JBU-CTF 2020

Warm UP 2문제, Binary 2문제

조재현

INDEX

1

JBU-CTF 2020

1. 템플릿
2. 컨셉
3. Flag 양식

JBU-CTF 2020

1. 템플릿

PLAN-B 코로나19상황판 게시판 일정보기 일정관리 심심풀이

심심풀이

오늘 당신의 운세는

@송태현

1. FaceBook CTF 양식 쓰자 > 1번 입력

퀄리티 높음, 시간 꽤 걸림, 손 볼 곳 많음

2. 기존 양식 개선해서 쓰자 > 2번 입력

퀄리티 중간, 시간 덜 걸림, 디자인 자유도 낮음

3. 우리가 하나 만들자(템플릿있음) > 3번 입력

퀄리티 중상, 시간 약간 필요, 디자인 자유도 높음

1. 기본 컨셉(컨셉 통일X) > 1번 입력

문제마다의 각자 컨셉

2. 학교 컨셉 > 2번 입력

문제 난이도를 학년 별, 교수님 성함 포함(~문제를 내셨는데) 등

3. 기타 아이디어 > 3번 입력

아무 아이디어나 좋아요~

3. Flag 양식

scpCTF{ }

scpCTF{H3110_W0R1D}

scpCTF{N0_m0r3_Ch34T}

...

2

문제

1. Warm UP
2. Binary

Warm Up 1 - U N What I Mean?

Base 64 Decoding

SkJVRmxhZ3thcjNfeTB1X3IzNGR5fQ==

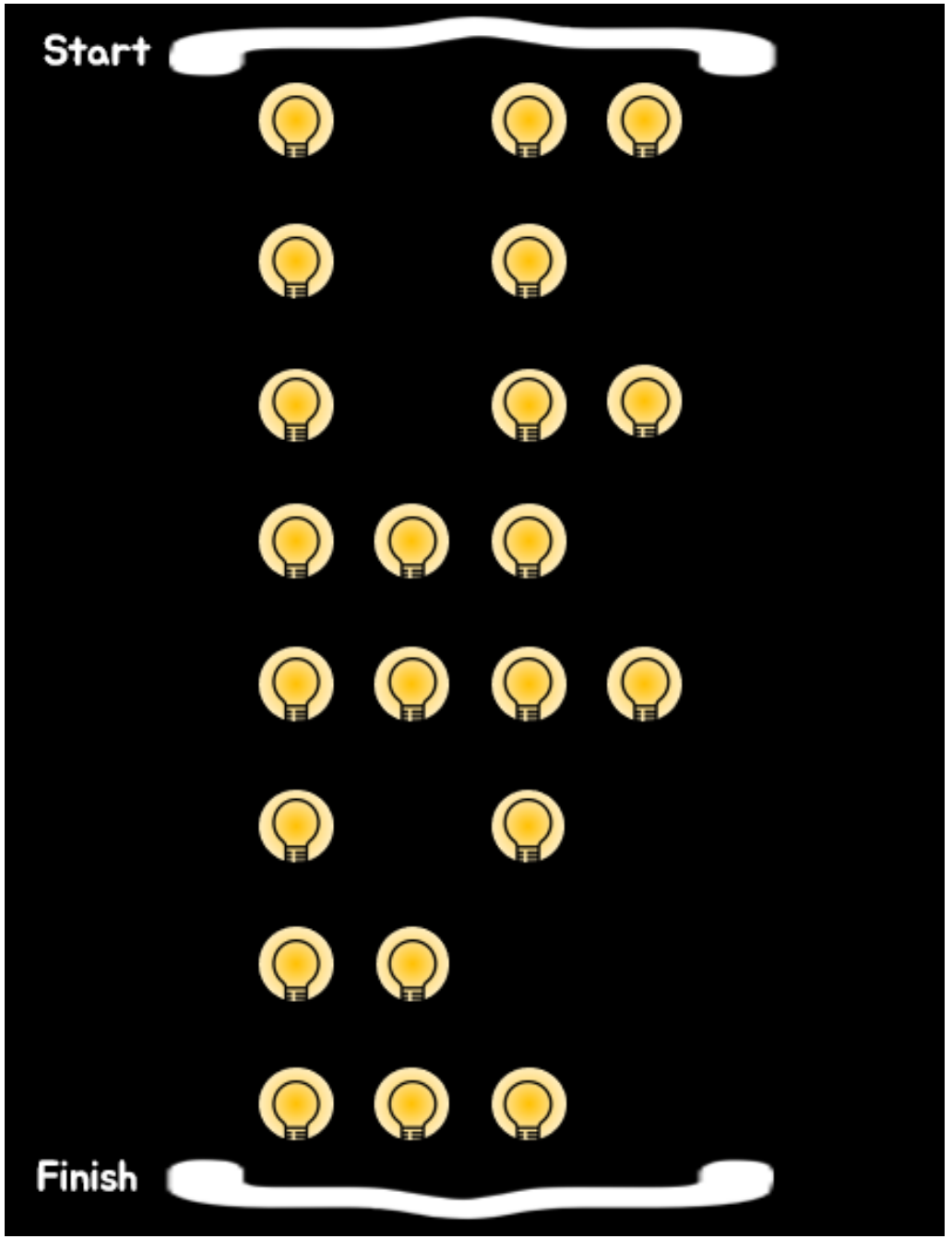
scpCTF{ar3_y0u_r34dy}

Warm Up 2 - 2 light

진수변환

{1011 1010 1011 1110
1111 1010 1100 1110}

scpCTF{BABEFACE}



Binary 1 - Access Code

- 분야 : Binary
- 세부 분야 : Reversing(ELF)
- 난이도 : 중
- 설명 : 인증 코드가 맞으면 플래그 줌

```
root@kali:~/ctf# ./auth_code_1
Hello Anonymous User! Please Input Your Code! ;)
Authdentication Code :
```

```
root@kali:~/ctf# ./auth_code_1
Hello Anonymous User! Please Input Your Code! ;)
Authdentication Code : CsOcDpE
Authdentication Success!
JBUFlag{R3verSing_4_Fun}
```

2. Binary - Access Code

소스 코드

```c
#include <stdio.h>
#include <string.h>

int main(void){
    char *key = "CsOcDpE";
    char input[20];
    printf("Hello Anonymous User! Please Input Your Code! ;) \n");
    printf("Authdentication Code : ");
    scanf("%s", input);
    if(strlen(input) >= 20){
        printf("Wrong Input Length!");
        return -1;
    }
    if(strcmp(key, input) == 0){
        printf("Authdentication Success! \n");
        printf("JBUFlag{R3verSing_4_Fun} \n\n");
    }else{
        printf("Authdentication Fail! \n\n");
    }
    return 0;
}
```

2 문제

2. Binary - Access Code

풀이 방법1

```
root@kali:~/ctf# file auth_code_1
auth_code_1: ELF 64-bit LSB shared object, x86-64
```

File 명령어
- 해당 파일이 어떤 파일인지 알려 줌

```
root@kali:~/ctf# strings auth_code_1
/lib64/ld-linux-x86-64.so.2
libc.so.6
__isoc99_scanf
puts
printf
strlen
__cxa_finalize
strcmp
__libc_start_main
GLIBC_2.7
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A\A]A^A_
CsOcDpE
Hello Anonymous User! Please Input Your Code! ;)
Authdentication Code :
Wrong Input Length!
Authdentication Success!
JBUFlag{R3verSing_4_Fun}
Authdentication Fail!
```

strings 명령어
- 해당 프로그램 속 문자열을 보여 줌

2. Binary - Access Code

풀이 방법2

```
root@kali:~/ctf# gdb auth_code_1
```

```
(gdb) b main
Breakpoint 1 at 0x1179
```

```
(gdb) disas main
Dump of assembler code for function main:
   0x0000000000001175 <+0>:     push   %rbp
   0x0000000000001176 <+1>:     mov    %rsp,%rbp
   0x0000000000001179 <+4>:     sub    $0x20,%rsp
   0x000000000000117d <+8>:     lea    0xe84(%rip),%rax
```

2 문제

2. Binary - Access Code

풀이 방법2

우리는 비교 부분만 보면 된다!

- test a b

a와 b를 AND연산하여
같으면 ZF=0, 다르면 ZF=1 설정

- jne [주소]

ZF=0이면 주소로 점프, 아니면 점프X

```
(gdb) disas main
Dump of assembler code for function main:
   0x0000000000001175 <+0>:     push   %rbp
   0x0000000000001176 <+1>:     mov    %rsp,%rbp
   0x0000000000001179 <+4>:     sub    $0x20,%rsp
   0x000000000000117d <+8>:     lea    0xe84(%rip),%rax        # 0x2008
   0x0000000000001184 <+15>:    mov    %rax,-0x8(%rbp)
   0x0000000000001188 <+19>:    lea    0xe81(%rip),%rdi        # 0x2010
   0x000000000000118f <+26>:    callq  0x1030 <puts@plt>
   0x0000000000001194 <+31>:    lea    0xea7(%rip),%rdi        # 0x2042
   0x000000000000119b <+38>:    mov    $0x0,%eax
   0x00000000000011a0 <+43>:    callq  0x1050 <printf@plt>
   0x00000000000011a5 <+48>:    lea    -0x20(%rbp),%rax
   0x00000000000011a9 <+52>:    mov    %rax,%rsi
   0x00000000000011ac <+55>:    lea    0xea7(%rip),%rdi        # 0x205a
   0x00000000000011b3 <+62>:    mov    $0x0,%eax
   0x00000000000011b8 <+67>:    callq  0x1070 <__isoc99_scanf@plt>
   0x00000000000011bd <+72>:    lea    -0x20(%rbp),%rax
   0x00000000000011c1 <+76>:    mov    %rax,%rdi
   0x00000000000011c4 <+79>:    callq  0x1040 <strlen@plt>
   0x00000000000011c9 <+84>:    cmp    $0x13,%rax
   0x00000000000011cd <+88>:    jbe    0x11e7 <main+114>
   0x00000000000011cf <+90>:    lea    0xe87(%rip),%rdi        # 0x205d
   0x00000000000011d6 <+97>:    mov    $0x0,%eax
   0x00000000000011db <+102>:   callq  0x1050 <printf@plt>
   0x00000000000011e0 <+107>:   mov    $0xffffffff,%eax
   0x00000000000011e5 <+112>:   jmp    0x1229 <main+180>
   0x00000000000011e7 <+114>:   lea    -0x20(%rbp),%rdx
   0x00000000000011eb <+118>:   mov    -0x8(%rbp),%rax
   0x00000000000011ef <+122>:   mov    %rdx,%rsi
   0x00000000000011f2 <+125>:   mov    %rax,%rdi
   0x00000000000011f5 <+128>:   callq  0x1060 <strcmp@plt>
   0x00000000000011fa <+133>:   test   %eax,%eax
   0x00000000000011fc <+135>:   jne    0x1218 <main+163>
   0x00000000000011fe <+137>:   lea    0xe6c(%rip),%rdi        # 0x2071
   0x0000000000001205 <+144>:   callq  0x1030 <puts@plt>
   0x000000000000120a <+149>:   lea    0xe7a(%rip),%rdi        # 0x208b
   0x0000000000001211 <+156>:   callq  0x1030 <puts@plt>
   0x0000000000001216 <+161>:   jmp    0x1224 <main+175>
   0x0000000000001218 <+163>:   lea    0xe87(%rip),%rdi        # 0x20a6
   0x000000000000121f <+170>:   callq  0x1030 <puts@plt>
   0x0000000000001224 <+175>:   mov    $0x0,%eax
   0x0000000000001229 <+180>:   leaveq
   0x000000000000122a <+181>:   retq
```

2. Binary - Access Code

풀이 방법2

```
(gdb) b *0x0000000000011b8
Breakpoint 2 at 0x11b8
(gdb) b *0x0000000000011fa
Breakpoint 3 at 0x11fa
```

입력 받는 부분에 breakpoint!

test하는 부분에 breakpoint!

하나씩 실행!

```
(gdb) r
Starting program: /root/ctf/auth_code_1

Breakpoint 1, 0x000055555555179 in main ()
(gdb) c
Continuing.
Hello Anonymous User! Please Input Your Code! ;)
Authdentication Code : ewrjiwerjiwerj

Breakpoint 5, 0x00005555555551fa in main ()
(gdb)
```

2. Binary - Access Code

풀이 방법2

```
⇒ 0x000055555555551fa <+133>:   test   %eax,%eax
  0x000055555555551fc <+135>:   jne    0x555555555218 <main+163>
  0x000055555555551fe <+137>:   lea    0xe6c(%rip),%rdi        # 0x555555556071
  0x0000555555555205 <+144>:   callq  0x555555555030 <puts@plt>
  0x000055555555520a <+149>:   lea    0xe7a(%rip),%rdi        # 0x55555555608b
  0x0000555555555211 <+156>:   callq  0x555555555030 <puts@plt>
  0x0000555555555216 <+161>:   jmp    0x555555555224 <main+175>
  0x0000555555555218 <+163>:   lea    0xe87(%rip),%rdi        # 0x5555555560a6
  0x000055555555521f <+170>:   callq  0x555555555030 <puts@plt>
  0x0000555555555224 <+175>:   mov    $0x0,%eax
  0x0000555555555229 <+180>:   leaveq
  0x000055555555522a <+181>:   retq
End of assembler dump.
(gdb) p $eax
$1 = -34
```

- p [변수]

변수의 value를 보여줘!

- set [변수] = [값]

변수의 value를 설정!

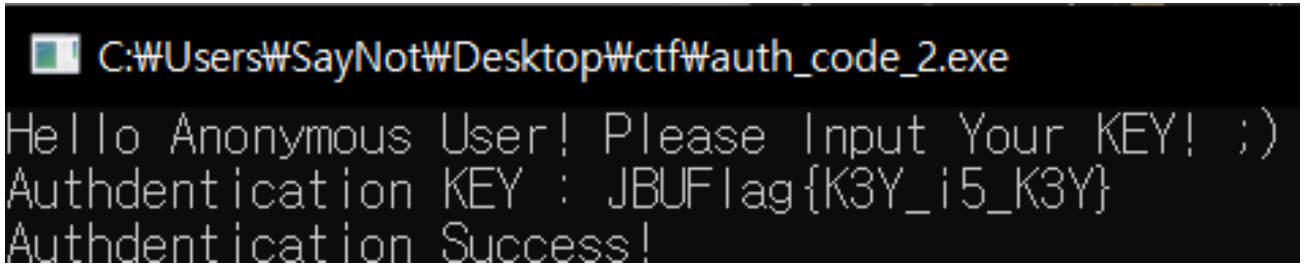
```
(gdb) set $eax = 0
(gdb) c
Continuing.
Authdentication Success!
JBUFlag{R3verSing_4_Fun}

[Inferior 1 (process 2145) exited normally]
```

앞서서 이상한 키를 입력했지만
Success 부분으로 jmp 됨

Binary 2 - Key is Key

- 분야 : Binary
- 세부 분야 : Reversing(EXE)
- 난이도 : 중
- 설명 : 인증 키 자체가 플래그



2. Binary - Access Code

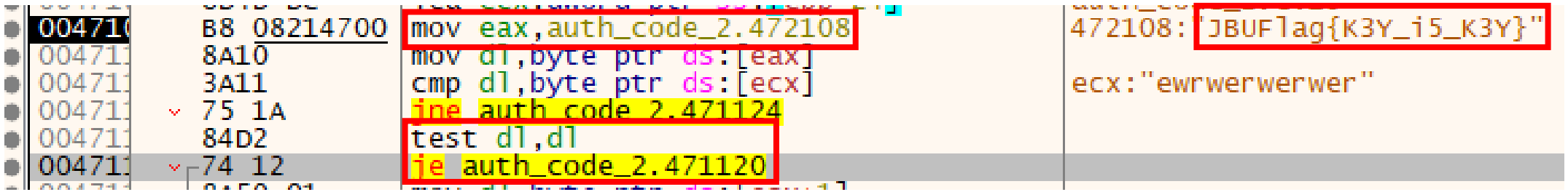
소스 코드

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void) {
    char *key = "JBUFlag{K3Y_i5_K3Y}";
    char input[30];
    printf("Hello Anonymous User! Please Input Your KEY! ;) \n");
    printf("Authdentication KEY : ");
    scanf_s("%s", input, 30);
    if (strlen(input) >= 30) {
        printf("Wrong Input Length!");
        return -1;
    }
    if (strcmp(key, input) == 0) {
        printf("Authdentication Success! \n");
    }
    else {
        printf("Authdentication Fail! \n\n");
    }
    system("pause");
    return 0;
}
```

2. Binary - Key is Key

풀이 방법



1. OllyDBG로 위에 과정처럼 진행!

2. F8로 스텝오버하며 실행과정 분석

3. test 전에 변수를 넘기는 과정에서 지역변수 속 키 노출

감사합니다 :)