Funzione "CreateProcessA" creata, viene caricato sullo stack il valore "cmd" dalla CommandLine.
Questo vuol dire che il programma avvia probabilmente il prompt dei comandi.

```
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>]
```

```
pProcessInfo

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

Ci spostiamo all'indirizzo 004015A3, il valore che ci interessa è quello di EDX.
In questo caso sono prima e dopo aver fatto lo "step-into".
Viene eseguita un'operazione XOR EDX,EDX.
Il risultato, come ci aspettiamo, diventa 0.

```
EAX 0A280105
ECX 7FFD8000
EDX 00000A28
EBX 7FFD8000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208
```

```
EAX 0A280105
ECX 7FFD8000
EDX 00000000
EBX 7FFD8000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208
```

Qui invece ci spostiamo all'indirizzo 004015AF, ci interessa il valore di ECX.
Prima e dopo aver fatto lo "step-into".
In questo caso però l'operazione è un AND.
AND ECX, 0FF (0FF in esadecimale è 255.

```
EAX 0A280105
ECX 0A280105
EDX 00000001
EBX 7FFD9000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208
```

```
EAX 0A280105
ECX 00000005
EDX 00000001
EBX 7FFD4000
ESP 0012FF94
EBP 0012FFC0
ESI FFFFFFFF
EDI 7C910208
```