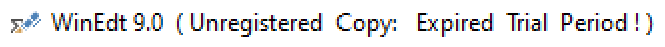
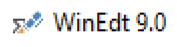
CS165 Project 2 Part 2 Report (Truncated)

Professor Zhiyun Qian
University of California, Riverside

By Sanchit Goel and Jiacheng Hou.

Overview

The main goal of this part was to remove the banner that appeared at the top of the screen after the trial period has ended (See figure 1).

Original	Patched
 WinEdt 9.0 (Unregistered Copy: Expired Trial Period !)	 WinEdt 9.0

(Figure 1)

Locating the correct instructions

The first step in this goal was to figure out where the instructions for displaying the banner text was located. We found the instructions by using the search feature within IDA to search for the string "trial" which appeared in the banner. Once the correct string was located, we used another IDA feature to find all references to the label "aUnregisteredCo" where the text was located. There was only one reference to the label which was within the sub-routine, "sub_76B074".

Determining the flow of execution

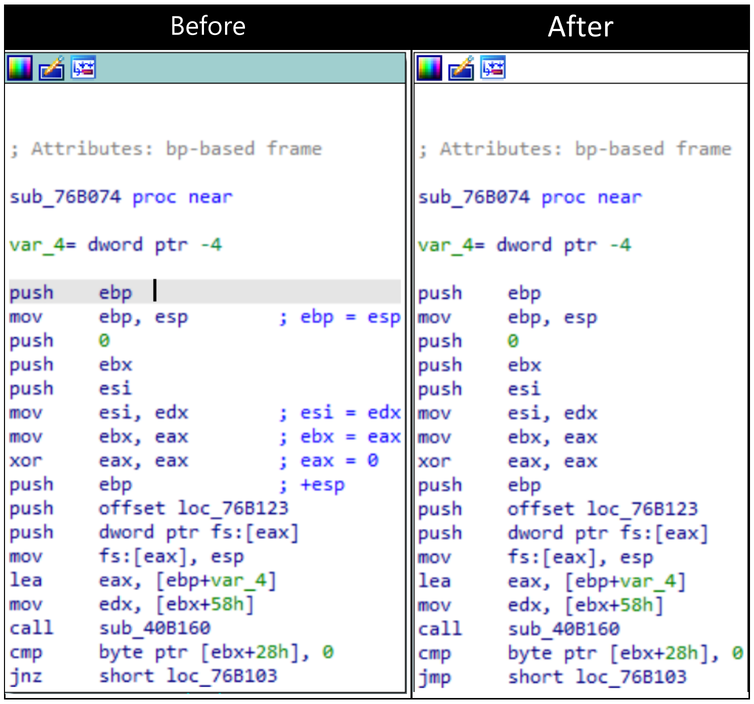
The next goal was to figure out the flow of the method and determine instructions that would be easy to manipulate to circumvent execution of other instructions we did not want. The primary goal was to identify an easy point to skip adding the banner. Sub-routine "sub_76B074" (which we will call "AddBanner" for easier reference) calls various sub-routines for the purpose of verifying whether user is registered or if they were using a trial and the trial has expired. At the end of the main block there is a jump instruction that jumps to the end of the sub-routine if a condition is not zero. If it is zero, then it jumps to several more comparison blocks each eventually leading up to applying a specific banner.

This is an assumption as to what the sub-routine does. One of the difficulties we faced was that the sub-routine invokes many other sub-routines which in turn invokes more sub-routines, so determining the full flow of execution would have taken an unreasonable amount of time. However, based on the comparison of the results of the sub-routines invocations, we made an educated guess as to the flow of execution.

Bypassing the banner addition

After learning how the execution of the sub-routine worked, it was clear that there was an easy way to circumvent the banner addition. All that needed to be done was to change the "jnz" instruction of the first sub-routine block to a forced "jmp" that would unconditionally jump to the end of the sub-routine without executing any of the logic that tested for which banner should be applied and applying the banner.

Thus, the only instruction changed was the operand of the "jnz" instruction at address 0x76B09C to the operand "jmp" (see figure 2).



Before:

```
; Attributes: bp-based frame

sub_76B074 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp        ; ebp = esp
push    0
push    ebx
push    esi
mov     esi, edx        ; esi = edx
mov     ebx, eax        ; ebx = eax
xor     eax, eax        ; eax = 0
push    ebp             ; +esp
push    offset loc_76B123
push    dword ptr fs:[eax]
mov     fs:[eax], esp
lea     eax, [ebp+var_4]
mov     edx, [ebx+58h]
call    sub_40B160
cmp     byte ptr [ebx+28h], 0
jnz     short loc_76B103
```

After:

```
; Attributes: bp-based frame

sub_76B074 proc near

var_4= dword ptr -4

push    ebp
mov     ebp, esp
push    0
push    ebx
push    esi
mov     esi, edx
mov     ebx, eax
xor     eax, eax
push    ebp
push    offset loc_76B123
push    dword ptr fs:[eax]
mov     fs:[eax], esp
lea     eax, [ebp+var_4]
mov     edx, [ebx+58h]
call    sub_40B160
cmp     byte ptr [ebx+28h], 0
jmp     short loc_76B103
```

(Figure 2)

Other difficulties faced

One of the difficulties we faced was locating the instructions regarding the pop-up that appears. We tried a similar methodology to the one used in finding the instructions for the banner, however no references to any of the text in the popup could be found. Thus, we focused our efforts on removing the banner instead of the popup.