

# ANALISI MALWARE CON ONLYDBG

L'esercizio di oggi ci richiede di effettuare un analisi con il tool ONLYDBG

La prima parte consiste nel trovare il valore del parametro (commandLine) che viene passato sullo Stack.

Il parametro passato è: **cmd** come si può notare dalla foto sotto

```
0040103B . 66:C745 D8 00 MOV WORD PTR SS:[EBP-28],0
00401041 . 8955 18 MOU EDX,DWORD PTR SS:[EBP+18]
00401044 . 8955 E0 MOU DWORD PTR SS:[EBP-20],EDX
00401047 . 8945 E0 MOU EAX,DWORD PTR SS:[EBP-20]
0040104A . 8945 E8 MOU DWORD PTR SS:[EBP-18],EAX
0040104D . 8840 E8 MOU ECX,DWORD PTR SS:[EBP-18]
00401050 . 8940 E4 MOU DWORD PTR SS:[EBP-1C],ECX
00401053 . 8055 F0 LEA EDX,DWORD PTR SS:[EBP-10]
00401056 . 52 PUSH EDX
00401057 . 8045 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]
00401074 . 8945 EC MOU DWORD PTR SS:[EBP-14],EAX
00401077 . 6A FF PUSH -1
00401079 . 884D F0 MOV ECX,DWORD PTR SS:[EBP-10]
0040107C . 51 PUSH ECK
0040107D . FF15 00404000 CALL DWORD PTR DS:[&KERNEL32.WaitForSingleObject]
00401083 . 33C0 XOR EAX,EAX
00401085 . 8BE5 MOU ESP,EBP
00401087 . 50 POP EBP
00401088 . C3 RETN
00401089 . 55 PUSH EBP
0040108A . 8BED MOU EBP,ESP
0040108C . B1EC 00010000 SUB ESP,108
00401092 . 57 PUSH EDI
00401093 . C785 F8FFFF MOU DWORD PTR SS:[EBP-108],0
0040109D . C685 00FFFFFF MOU BYTE PTR SS:[EBP-100],0
004010A4 . B9 3F000000 MOU ECX,3F
004010A9 . 33C0 XOR EAX,EAX
004010AB . 80BD 01FFFFFF LEA EDI,DWORD PTR SS:[EBP-FF]

00405030=Malware_.00405030 (ASCII "cmd")
```

Il secondo punto dell'esercizio invece ci chiede di inserire un breakpoint software all'indirizzo 004015A3.

Il valore di partenza di EDX è di 7c90e4f4 in esadecimale mentre in decimale è 208987050

Dopo di che facendo step-into ad un certo punto ci porta dentro la funzione dove il valore di EDX cambia nuovo è diventa in esadecimale 00000a28 in decimale 2600 come possiamo vedere dalla foto qui sotto

The screenshot shows the Immunity Debugger interface. The assembly pane displays the following assembly code:

```
7C81126A 64:A1 18000000 MOV EDX,DWORD PTR FS:[18]
7C811270 8848 30 MOV ECX,DWORD PTR DS:[ECX+30]
7C811273 8800 B00000000 MOV EDX,DWORD PTR DS:[ECX+B0]
7C81127E 8B00 A00000000 MOV EDX,DWORD PTR DS:[ECX+A0]
7C811280 89F9 FE XOR EDX,0FFFFFFE
7C811281 C1E0 0E SHL EDX,8
7C811286 0B02 OR EDX,EDX
7C811288 C1E0 08 SHL EDX,8
7C81128B 0B01 A80000000 OR EDX,DWORD PTR DS:[ECX+A8]
7C811291 C1E0 08 SHL EDX,8
7C811294 0B01 A40000000 OR EDX,DWORD PTR DS:[ECX+A4]
7C81129A C3 RETN
7C81129B 90 NOP
7C81129C 90 NOP
7C81129D 90 NOP
7C81129E 90 NOP
7C81129F 90 NOP
7C8112A0 8BF7 MOU EDI,EDI
7C8112A2 55 PUSH EBP
7C8112A3 8BEC MOV EBP,ESP
7C8112A5 83EC 24 SUB ESP,24
7C8112A8 57 PUSH EDI
7C8112A9 6A 07 PUSH 7
7C8112AB 33C0 XOR EDX,EDX
7C8112AD 2145 FC AND DWORD PTR SS:[EBP-4],EDX
7C8112B0 59 POP ECX
7C8112B1 807D F0 LEA EDI,DWORD PTR SS:[EBP-20]
7C8112B4 C745 DC 20000000 MOV EDI,DWORD PTR SS:[EBP-20]
7C8112B8 F3:AB REP STOS DWORD PTR ES:[EDI]
7C8112BD 8D 10 MOV EDI,DWORD PTR SS:[EBP+10]
7C8112C0 85FF TEST EDI,EDI
7C8112C2 0F84 17EA0100 JE kernel32._7C82FCDF
7C8112C8 2107 AND DWORD PTR DS:[EDI],EAX
7C8112CB 56 PUSH ESI
7C8112CE 8875 08 MOV ESI,DWORD PTR SS:[EBP+8]
7C8112CD 8045 FC LEA EAX,DWORD PTR SS:[EBP-4]
7C8112D1 50 PUSH EAX
7C8112D2 A0 00 PIMH ?
```

The registers pane shows the following register state:

Register	Value
ERX	00000002
ECX	7FFDE000
EDX	00000A28
EBX	7FFDE000
ESP	0012FF90
EIP	0012FC00
ESI	FFFFFFFFFF
EDI	7C910208 ntdll.7C910208
EIP	7C811280 kernel32.7C811280
C	0 ES 0023 32bit 0xFFFFFFFF
P	1 CS 0018 32bit 0xFFFFFFFF
A	0 SS 0023 32bit 0xFFFFFFFF
Z	0 DS 0023 32bit 0xFFFFFFFF
S	0 FS 003B 32bit 0xFFFF0000
T	0 GS 0000 NULL
D	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

Dopo usciti dalla funzione possiamo notare come EDX cambia di nuovo di valore è diventa 0 perché xor dello stesso oggetto dà come risultato sempre 0

```
Registers (FPU) < < < < < < < < <
EAX 0A280105
ECX 7FFD9000
EDX 00000000
EBX 7FFD9000
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
          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 023F Prec NEAR,53 Mask 1 1 1 1 1 1
```

Successivamente il valore di EDX cambia di nuovo è diventa 1

```
Registers (FPU) < < < < < < < < <
EAX 0A280105
ECX 7FFD9000
EDX 00000001
EBX 7FFD9000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015A7 Malware_.004015A7
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
          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 023F Prec NEAR,53 Mask 1 1 1 1 1 1
```

L'ultimo punto dell'esercizio ci richiede di mettere un secondo breakpoint all'indirizzo di memoria 004015AF e ci chiede di controllare il valore di ECX che in questo caso è in esadecimale 0012ffb0 in decimale è 1245104

```

00401572: $5 PUSH EBP
00401573: .8BEC MOU EBP,ESP
00401574: .6A FF PUSH -1
00401575: .68 C0404000 PUSH Malware_.004040C0
00401576: .68 3C004000 PUSH Malware_.004040C0
00401577: .50 PUSH EAX
00401578: .64:A1 00000000 MOU EAX,DWORD PTR FS:[0]
00401579: .50 PUSH EAX
0040157A: .64:8925 00000000 SUB DWORD PTR FS:[0],ESP
0040157B: .89EC 10 SUB ESP,10
0040157C: .56 PUSH EDI
0040157D: .57 PUSH EDI
0040157E: .89E5 E8 MOU DWORD PTR SS:[EBP-18],ESP
0040157F: .FF15 30404000 CALL DWORD PTR DS:[<&KERNEL32.GetVersion>]
00401580: .33D2 ADD ECX,EDX
00401581: .8AD4 MOU DL,AL
00401582: .8915 D4524000 MOU DWORD PTR DS:[4052D4],EDX
00401583: .8BC8 MOU ECX,EAX
00401584: .81E1 FF000000 AND ECX,BFF
00401585: .89E5 00524000 MOU DWORD PTR DS:[4052D0],ECX
00401586: .89E1 08 SHL ECX,8
00401587: .83CA ADD ECX,EDX
00401588: .8900 CCS524000 MOU DWORD PTR DS:[4052CC],ECX
00401589: .C1E8 10 SHR EAX,10
0040158A: .A3 C8524000 MOU DWORD PTR DS:[4052C8],ERX
0040158B: .6A 00 PUSH 0
0040158C: .E8 33090000 CALW Malware_.00401F08
0040158D: .59 POP ECX
0040158E: .85C0 TEST EAX,EAX
0040158F: .75 08 JNZ SHORT Malware_.004015E2
00401590: .6A 1C PUSH ECX
00401591: .E8 9A000000 CALL Malware_.0040167B
00401592: .E9 09 JMP ECX
00401593: .83E5 FC 00 AND DWORD PTR SS:[EBP-4],0
00401594: .E8 72070000 CALL Malware_.00401D50
00401595: .FF15 2C404000 CALL DWORD PTR DS:[<&KERNEL32.GetCommandLineA>]
00401596: .A3 D8574000 MOU DWORD PTR DS:[4057D8],ERX
00401597: .E8 30060000 CALL Malware_.00401C28

```

Registers (FPU)

ERX	00000000
ECX	0012FFB0
EDX	7990E4F4 ntdll.KiFastSystemCallRet
EBX	7FFD90000
ESP	0012FFC4
EBP	0012FFC0
ESI	FFFFFFF0
EDI	7C910208 ntdll.7C910208
EIP	00401577 Malware_.<ModuleEntryPoint>
C 0	ES 0023 32bit 0xFFFFFFFF
P 1	CS 001B 32bit 0xFFFFFFFF
A 0	SS 0023 32bit 0xFFFFFFFF
Z 1	S 0023 32bit 0xFFFFFFFF
S 0	FS 003B 32bit 7FFDF000(FFF)
T 0	GS 0000 NULL
D 0	LastErr ERROR_INVALID_HANDLE (00000006)
EFL	00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty -UNORM BCBC 010501B4 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 3 2 1 0 Err 0 S P U O Z D I  
FCW 023F Prec NEAR,S3 Mask 1 1 1 1 1 1 (GT)

Possiamo vedere con all'interno della funzione il valore di ECX cambia di nuovo diventando in esadecimale 7ffd9000 in decimale è 2147323904

```

7C81126H: 64:61 18000000 MOU ECX,DWORD PTR FS:[18]
7C811270: 8B04 30 MOU ECX,DWORD PTR DS:[ECX+30]
7C811271: 8881 00000000 MOU ECX,DWORD PTR DS:[ECX+B0]
7C811272: 0FB791 AC000000 MOUZX EDX,WORD PTR DS:[ECX+AC]
7C811273: 89F0 FE XOR EAX,FFFFFE
7C811274: C1E0 0E SHL EAX,0E
7C811275: 0BC2 OR ERX,EDX
7C811276: C1E0 08 SHL ERX,8
7C811277: 00 00000000 MOU ECX,DWORD PTR DS:[ECX+A8]
7C811278: 00 00000000 SHL ERX,8
7C811279: C1E0 08 OR ERX,DWORD PTR DS:[ECX+A4]
7C81127A: C3 RETN
7C81127B: 90 NOP
7C81127C: 90 NOP
7C81127D: 90 NOP
7C81127E: 90 NOP
7C81127F: 90 NOP
7C811280: 8FF MOU EDI,EDI
7C811281: 55 PUSH EBP
7C811282: 88EC 24 MOU EBP,ESP
7C811283: 89EC 24 SUB ESP,24
7C811284: 57 PUSH EDI
7C811285: 6A 07 PUSH EAX
7C811286: 00 000000 MOU ECX,EAX
7C811287: 2145 FC AND DWORD PTR SS:[EBP-4],EAX
7C811288: 59 LEA EDI,DWORD PTR SS:[EBP-20]
7C811289: 8070 E0 MOV DWORD PTR SS:[EBP-24],20
7C81128A: C745 DC 20000000 REP STOS DWORD PTR ES:[EDI]
7C81128B: F3:AB MOU EDI,DWORD PTR SS:[EBP+10]
7C81128C: 8870 10 TEST EDI,EDI
7C81128D: 00 000000 JE kernel32._7C82FCDF
7C81128E: 2107 AND DWORD PTR DS:[EDI],EAX
7C81128F: 56 PUSH ESI
7C811290: 8875 08 MOU ESI,DWORD PTR SS:[EBP+8]
7C811291: 8D45 FC LEA EAX,DWORD PTR SS:[EBP-4]
7C811292: 59 PUSH EAX
7C811293: 8B 02 PUSH ECX

```

Registers (FPU)

ERX	7FFD9000
ECX	7FFD9000
EDX	7990E4F4 ntdll.KiFastSystemCallRet
EBX	7FFD9000
ESP	0012FF90
EBP	0012FFC0
ESI	FFFFFFF0
EDI	7C910208 ntdll.7C910208
EIP	7C811273 kernel32._7C811273
C 0	ES 0023 32bit 0xFFFFFFFF
P 1	CS 001B 32bit 0xFFFFFFFF
A 0	SS 0023 32bit 0xFFFFFFFF
Z 0	DS 0023 32bit 0xFFFFFFFF
S 0	FS 003B 32bit 7FFDF000(FFF)
T 0	GS 0000 NULL
D 0	LastErr ERROR_INVALID_HANDLE (00000006)
EFL	00000206 (NO,NB,A,NS,PE,GE,G)
ST0	empty -UNORM BCBC 010501B4 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 3 2 1 0 Err 0 S P U O Z D I  
FCW 023F Prec NEAR,S3 Mask 1 1 1 1 1 1 (GT)

Usciti dalla funzione il valore del registro ECX cambia nuovamente diventando in esadecimale 0a280105 in decimale è 170393861

**Registers (FPU)**

```

ERX 004290195
ECX 00000005
EDX 00000001
EBX 7FFD9000
ESP 0012FF94
EBP 0012FC00
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 0 (GT)**

**FCW 023F Prec NEAR,53 Mask 1 1 1 1 1 1 1 1**

**GetCommandLineA**

**Arg3 => 00000000**  
**Arg2 = 00000000**  
**Arg1 = 00000000**  
**Malware\_.00401128**

Possiamo notare come con l'istruzione and il valore di ECX cambia di nuovo in esadecimale è 00000005 e in decimale 5

**Registers (FPU)**

```

ERX 004290195
ECX 00000005
EDX 00000001
EBX 7FFD9000
ESP 0012FF94
EBP 0012FC00
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 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 00000206 (NO,NB,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 023F Prec NEAR,53 Mask 1 1 1 1 1 1 1 1**

**GetCommandLineA**

**Arg3 => 00000000**  
**Arg2 = 00000000**  
**Arg1 = 00000000**  
**Malware\_.00401128**

Infine possiamo vedere con l'istruzione add come il valore di ECX cambia di nuovo è diventa in esadecimale 00000500 in decimale 1280 con l'istruzione add si va ad aggiungere il valore contenuto in EDX ad ECX e salva il risultato in ECX

**Registers (FPU)**

```

ERX 004290195
ECX 00000500
EDX 00000001
EBX 7FFD9000
ESP 0012FF94
EBP 0012FC00
ESI FFFFFFFF
EDI 7C910208 ntdll.7C910208
EIP 004015BE Malware_.004015BE
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 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 023F Prec NEAR,53 Mask 1 1 1 1 1 1 1 1**

**GetCommandLineA**

**Arg3 => 00000000**  
**Arg2 = 00000000**  
**Arg1 = 00000000**  
**Malware\_.00401128**

Malware\_.<ModuleEntryPoint>+47

