

Analisi Malware con debugger


00401053	8D55 F0	LEA EDX,DWORD PTR SS:[EBP-10]	
00401056	52	PUSH EDX	
00401057	8D45 A8	LEA EAX,DWORD PTR SS:[EBP-58]	
0040105A	50	PUSH EAX	
0040105B	6A 00	PUSH 0	
0040105D	6A 00	PUSH 0	
0040105F	6A 00	PUSH 0	
00401061	6A 01	PUSH 1	
00401063	6A 00	PUSH 0	
00401065	6A 00	PUSH 0	
00401067	68 30504000	PUSH Malware_.00405030	
0040106C	6A 00	PUSH 0	
0040106E	FF15 04404000	CALL DWORD PTR DS:[<&KERNEL32.CreateProcessA>]	CreateProcessA
00401074	094E 5C	MOVB BYTE PTR DS:[EBP-14],EBX	

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

Come vediamo Il malware crea un processo passando il parametro "cmd"

Andiamo poi a impostare il breakpoint:

0040159H	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 D4524000	MOV DWORD PTR DS:[4052D4],EDX	

Eseguiamo il malware con  (play) e vediamo il valore di edx prima dell'operazione al breakpoint

Registers (MMX)	
EAX	0A280105
ECX	7FFD9000
EDX	00000A28
EBX	7FFD9000
ESP	0012FF94
EBP	0012FFC0

Facciamo uno  (step-into) il breakpoint e notiamo il valore che assume EDX

Registers (MMX)	
EAX	0A280105
ECX	7FFDE000
EDX	00000000
EBX	7FFDE000
ESP	0012FF94
EBP	0012FFC0
ESI	FFFFFFFF
EDI	7C910208 ntdll.7

Lo XOR di un valore per se stesso non può che dare 0:

A	B	A ∨ B
0	0	0
0	1	1
1	0	1
1	1	0

(tabella di XOR)

Impostiamo ora il secondo breakpoint, e andiamo a eseguire:

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	

Questo è il valore di ECX prima dell'operazione

```
Registers (MMX)
EAX 0A280105
ECX 0A280105
EDX 00000001
EBX 7FFD5000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C9
```

Dopo lo step-into vediamo che il valore è cambiato:

```
Registers (MMX)
EAX 0A280105
ECX 00000005
EDX 00000001
EBX 7FFD5000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
```

Questo perché il calcolo tra quei due valori esadecimali in binario da questo risultato:

Porta logica AND $Y = A \cdot B$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



(tabella AND)

Possiamo anche andarlo a testare manualmente

Per prima convertiamo in binario

Hex to Binary converter

From: Hexadecimal To: Binary

Enter hex number: 0A280105

= Convert x Reset Swap

Binary number (28 digits): 1010001010000000000100000101

Hex to Binary converter

From: Hexadecimal To: Binary

Enter hex number: FF

= Convert x Reset Swap

Binary number (8 digits): 11111111

L'AND ora si calcola molto semplicemente incolonnando i risultati e applicando la tabella precedente

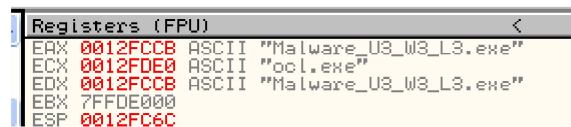
```

1010001010000000000100000101 AND
0000000000000000000011111111 =
0000000000000000000000000101 RISULTATO
  
```

Funzione del malware

Il malware è una reverse shell, è abbastanza facile da capire, apre cmd in background, usa un socket per connettersi e resta in attesa.

Il malware ha una condizione di funzionamento, fa un controllo di stringhe, tra nome programma e "ocl.exe", se il nome del programma è diverso si chiude.



Qui vediamo il salto condizionale

```
CALL Malware_.00701700  
ADD ESP, 8  
TEST EAX, EAX  
JE SHORT Malware_.0040124C  
MOV EAX, 1  
JMP Malware_.00401306
```

Se il test ritorna vero, allora il programma salterà a questo indirizzo, altrimenti si chiuderà