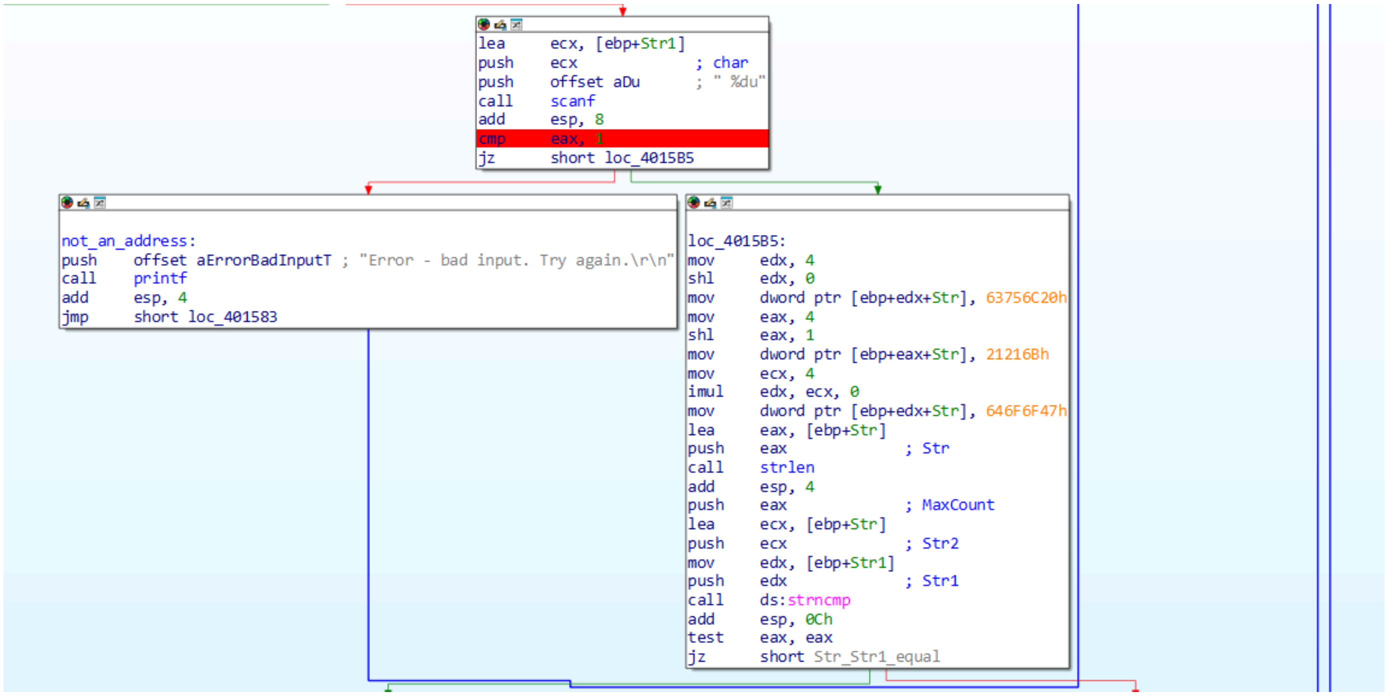
05 Dynamic Analysis

Author: Daniel Attali
Date: 06/06/2025

Wonderland.exe

Level 4

In this level, we see the following:



```
lea     ecx, [ebp+Str1]
push    ecx             ; char
push    offset aDu      ; " %du"
call    scanf
add     esp, 8
cmp     eax, 1
jz      short loc_4015B5
```

```
not_an_address:
push    offset aErrorBadInputT ; "Error - bad input. Try again.\r\n"
call    printf
add     esp, 4
jmp     short loc_401583
```

```
loc_4015B5:
mov     edx, 4
shl     edx, 0
mov     dword ptr [ebp+edx+Str], 63756C20h
mov     eax, 4
shl     eax, 1
mov     dword ptr [ebp+eax+Str], 21216Bh
mov     ecx, 4
imul    edx, ecx, 0
mov     dword ptr [ebp+edx+Str], 646F6F47h
lea     eax, [ebp+Str]
push    eax             ; Str
call    strlen
add     esp, 4
push    eax             ; MaxCount
lea     ecx, [ebp+Str]
push    ecx             ; Str2
mov     edx, [ebp+Str1]
push    edx             ; Str1
call    ds:strncmp
add     esp, 0Ch
test    eax, eax
jz      short Str_Str1_equal
```

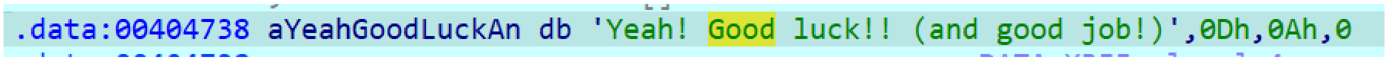
A `scanf` call that if input anything of than an `unsigned int` (`%du`) will cause a crash (using stack overflow).

And if input a number if will treat it as an address and try to compare it with the `Str` variable (which is a string that is initialized at runtime to be "Good luck!!")

So the first Idea was to give as an input the address of the `Str` variable (The runtime address so we need to use ida to debug it the address we try was `0x0017FEF8` converted to decimal `1703672`)

And the program output was `"Cheater ..."` so we understood that there was a check to see if `Str == Str1` so we had to give the program an address of a string that was `"Good luck!!"` while at the same time not being the same address of `Str` , the trick was to do it dynamically so we found the a random address and put the the string `"Good luck!!"` and gave the address as input

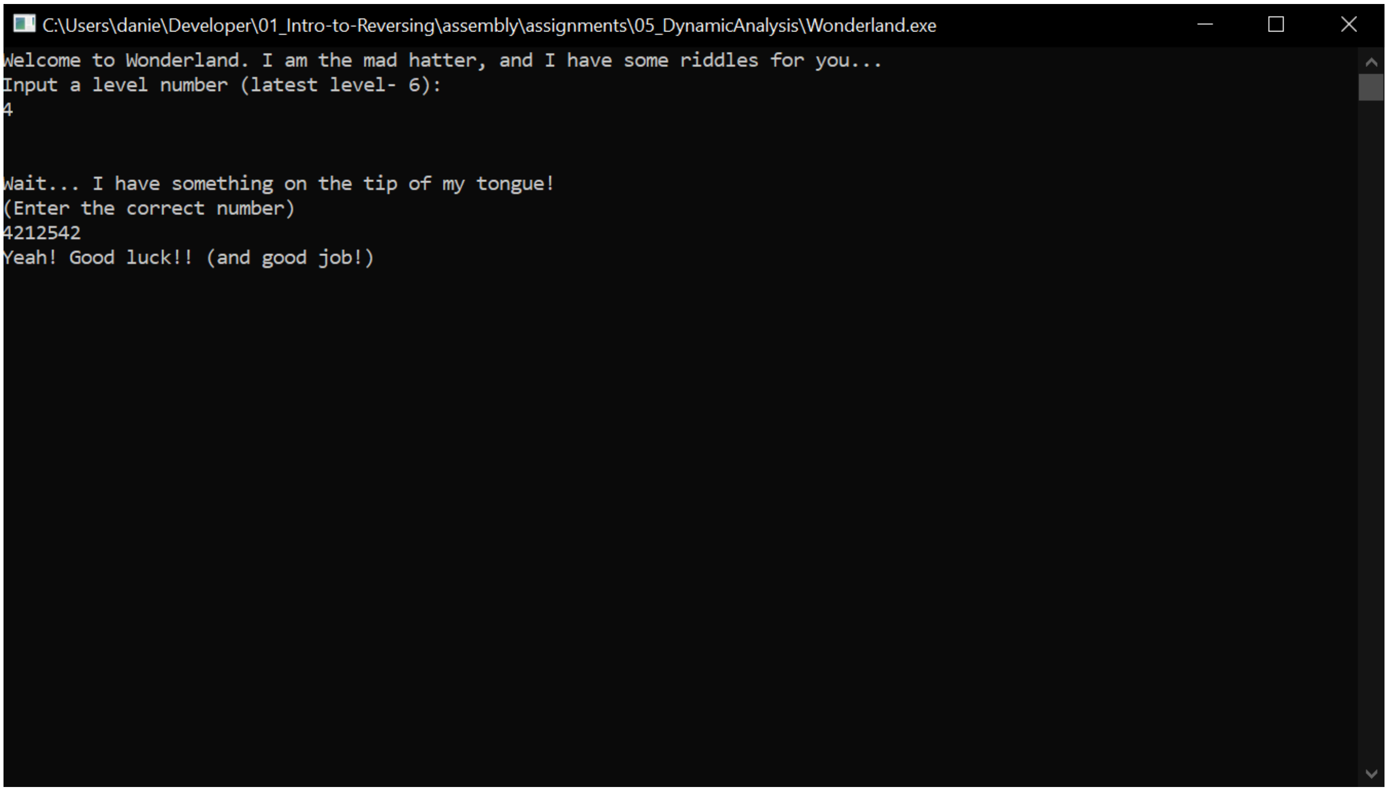
So the idea was to use the search functionality of ida to search for a `"Good luck!!"` string in the .exe and then we found the obvious answer



```
.data:00404738 aYeahGoodLuckAn db 'Yeah! Good luck!! (and good job!)',0Dh,0Ah,0
```

The address of the success string `0x00404738 + 0x6` = `4212542`

And when giving this address to the program we get the success screen



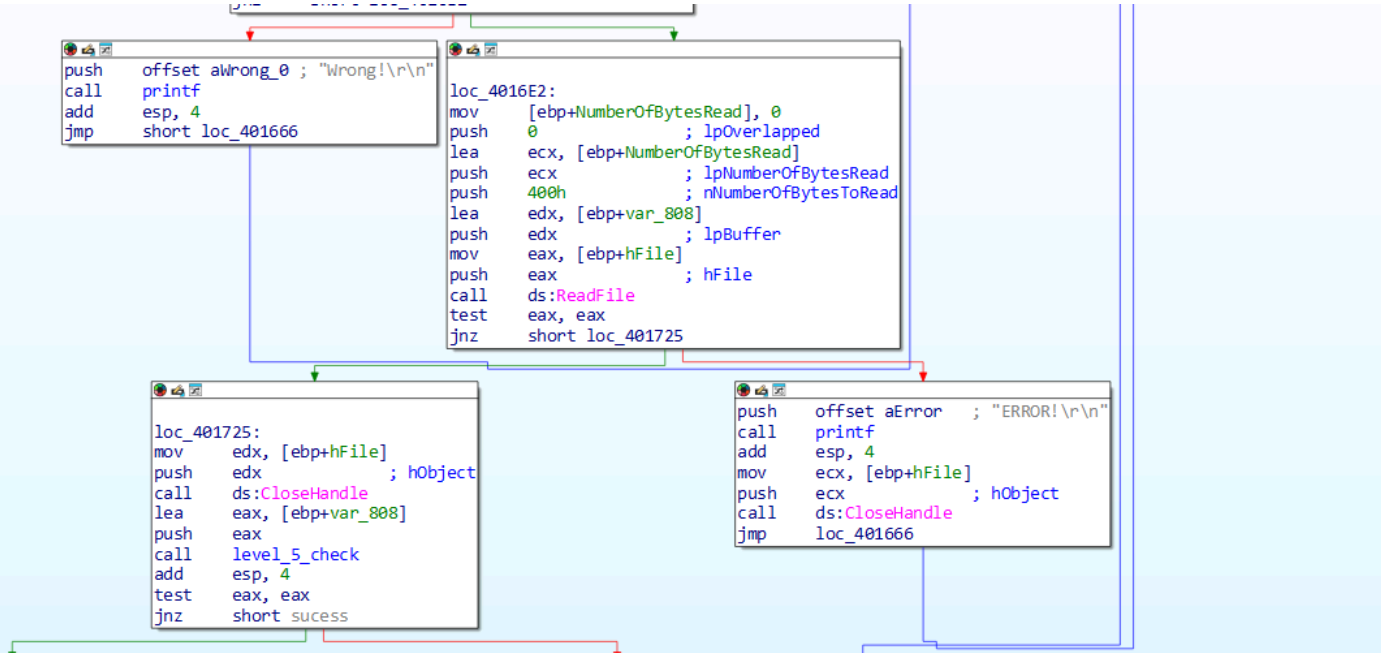
C:\Users\danie\Developer\01_Intro-to-Reversing\assembly\assignments\05_DynamicAnalysis\Wonderland.exe

```
Welcome to Wonderland. I am the mad hatter, and I have some riddles for you...
Input a level number (latest level- 6):
4


Wait... I have something on the tip of my tongue!
(Enter the correct number)
4212542
Yeah! Good luck!! (and good job!)
```

Level 5

In this level we see function call like `ReadFile` with options (only if already existed)



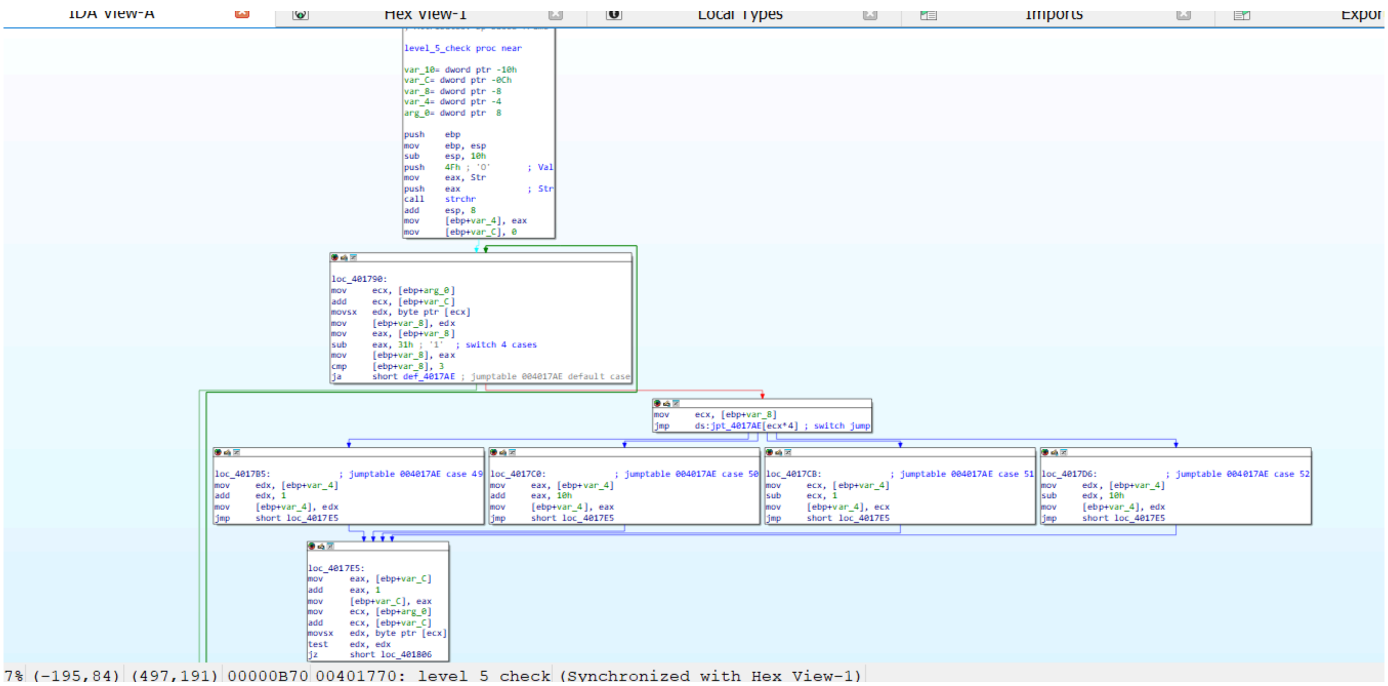
```
push    offset aWrong_0 ; "Wrong!\r\n"
call    printf
add     esp, 4
jmp     short loc_401666
```

```
loc_4016E2:
mov     [ebp+NumberOfBytesRead], 0
push    0               ; lpOverlapped
lea     ecx, [ebp+NumberOfBytesRead]
push    ecx             ; lpNumberOfBytesRead
push    400h            ; nNumberOfBytesToRead
lea     edx, [ebp+var_808]
push    edx             ; lpBuffer
mov     eax, [ebp+hFile]
push    eax             ; hFile
call    ds:ReadFile
test    eax, eax
jnz     short loc_401725
```

```
loc_401725:
mov     edx, [ebp+hFile]
push    edx             ; hObject
call    ds:CloseHandle
lea     eax, [ebp+var_808]
push    eax
call    level_5_check
add     esp, 4
test    eax, eax
jnz     short sucess
```

```
push    offset aError   ; "ERROR!\r\n"
call    printf
add     esp, 4
mov     ecx, [ebp+hFile]
push    ecx             ; hObject
call    ds:CloseHandle
jmp     loc_401666
```

So we try and input file name `test.txt` and it didn't work then we try to crate the file first and then input the name and we got through the first stage and now we needed to understand what the function, so we understood that the program read the file at tha path specified and read into a buffer and then it calls a function (I named it `level_5_check`) and if this function return anything other than `0` we solve the riddle



```
level_5_check proc near

var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 10h
push    4Fh ; 'O'       ; Val
mov     eax, Str
push    eax             ; Str
call    strchr
add     esp, 8
mov     [ebp+var_4], eax
mov     [ebp+var_C], 0
```

```
loc_401790:
mov     ecx, [ebp+arg_0]
add     ecx, [ebp+var_C]
movsx   edx, byte ptr [ecx]
mov     [ebp+var_8], edx
mov     eax, [ebp+var_8]
sub     eax, 31h ; '1'   ; switch 4 cases
mov     [ebp+var_8], eax
cmp     [ebp+var_8], 3
ja      short def_4017AE ; jumptable 004017AE default case
```

```
mov     ecx, [ebp+var_8]
jmp     ds:jpt_4017AE[ecx*4] ; switch jump
```

```
loc_4017B5:                 ; jumptable 004017AE case 49
mov     edx, [ebp+var_4]
add     edx, 1
mov     [ebp+var_4], edx
jmp     short loc_4017E5
```

```
loc_4017C0:                 ; jumptable 004017AE case 50
mov     eax, [ebp+var_4]
add     eax, 10h
mov     [ebp+var_4], eax
jmp     short loc_4017E5
```

```
loc_4017CB:                 ; jumptable 004017AE case 51
mov     ecx, [ebp+var_4]
sub     ecx, 1
mov     [ebp+var_4], ecx
jmp     short loc_4017E5
```

```
loc_4017D6:                 ; jumptable 004017AE case 52
mov     edx, [ebp+var_4]
sub     edx, 10h
mov     [ebp+var_4], edx
jmp     short loc_4017E5
```

```
loc_4017E5:
mov     eax, [ebp+var_C]
add     eax, 1
mov     [ebp+var_C], eax
mov     ecx, [ebp+arg_0]
add     ecx, [ebp+var_C]
movsx   edx, byte ptr [ecx]
test    edx, edx
jz      short loc_401806
```

7% (-195,84) (497,191) 00000B70 00401770: level 5 check (Synchronized with Hex View-1)

This function as a switch case form, so we took a look and found we use the `strchr` function with `79` which is `'O'` in ASCII and this function return the first position of a char in a string so we converted the Str string into a more readable form and got:

```
"###################O####...########.##...#.########.##.###.########....###...X##################"
```

And then we took at look at the switch statement and saw the following:

1. `case 1:` we move the sort of cursor `+1` so right
2. `case 2:` we do `+16` meaning we jump a line (so the we need to divide the line into chunks of 16)
3. `case 3:` we do `-1` so we move left
4. `case 4:` we do `-16` so we jump a line (up)

So after that we took the sting and made a 16x6 matrix and got the following

```
################
###O####...#####
###.##...#.#####
###.##.###.#####
###....###...X##
################
```

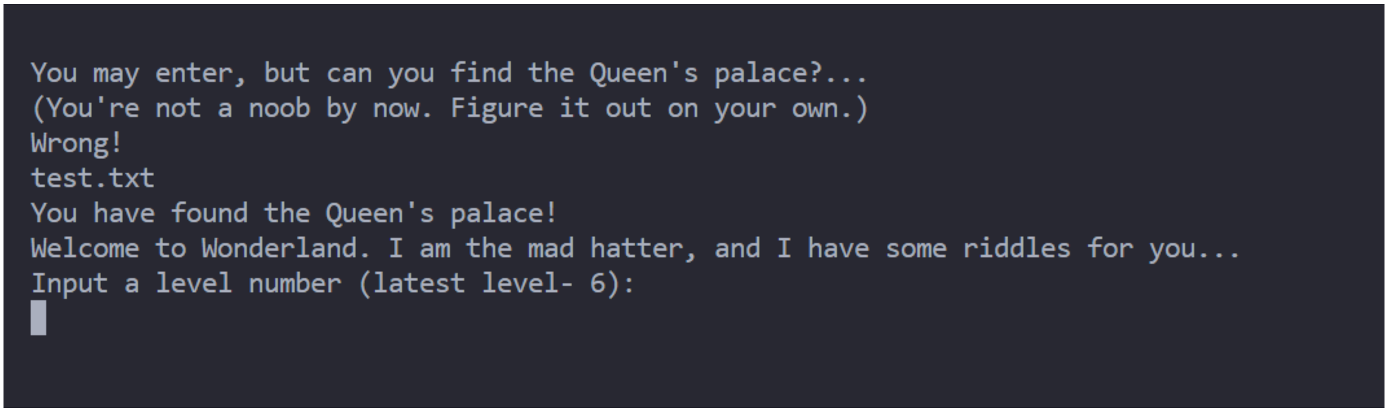
So we understood that this is a maze and we start at `'O'` and finish at `'X'` (X mark the spot) and the `'#'` are walls and the `'.'` is the path.

Solution is:

```
DDDRRRUURRURRDDDRRR
```

```
2221114411411222111
```

So we put into the `text.txt` file the solution and got the right answer



```
You may enter, but can you find the Queen's palace?...
(You're not a noob by now. Figure it out on your own.)
Wrong!
test.txt
You have found the Queen's palace!
Welcome to Wonderland. I am the mad hatter, and I have some riddles for you...
Input a level number (latest level- 6):
```