

Reloaded

Thursday, February 17, 2022 11:01 AM

Level 0

Looking through the strings we quickly find some flag looking strings

```
?? ??_L3@LZ340_is_002e_004b1045 XREF[1]
    s_L3@LZ340_is_002e_004b1045 ds
    33 76 ??_L3@LZ340_is_002e_004b1045
    9 6c 5a ??_L3@LZ340_is_002e_004b1045
    9 34 30.. ??_L3@LZ340_is_002e_004b1045

    s_That_was_easy...Bruh!!_004b1057 XREF[1]
    68 61 ??_L3@LZ340_is_002e_004b1057
    20 77 ??_L3@LZ340_is_002e_004b1057
    73 20.. ??_L3@LZ340_is_002e_004b1057

    s_Dont_worry_its_a_start_;_004b1070 XREF[1]
    6f 6e ??_L3@LZ340_is_002e_004b1070
    20 57 ??_L3@LZ340_is_002e_004b1070
    72 72 ??_L3@LZ340_is_002e_004b1070
```

Level 1

Checking the strings again, we find some interesting items

004b1000	libgcc_s_dw2-1.dll	"libgcc_s_dw2-1.dll"
004b1013	_register_frame_info	"_register_frame_info"
004b1029	__deregister_frame_info	"__deregister_frame_info"
004b1045	Thats ur lucky number !!!	"Thats ur lucky number ..."
004b105f	Try again	"Try again"
004b106a	Noob a day, pro for life	"NOOb a day, pro for life"

But nothing resembling a flag...

We find an interesting comparison that references some of these strings

```
0040141f 75 0e JNZ 148_0040142d
0040141f c7 04 24 MOV dword ptr [ESP] => local_1c,s_Thats_ur_lucky_num
    45 10 4b 00
0040142e e8 75 b1 CALL MSVCRT.DLL::printf
    int printf(char * _Fo
    01 00
0040142e eb 0d JMP LAB_0040143a

LAB_0040142d
0040142f c7 04 24 MOV dword ptr [ESP] => local_1c,s_Try_again_00a_m "Try again"
    45 10 4b 00
0040143f e8 5f b1 CALL MSVCRT.DLL::puts
    int puts(char * _Str)
    01 00
```

It looks like we want to get to the "That's ur lucky number"

We have a hardcoded comparison

```
B CMP dword ptr [EBP + param_1], 0x6ad
```

Converting that to a decimal

```
08 CMP dword ptr [EBP + param_1], 0x6ad
```

We validate that by running the program

```
PS C:\Users\ssali\Downloads> ./Level.exe ...
Thats ur lucky number !!!
PS C:\Users\ssali\Downloads>
```

Level 2

Looking for strings again, we find some useful stuff

004b1029	__deregister_frame_info	"__deregister_frame_info"
004b1045	Get Ready For L4 ;)	"Get Ready For L4 ;)"
004b105c	In order to advance you ...	"In order to advance you have to break your mindset"
004b108f	Wow Ur AtL3?	"Wow Ur AtL3?"
004b1090	#A cool concure... #A cool concure... lock_error"	"#A cool concure... #A cool concure... lock_error"

```
PS C:\Users\ssali\Downloads> .\'Level (1).exe'
Wow Ur At L3?
PS C:\Users\ssali\Downloads> .\'Level (1).exe' 1234
In order to advance you have to break your mindset
PS C:\Users\ssali\Downloads>
```

It looks like we want the "Get Ready For L4" string

We patch the comparison that skips over this instruction

```
00401446 e8 95 b1 CALL MSVCRT.DLL::strcmp
    int strcmp
    01 00
0040144b 85 c0 TEST EAX,EAX
0040144d 74 21 JZ LAB_00401470
0040144f c7 04 24 MOV dword ptr [ESP] => local_3c,s_Get_Ready_For_L4_...
    45 10 4b 00
00401456 e8 bd b1 CALL MSVCRT.DLL::puts
    int puts
```

and run the program again

```
PS C:\Users\ssali\Downloads> .\'Level (1).exe' 1234
Get Ready For L4 ;)
```

Level 3

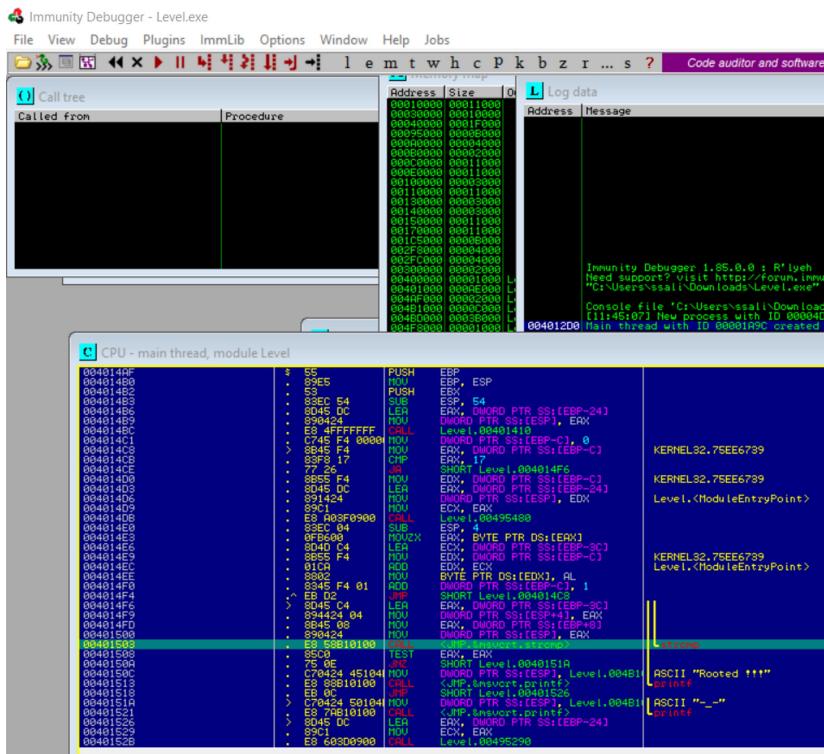
We start off looking at the strings again

004b1000	libgcc_s_dw2-1.dll	"libgcc_s_dw2-1.dll"
004b1045	Rooted !!!	"Rooted !!!"
004b1054	Enter the flag ::	"Enter the flag ::"
004b1080	__gnu_cxx::__concurrency_loc...	"__gnu_cxx::__concurrency_lock_error"
004b1094	"__gnu_cxx::__concurrency unl...	"__gnu_cxx::__concurrency unlock"

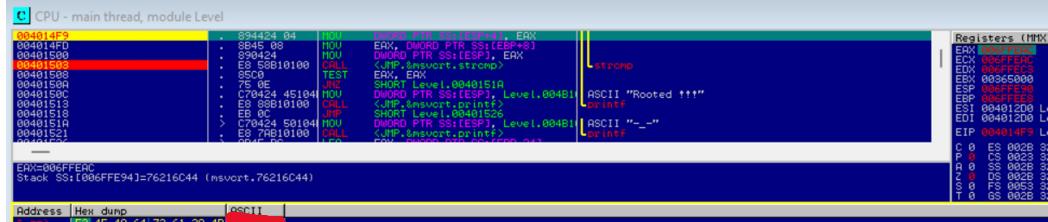
If we run the program in a debugger, we should be able to see the flag once it reaches this point

```
LAB_004014f6
004014f6 8d 45 c4 LEA EAX=>local_40,[EBP + -0x3c]
004014f9 89 44 24 04 MOV dword ptr [ESP + local_58],EAX
004014fd 8b 45 08 MOV EAX,dword ptr [EBP + param_1]
00401500 89 04 24 MOV dword ptr [ESP] => local_5c,EAX
00401503 e8 58 b1 CALL MSVCRT.DLL::strcmp
    int strcmp(char * _Str)
    01 00
00401508 85 c0 TEST EAX,EAX
```

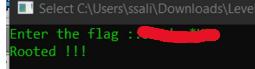
We throw the program into Immunity Debugger and find the strcmp again



We know that eax stores the password before the cmp

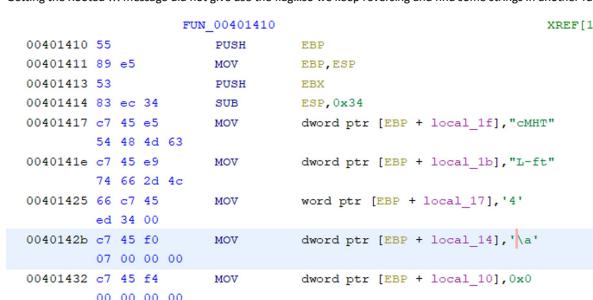


So we dump eax as it gets loaded to memory and find the password



Password: SC_____ (C_____)

Getting the Rooted !!! message did not give use the flag...so we keep reversing and find some strings in another function that look like a flag



Level 4

Looking through the strings we find a couple that stand out

004b2785	-+X01...	-+X012345...	ds
004b27a1	01234...	"0123456789"	ds
004b27bc	AKST	u"AKST"	ds
004b284c	basic ...	"basic string..."	ds

AKST:0123456789

Running the program we get the following

```
PS C:\Users\ssali\Downloads> .\Level_Final.exe  
Amcm‡QBu^YP+'ld‡V1pvY^BdR  
PS C:\Users\ssali\Downloads>
```

Amcm†QBu^YP+'ID†V1pvY^BdR

Lets find where this is getting printed

```

1 void __cdecl FUN_00401453(char *param_1)
2 {
3     size_t sVar1;
4     undefined4 uVar2;
5     int local_10;
6
7     sVar1 = strlen(param_1);
8     for (local_10 = 0; local_10 < (int)sVar1; local_10 = local_10 + 1) {
9         uVar2 = FUN_00401410(local_10);
10        if ((char)uVar2 == '\0') {
11            param_1[local_10] = param_1[local_10] ^ '7';
12        }
13        else {
14            param_1[local_10] = param_1[local_10] ^ (byte)local_10;
15        }
16    }
17    printf("%s", param_1);
18    return;
19 }

```

Looks like param_1 is enc_flag

and enc_flag is getting passed into this function

```
void __cdecl FUN_00401453(char *enc_flag)
```

Looking for cross-references

References to FUN_00401453 - 1 locations [CodeBrowser: RE/Level_Final.exe]

Edit Help

References to FUN_00401453 - 1 locations			
Location	Label	Code Unit	Context
00401532		CALL FUN_00401453	UNCONDITIONAL_CALL

```
undefined4 FUN_004014e6(void)
```

```
{
    undefined4 local_2a;
    undefined4 local_26;
    undefined4 local_22;
    undefined4 local_1e;
    undefined4 local_1a;
    undefined4 local_16;
    undefined2 local_12;

    FUN_0040c000();
    local_2a = 0x6e616c41;
    local_26 = 0x72755420;
    local_22 = 0x20676e69;
    local_1e = 0x20736157;
    local_1a = 0x65472061;
    local_16 = 0x7375696e;
    local_12 = 0x65;
    FUN_00401453((char *)&local_2a);
    return 0;
}
```

Renaming that function for clarity

```
FUN_0040c000();
local_2a = 0x6e616c41;
local_26 = 0x72755420;
local_22 = 0x20676e69;
local_1e = 0x20736157;
local_1a = 0x65472061;
local_16 = 0x7375696e;
local_12 = 0x65;
print_enc_flag((char *)&local_2a);
return 0;
```

Converting the hex stored at local_2a to strings, we get our flag

```
14ef e8 0c ab      CALL    FUN_0040c000
00 00
14f4 c7 44 24      MOV     dword ptr [ESP + local_2a], "████"
16 41 6c
61 6e
14fc c7 44 24      MOV     dword ptr [ESP + local_26], "████"
1a 20 54
75 72
1504 c7 44 24      MOV     dword ptr [ESP + local_22], "████"
1e 69 6e
67 20
150c c7 44 24      MOV     dword ptr [ESP + local_1e], "████"
22 57 61
73 20
1514 c7 44 24      MOV     dword ptr [ESP + local_1a], "████"
26 61 20
47 65
151c c7 44 24      MOV     dword ptr [ESP + local_16], "████"
2a 6e 69
75 73
1524 66 c7 44     MOV     word ptr [ESP + local_12], "█"
```

```
undefined4 FUN_004014e6(void)
{
    undefined4 local_2a;
    undefined4 local_26;
    undefined4 local_22;
    undefined4 local_1e;
    undefined4 local_1a;
    undefined4 local_16;
    undefined2 local_12;

    FUN_0040c000();
    local_2a = L"\x6e616c41";
    local_26 = L"\x72755420";
    local_22 = L"\x20676e69";
    local_1e = L"\x20736157";
    local_1a = L"\x65472061";
    local_16 = L"\x7375696e";
    local_12 = L"e";
    print_enc_flag((char *)&local_2a);
    return 0;
}
```