3S CTF

PWN-lucky_cook

SWING 32 기 한재희

우선 압축을 풀고 파일을 우분투로 옮겨서 checksec 로 확인해줬다.

```
(gdb) file lucky_cook
BFD: warning: /home/wogml/lucky_cook: unsupported GNU_PROPERTY_TYPE (5) type:
xc0008002
Reading symbols from lucky_cook...(no debugging symbols found)...done.
(gdb) disas main
Dump of assembler code for function main:
   0x0000000000401430 <+0>:     endbr64
   0x0000000000401434 <+4>:     push   %rbp
   0x0000000000401435 <+5>:     mov    %rsp,%rbp
   0x0000000000401438 <+8>:     sub    $0x120,%rsp
   0x000000000040143f <+15>:    mov    %fs:0x28,%rax
   0x0000000000401448 <+24>:    mov    %rax,-0x8(%rbp)
   0x000000000040144c <+28>:    xor    %eax,%eax
   0x000000000040144e <+30>:    lea    0xbb3(%rip),%rax        # 0x402008
   0x0000000000401455 <+37>:    mov    %rax,%rdi
   0x0000000000401458 <+40>:    callq  0x4010e0 <puts@plt>
   0x000000000040145d <+45>:    mov    $0x0,%eax
   0x0000000000401462 <+50>:    callq  0x401305 <get_rand>
   0x0000000000401467 <+55>:    mov    %al,-0x111(%rbp)
   0x000000000040146d <+61>:    cmpb   $0x4c,-0x111(%rbp)
   0x0000000000401474 <+68>:    jne    0x401499 <main+105>
   0x0000000000401476 <+70>:    lea    0xba5(%rip),%rax        # 0x402022
   0x000000000040147d <+77>:    mov    %rax,%rdi
   0x0000000000401480 <+80>:    mov    $0x0,%eax
   0x0000000000401485 <+85>:    callq  0x401100 <printf@plt>
   0x000000000040148a <+90>:    mov    $0x0,%eax
   0x000000000040148f <+95>:    callq  0x401269 <lucky>
   0x0000000000401494 <+100>:   jmpq   0x401571 <main+321>
```

나는 그냥 gdb 여서 이걸로는 확인할 수 있는 방법이 없을까 하고 찾다가 file 파일명 하면 설정이 되고 disas 함수명을 입력하면 특정 함수의 어셈블리 코드가 출력된다고 해서 입력했더니 출력되었다.

근데 뭔가 봐도 이해가 안되고 그래서 disas lucky 입력해봤더니

```
(gdb) disas lucky
Dump of assembler code for function lucky:
   0x0000000000401269 <+0>:     endbr64
   0x000000000040126d <+4>:     push   %rbp
   0x000000000040126e <+5>:     mov    %rsp,%rbp
   0x0000000000401271 <+8>:     sub    $0x50,%rsp
   0x0000000000401275 <+12>:    mov    %fs:0x28,%rax
   0x000000000040127e <+21>:    mov    %rax,-0x8(%rbp)
   0x0000000000401282 <+25>:    xor    %eax,%eax
   0x0000000000401284 <+27>:    lea    -0x50(%rbp),%rax
   0x0000000000401288 <+31>:    mov    $0x40,%edx
   0x000000000040128d <+36>:    mov    %rax,%rsi
   0x0000000000401290 <+39>:    mov    $0x0,%edi
   0x0000000000401295 <+44>:    callq  0x401120 <read@plt>
   0x000000000040129a <+49>:    lea    -0x50(%rbp),%rax
   0x000000000040129e <+53>:    mov    %rax,%rdi
   0x00000000004012a1 <+56>:    mov    $0x0,%eax
   0x00000000004012a6 <+61>:    callq  0x401100 <printf@plt>
   0x00000000004012ab <+66>:    lea    -0x50(%rbp),%rax
   0x00000000004012af <+70>:    mov    %rax,%rdi
   0x00000000004012b2 <+73>:    callq  0x401160 <atoi@plt>
   0x00000000004012b7 <+78>:    cmp    $0x1,%eax
   0x00000000004012ba <+81>:    je     0x4012d4 <lucky+107>
   0x00000000004012bc <+83>:    lea    -0x50(%rbp),%rax
   0x00000000004012c0 <+87>:    mov    $0x40,%edx
   0x00000000004012c5 <+92>:    mov    $0x0,%esi
```

또 나온걸 봐서 lucky 라는 함수도 있는 거 같다.

구글링을 하다가

: x/[범위][출력 형식][범위의 단위 (기본 4byte)][메모리 주소나 함수명]

출력 형식

- t: 2진수
- o: 8진수
- d: 10진수
- u: 부호 없는 10진수
- x: 16진수 (주소를 보기 위해 가장 많이 사용)
- c: 문자형 출력(크기가 4바이트 이상이면 처음 1바이트 출력)
- f: 부동 소수점 값 형식으로 출력
- a: 가장 가까운 심볼의 오프셋 출력

범위의 단위

- b: 1byte
- h: 2byte
- w: 4byte
- g: 8byte

ex) (gdb)x/32bx main
:: 메인함수 시작지점부터 b(1바이트)를 32번 출력하는데, x(16진수)로 출력하라.

이런 명령어를 확인했는데 예시에 나와있는 명령어를 입력해보니

```
(gdb) x/32bx main
0x401430 <main>:        0xf3    0x0f    0x1e    0xfa    0x55    0x48    0x89    0
xe5
0x401438 <main+8>:      0x48    0x81    0xec    0x20    0x01    0x00    0x00    0
x64
0x401440 <main+16>:     0x48    0x8b    0x04    0x25    0x28    0x00    0x00    0
x00
0x401448 <main+24>:     0x48    0x89    0x45    0xf8    0x31    0xc0    0x48    0
x8d
```

이렇게 나왔다. Pwngdb 를 설치하지 못해서 슬펐는데 그냥 gdb 로도 뭔가 출력되어서 신기하고 더 공부 해봐야겠다.