[2015 Codegate Prequal - Bookstore]

프로그램을 실행시키면 먼저 로그인을 하라고 한다.

해당 부분을 IDA로 열어보자.

```
int login_1495()
{
  int result; // eax@3
  char id[4]; // [sp+1Dh] [bp-48h]@1
  char pass[4]; // [sp+28h] [bp-40h]@1
  char buf; // [sp+34h] [bp-34h]@1
  char s2; // [sp+48h] [bp-20h]@1
  int v5; // [sp+5Ch] [bp-Ch]@1

  v5 = *MK_FP(__GS__, 20);
  read_file_8DB("/home/bookstore/famous_saying.txt");
  strcpy(id, "helloadmin");
  strcpy(pass, "iulover!@#$");
  puts("== Bookstore Management Application ==");
  puts("== Login");
  printf("== Input Your ID : ");
  read(0, &buf, 0x14u);
  printf("== Input Your PASSWORD : ");
  read(0, &s2, 0x14u);
  puts("========================================Wn");
  result = !strncmp(id, &buf, 0xAu) && !strncmp(pass, &s2, 0xBu);
  if ( *MK_FP(__GS__, 20) != v5 )
    sub_565A9B60();
  return result;
}
```

id : helloadmin

pw : iulover!@#$

로그인을 하고나면 여러 가지 메뉴를 보여준다.

```
int print_menu()
{
  puts("== Bookstore Management Application ==");
  puts("1. Add new item");
  puts("2. Modify the item");
  puts("3. View item's information.");
  puts("4. Show item list");
  puts("0. Exit");
  return puts("========================================");
}
```

이 프로그램은 책을 추가하고 여러 가지 수정을 할 수 있고 책의 리스트와 자세한 정보까지 출력해준다. 먼저 가장 만만한 오버플로우가 일어나는 부분이 있는지 찾아봤는데 한 군데도 찾지 못했다. 하지만 몇가지 의심스러운 부분을

찾아냈다.

먼저 책을 추가하는 메뉴에서 발견한 것이다.

```
print_D = PRINT_DESCRIPTION_9AD;
if ( _type_ == 1 )
{
  puts("Input Max Download : ");
  __isoc99_scanf("%d", &max);
  v16 = v4;
}
else
{
  v14 = 1;
  print_N = PRINT_PRODUCT_NAME_9DA;
}
```

책의 정보는 하나의 구조체에 저장되는데 이때 책의 타입에 따라서 사용하는 공간이 조금 다르다.(구조체는 같음)
그런데 위에서 보다시피 description과 name을 출력해주는 함수를 구조체에 함수포인터 형식으로 집어넣는다. 이 부분을 릭 하거나 덮어써서 eip를 잡으면 될 것 같다.

다음은 여러 부분에서 공통적으로 나타나는 부분이다.

```
qmemcpy(a1, &_num_, 0x164u);
```

책을 생성하거나 정보를 수정하는 등의 함수에서는 입력값을 구조체에 직접 넣는 것이 아니라 지역변수에 넣어놓은 뒤 스택을 그대로 복사해 구조체에 집어넣는다.
이걸 어떻게 이용할까 고민하던 중에 다음 부분을 발견했다.

```
v13 = PRINT_DESCRIPTION_9AD;
puts("Set Free Shipping? (1 : free shipping | 0 : not) ");
__isoc99_scanf("%d", &v6);
if ( v6 != 1 && v6 )
{
  puts("Wrong Value.. and set free-shipping");
  free_ship = 1;
}
else
{
  free_ship = v6;
}
if ( free_ship == 1 )                          // free shipping -> 0
  v15 = PRINT_PRODUCT_NAME_9DA;
```

book 타입의 책의 Info를 수정하는 메뉴에서 free shipping 이 0으로 설정되어 있으면 함수포인터의 자리에 함수포인터를 넣어주지 않는다. 하지만 이 함

수 역시 스택을 구조체에 복사하는 방식으로 입력을 한다. 즉 스택상의 임의의 값이 함수포인터로 쓰일 수 있다는 것이다.

따라서 스택에 내가 원하는 값을 남길 수만 있다면 함수 포인터 자리에 내가 원하는 값을 써넣을 수 있을 것이다.

함수 포인터 자리를 덮어쓰려면 main->modify->modify_info 부분에서 덮어써야 한다. modify_info가 사용하는 스택에 원하는 값을 써넣어야 하므로 modify부분에서 호출되는 다른 함수를 살펴보자.

```
int __cdecl NEW_DESCRIPTION_FB9(int a1)
{
  int result; // eax@1
  char s; // [sp+14h] [bp-BC4h]@1
  int v3; // [sp+BCCh] [bp-Ch]@1

  v3 = *MK_FP(__GS__, 20);
  puts("Input new description");
  memset(&s, 0, 0xBB8u);
  read(0, &s, 0xBB8u);
  memset((void *)(a1 + 56), 0, 0x12Cu);
  strncpy((char *)(a1 + 56), &s, 0x12Cu);
  puts("Complete!");
  result = *MK_FP(__GS__, 20) ^ v3;
  if ( *MK_FP(__GS__, 20) != v3 )
    sub_565A9B60();
  return result;
}
```

description을 수정하는 함수에서 스택에 엄청나게 많은 값을 써넣을 수 있다는 것을 볼 수 있다. 이때 우리가 덮어쓰고자 하는 곳은 [bp-0x154] 부분이므로 0xBC4 - 0x154 = 0xA70 만큼의 더미값을 넣어주고 4바이트만큼의 함수포인터를 넣어주면 해당 부분을 호출할 때 eip를 잡을 수 있게 된다.

```
int __cdecl read_file_8DB(const char *a1)
{
  int result; // eax@3
  FILE *stream; // [sp+28h] [bp-420h]@1
  char ptr; // [sp+2Ch] [bp-41Ch]@1
  int v4; // [sp+42Ch] [bp-1Ch]@1

  v4 = *MK_FP(__GS__, 20);
  memset(&ptr, 0, 0x400u);
  stream = fopen(a1, (const char *)&unk_565A9B94);
  if ( stream )
  {
    fread(&ptr, 0x400u, 1u, stream);
    printf((const char *)&unk_565A9B97, &ptr);
  }
  fclose(stream);
  result = *MK_FP(__GS__, 20) ^ v4;
  if ( *MK_FP(__GS__, 20) != v4 )
    sub_565A9B60();
  return result;
}
```

또한 함수들을 살펴보면 파일을 열어서 해당 내용을 출력까지 해주는 함수가 들어있는 것을 볼 수 있다. 읽어올 파일의 이름은 인자로 넘겨주면 된다.

```
else if ( a11 )
{
  print_N(&name);
}
```

우리가 덮어쓴 함수포인터를 호출할 때 넘겨지는 인자는 name이므로 책의 이름에 열고자 하는 파일의 이름을 넣어놓으면 된다.

readfile()함수의 주소로 함수포인터를 덮어쓰기 위해서는 메모리 릭을 통해 베이스 주소를 구해야 하는데 이건 구조체의 구조를 조금만 살펴보면 어렵지 않게 할 수 있다.

```
int _name_; // [sp+170h] [bp-178h]@1 name[20]
char v10; // [sp+183h] [bp-165h]@1
int _price_; // [sp+184h] [bp-164h]@5
int _stock_; // [sp+188h] [bp-160h]@5
int (__cdecl *print_D)(int); // [sp+18Ch] [bp-15Ch]@5
```

name에 20바이트를 전부 써넣고 price와 stock에 4바이트 정수를 써넣는다면 description을 출력해수는 함수의 포인터를 릭 할 수 있다.

[exploit]
[book 생성] -> [info 수정] -> [메모리 릭] -> [description 수정] -> [info 수정] -> [view items info] -> exploit!!