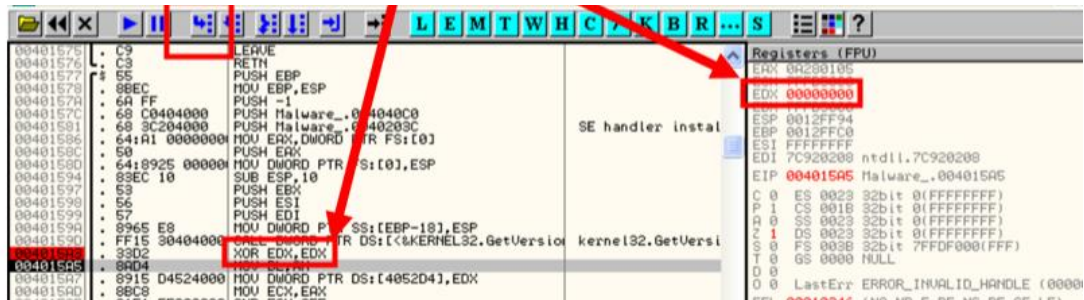
0040106C	68 30504000	PUSH Malware_.00405030	CommandLine = "cmd"
0040106E	6A 00	PUSH 0	hObject = NULL
	FF15 04404000	CALL DWORD PTR DS:[&KERNEL32.CreateProcessA]	CreateProcessA
	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
00401077	6A FF	PUSH -1	Timeout = INFINITE
00401079	8B40 F0	MOV ECX, DWORD PTR SS:[EBP-10]	
0040107C	51	PUSH ECX	hObject
0040107D	FF15 00404000	CALL DWORD PTR DS:[&KERNEL32.WaitForSingleObject]	WaitForSingleObject
00401083	33C0	XOR EAX,EAX	
00401085	8BE5	MOV ESP,EBP	
00401087	5D	POP EBP	
00401089	C3	RETN	

- 2) Andando ad inserire un Breakpoint nello specifico indirizzo di memoria attraverso il comando “Toggle Breakpoint” del tool, possiamo andare a forzare il punto di “arresto” dell'esecuzione per analizzare più nello specifico l'eseguibile senza che il debugger continui a leggerlo. Una volta fatto ciò, si può notare che il valore di EDX corrisponde a 00000A28

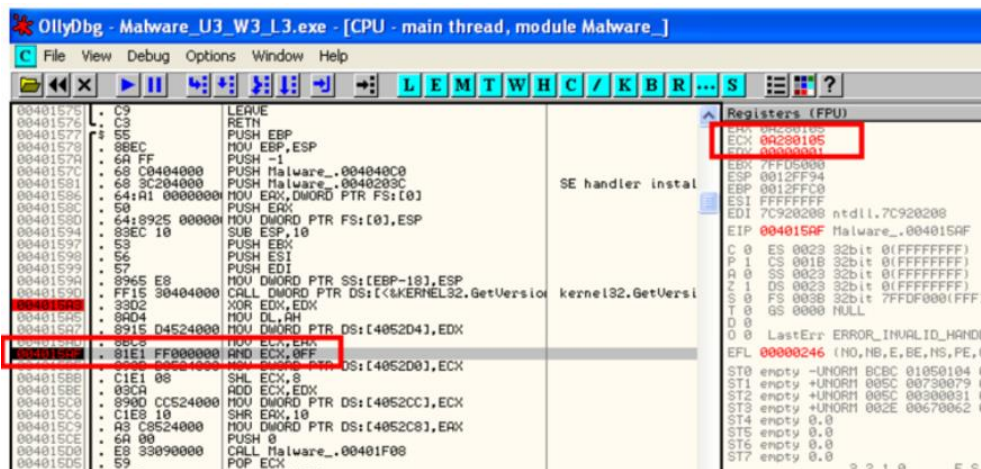
004015A3	FF15 00404000	CALL DWORD PTR DS:[&KERNEL32.GetUserS	kernel32.GetUserS
004015A5	33D2	XOR EDI,EDI	
004015A7	8A04	MOV DL,AH	
004015A9	8715 04524000	MOV DWORD PTR DS:[4052D4],EDI	
004015AD	8BC8	MOV EAX,ECX	

EDX	00000A28
EDI	77F52000
EIP	004015A3
EBP	0012FFC0
ESI	FFFFFFFF
EDI	7C920200 ntdll.7C920200
EIP	004015A3 Malware_.004015A3
C 0	ES 0023 32bit 0(FFFFFFFF)
P 1	CS 001B 32bit 0(FFFFFFFF)
D 0	DS 0023 32bit 0(FFFFFFFF)
Z 0	DS 0023 32bit 0(FFFFFFFF)
S 0	FS 005B 32bit 77FDF000(FF)
T 0	GS 0000 NULL

- 3) Per praticità, anche i punti 4 e cinque della traccia verranno trattati insieme al punto tre. Con la funzione “Step Into” sarà possibile analizzare in maniera più approfondita l’eleguibile in particolare in presenza di Funzioni “Custom” che potrebbero essere di più difficile riconoscimento e/o comprensione. Nel caso specifico possiamo notare che il registro EDX è inizializzato a zero come riportato dallo XOR (OR esclusivo) individuato nella chiamata di funzione.



- 6) Una volta inserito un secondo breakpoint all’indirizzo di memoria richiesto, e premuto “play” per analizzare la funzione in maniera specifica, trovo che in questo caso il registro ECX si trova in 0A280105.



- 7-8) Dopo aver utilizzato il comando “Step Into” e aver eseguito di nuovo il comando play, vedo che il valore di ECX, prima uguale al registro EAX, viene modificato dal valore 0FF (esadecimale di 255) mediante l’operatore logico “AND”, andando a “pulire” il registro ad eccezione del valore esadecimale di cui sopra, ottenendo il valore 00000005.

```
Registers (FPU)
EAX 0A280105
ECX 00000005
EDX 00000001
EBX 7FFD4000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015B5 Malware_.004015B5
```

- 9) A giudicare dalle funzioni analizzate, dal valore “cmd” individuato al punto uno, dalla funzione “GetCommandLine”, ormai nota, insieme anche a tutte le funzioni relative alla gestione dei processi (“Create/Terminate Process”), all’individuazione della libreria WS2_32, utilizzata per la creazione e gestione di risorse internet e Socket, insieme alla funzione “sleep”, che potrebbe essere indicativa della Persistenza ottenuta sulla macchina infetta, mi porta a pensare che l’eleggibile sia a tutti gli effetti un Trojan.

```
Flags = 0
Group = 0
pWSAprotocol = NULL
Protocol = IPPROTO_TCP
Type = SOCK_STREAM
Family = AF_INET
WSASocketA

Socket
closesocket
WSACleanup
Timeout = 30000. ms
Sleep

pStartupInfo
GetStartupInfoA

pProcessInfo
pStartupInfo
CurrentDir = NULL
pEnvironment = NULL
CreationFlags = 0
InheritHandles = TRUE
pThreadSecurity = NULL
pProcessSecurity = NULL
CommandLine = "cmd"
ModuleFileName = NULL
CreateProcessA
```