Reversing: few words about a trivial code

Dear readers, how are you? During my classes and presentations, it is extremely common to hear students and professionals comparing different areas inside IT security, but honestly I think is neither possible nor feasible to do this “mental exercise”. Yesterday, I received one of this kind of message and, in the middle of the e-mail, I could read that “doubtless, hacking (pentest) is more difficult than reverse engineering and malware analysis”. I am not sure if it’s possible to state it. As an super easy educative example, I sent this code (I cleaned it a bit to make it clearer) below to my student and I asked him about two things: a) What’s the equivalent structure in C that the code is representing? b) How does it work?

```
.text:004028BC              mov    [ebp+var_C], eax
.text:004028BF              mov    ecx, [ebp+var_10]
.text:004028C2              movsx  edx, byte ptr [ecx]
.text:004028C5              mov    [ebp+var_14], edx
.text:004028C8              mov    eax, [ebp+var_14]
.text:004028CB              sub    eax, 64h
.text:004028CE              mov    [ebp+var_14], eax
.text:004028D1              cmp    [ebp+var_14], 0Fh ;
.text:004028D5              ja     short loc_402923 ;
.text:004028D7              mov    edx, [ebp+var_14]
.text:004028DA              xor    ecx, ecx
.text:004028DC              mov    cl, ds:byte_40293E[edx]
.text:004028E2              jmp    ds:off_40292A[ecx*4] ; switch jump
.text:004028E9 ; -----------------------------------------------------------------------
.text:004028E9 loc_4028E9:
.text:004028E9             ; CODE XREF: sub_402884+5Ej
.text:004028E9                                ; DATA XREF: .text:off_40292Ao
.text:004028E9              mov    eax, [ebp+var_C] ;
.text:004028EC              push   eax           ; char *
.text:004028ED              call   sub_401565
.text:004028F2              add    esp, 4
.text:004028F5              jmp    short loc_402923
.text:004028F7 ; -----------------------------------------------------------------------
.text:004028F7 loc_4028F7:
.text:004028F7             ; CODE XREF: sub_402884+5Ej
.text:004028F7                                ; DATA XREF: .text:off_40292Ao
.text:004028F7              mov    [ebp+var_4], 1  ;
.text:004028FE              jmp    short loc_402923 ;
.text:00402900 ; -----------------------------------------------------------------------
.text:00402900 loc_402900:
.text:00402900             ; CODE XREF: sub_402884+5Ej
.text:00402900
```

```
.text:00402900              mov    ecx, [ebp+var_C] ;
.text:00402903              push   ecx          ; char *
.text:00402904              call   sub_402813
.text:00402909              add    esp, 4
.text:0040290C              jmp    short loc_402923 ;
.text:0040290E ; ---------------------------------------------------------------------------
.text:0040290E
.text:0040290E loc_40290E:
.text:0040290E
.text:0040290E              mov    edx, [ebp+var_C] ;
.text:00402911              push   edx          ; char *
.text:00402912              call   sub_402851
.text:00402929              add    esp, 4
.text:0040291A              mov    eax, [ebp+arg_4]
.text:0040291D              mov    dword ptr [eax], 1
.text:00402923
.text:00402923 loc_402923:
.text:00402923
.text:00402923              mov    eax, [ebp+var_4] ; jumptable 004028E2 default case
.text:00402926              mov    esp, ebp
.text:00402928              pop    ebp
.text:00402929              retn
.text:00402929 sub_402884     endp
.text:00402929
.text:00402929 ; ---------------------------------------------------------------------------
.text:0040292A             dd offset loc_4028E9    ; DATA XREF: sub_402884+5Er
.text:0040292A             dd offset loc_4028F7    ; jump table for switch statement
.text:0040292A             dd offset loc_40290E
.text:0040292A             dd offset loc_402900
.text:0040292A             dd offset loc_402923
.text:0040293E             db     0,    4,    4,    4 ; DATA XREF: sub_402884+58r
.text:0040293E             db     4,    4,    4,    4 ; indirect table for switch statement
.text:0040293E             db     4,    4,    1,    4
.text:0040293E             db     4,    4,    2,    3
.text:0040294E
```

As I stated previously, the code above is trivial and, in a nutshell, although this code have been extracted from a malware, there is only reverse engineering here. Few comments follow:

- The represented structure is a simple “switch case” statement (it is easily identified by IDA Pro).
- There’re 16 possible cases (you should pay attention in the comparison at **0x004028D1** and remember that 0x0F is equal to 16).
- The variable which is defining the cases is var_16 (look at **0x004028D7**). It is will be loaded to edx and it will be acting as an index (more details below).
- A jump table (**0x0040292A**) is being used to represent the switch case statements.
- Looking at the jump table pointers (**0x0040293E**), we notice that there’re only five different indexes (0 to 4), so we have only five different statements in a nutshell. Therefore, the

instruction **mov cl, ds:byte_40293E[edx]** (at **0x004028DC**) servers as an index to jump table pointers. Depending on this index (0 to 15 – you remember that there are 16 possible cases in this example), the program chooses a pointer. For example, if the index is A(0x10) then the index in the jump table pointer is "1" (check this information by counting the values at 0x0040293E lines). Looking at jump table(**0x0040292A**), the second switch statement (remember, the range is from 0 to 4) is the address **0x004028F7** **(dd offset loc_4028F7).**

- Thus, the "switch jump" instruction **jmp ds:off_40292A[ecx*4]** at address **0x004028E2** finally jumps the code flow to the mentioned address above (**0x004028F7**).

As I said previously, this is an super easy and basic construction, but most time while I am analyzing malwares I see pieces of code like that. In fact, it is suitable to tell that malware analysis is much more difficult than a simple switch case statement. Sure, I could explain several kind of hooking, injections, hijacking , and so on, but I chose this example to prove to my student that is not possible to compare different areas before having a better knowledge about both them (in time: my student wasn't able to answer my questions at beginning of this write up).

Personally, my life is IT Security and I have a strong preference by malware analysis, so I am available to help you when necessary. If you want, I will be teaching few courses this year (more at http://alexandreborges.org/my-courses/) and I hope see you there.

Have a nice day.

Alexandre Borges.

(LinkedIn: http://www.linkedin.com/in/aleborges and twitter: @ale_sp_brazil).