Pwndbg와 함께하는

Reversing_Wargame

2학년 이지훈

목차

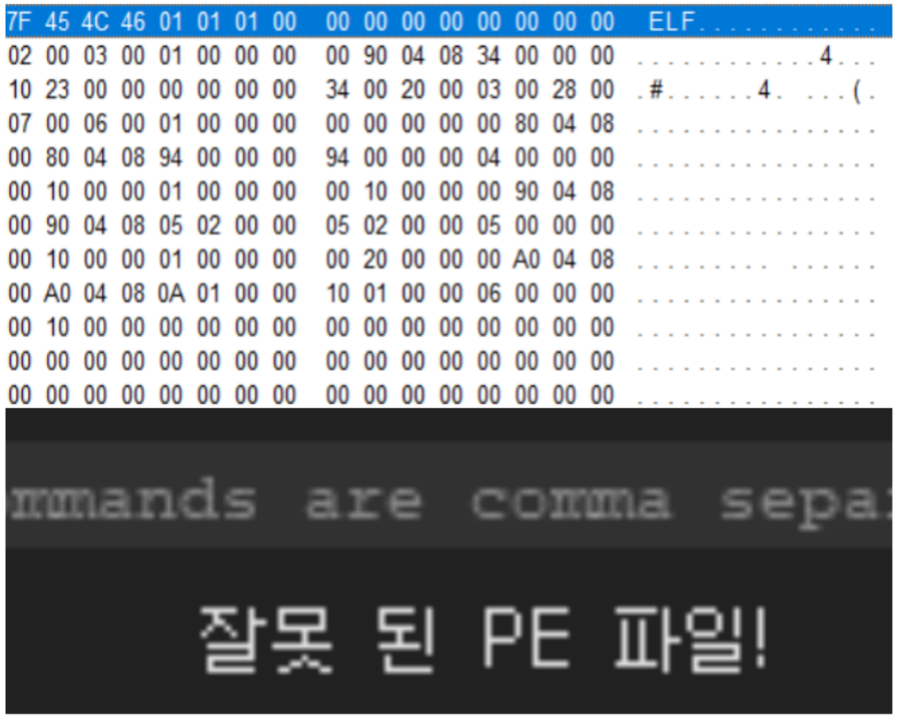
Reversing_with

01 | PwnDbg?

GDB에서 실행된 프로그램의 주요 메모리들의 상태를 가독성 있게 표현해주는 인터페이스 플러그인

- GDB (GNU's DeBugger) : 리눅스의 대표 Debugger 중 하나
- 디버거를 이용하여 프로그램의 실행 과정을 자세히 관찰하려면 컴퓨터의 각종 메모리를 한눈에 파악할 수 있는 것이 좋다.

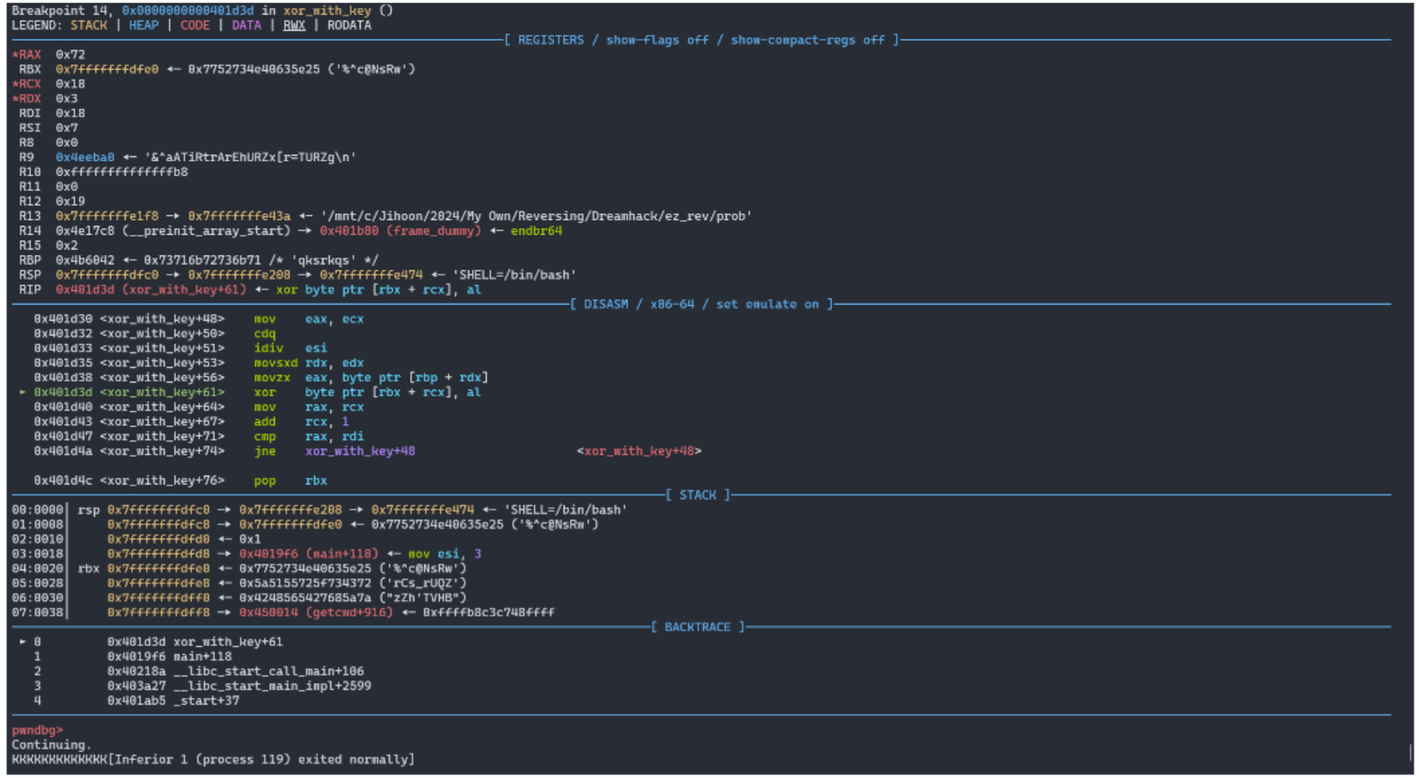
02 | PwnDbg : Why?



PE 파일 (.exe, .dll ...) 은 윈도우에서,
ELF 파일은 리눅스/유닉스에서!

- 윈도우 실행 파일인 PE 파일의 경우 ollyDbg, x64Dbg 에서 디버깅 가능!
- 그렇기에 당연히 리눅스/유닉스 실행파일인 ELF 파일의 경우
  리눅스/유닉스 환경에서 디버깅 하는 것!

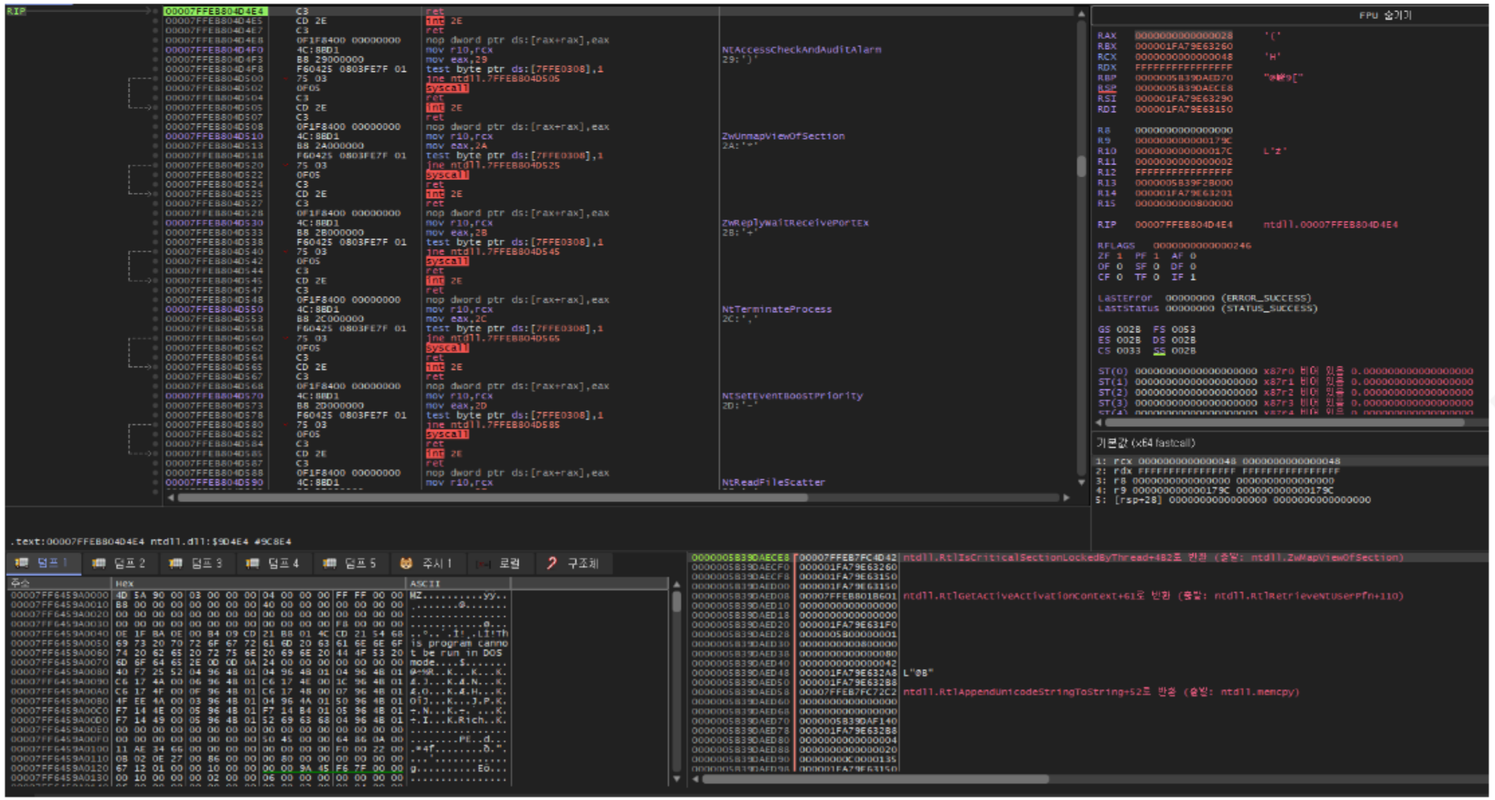
03 | PwnDbg : Contents



레지스터, 코드, 스택 등의 정보를 한눈에

- 레지스터 : CPU가 바로 사용할 수 있는 데이터
- Disassemble Code : 프로그램의 코드를 인간이 이해할 수 있는 언어로 역해석한 것
- 스택 : 함수의 호출과 관계되는 지역 변수와 매개변수가 저장되는 영역

04 | PwnDbg Vs : x64Dbg



GUI의 편의성을 이길 순 없지만..

- CLI 에서 실행되는 pwndbg
  → 많은 정보를 한번에 보여주기 어려움
- 가독성에서도 크게 떨어지지 않고 CLI 이기에 속도도 더 빠르다!
- 그리고.. 리눅스 실행 파일을 디버깅 하려면 어쩔 수 없다.

05

Training : Small Counter

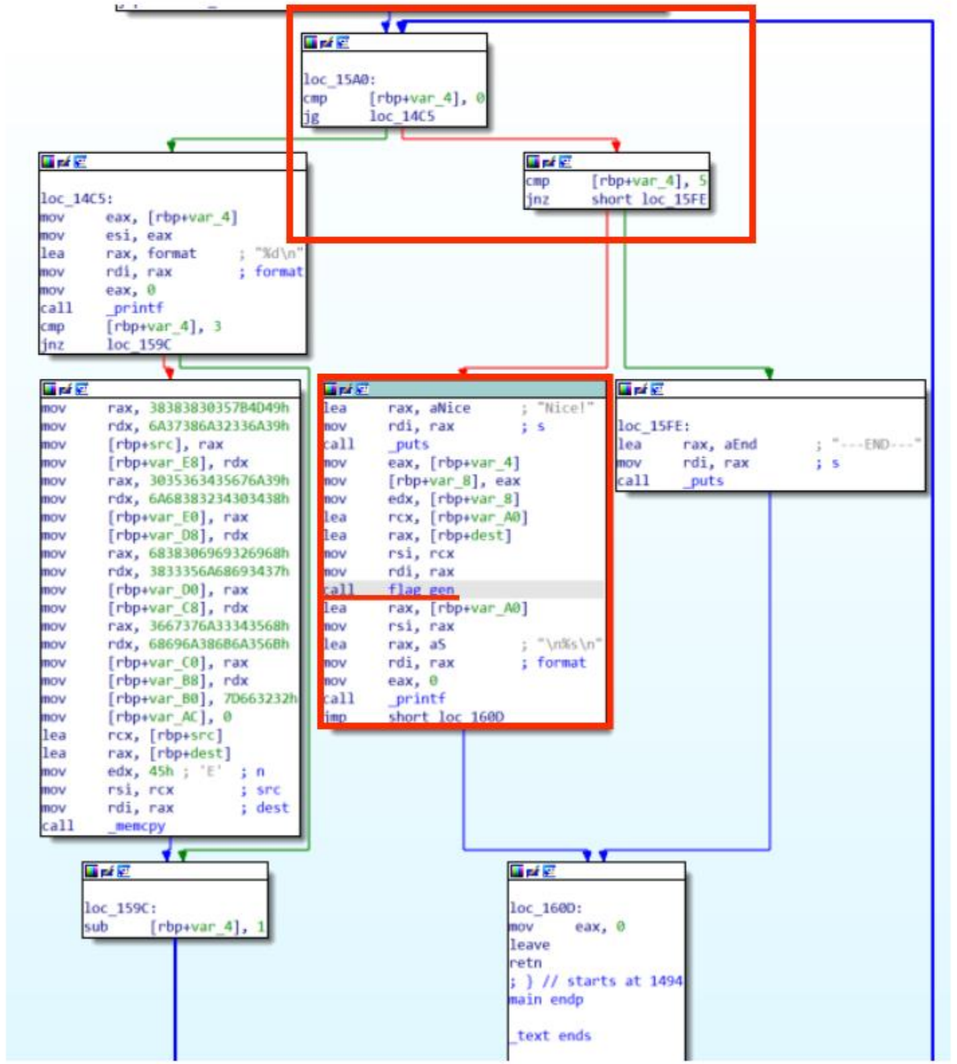
실행 : 10부터1까지 숫자를 세고 끝난다.

```
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/small counter$ ./chall
---Counter---
10
9
8
7
6
5
4
3
2
1
---END---
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/small counter$
```

05

Training : Small Counter

정적분석(IDA) : 실행에서 확인했던 값이 0과 같거나 작으면서, 5여야 flag_gen 함수에 도달?



05

Training : Small Counter

동적분석(PwnDbg) : 분기점에 BP(명령어 b)

```
   0x5555555555a0 <main+268>    cmp    dword ptr [rbp - 4], 0
 ► 0x5555555555a4 <main+272>    jg     main+49                    <main+49>
    ↓
   0x5555555554c5 <main+49>     mov    eax, dword ptr [rbp - 4]
   0x5555555554c8 <main+52>     mov    esi, eax
   0x5555555554ca <main+54>     lea    rax, [rip + 0xb41]
   0x5555555554d1 <main+61>     mov    rdi, rax
   0x5555555554d4 <main+64>     mov    eax, 0

00:0000| rsp 0x7fffffffdfe0 ◂— 0x0
... ↓        7 skipped

 ► 0   0x5555555555a4 main+272
   1   0x7ffff7db6d90 __libc_start_call_main+128
   2   0x7ffff7db6e40 __libc_start_main+128
   3   0x555555555105 _start+37

pwndbg> b
Breakpoint 3 at 0x5555555555a4
pwndbg>
```

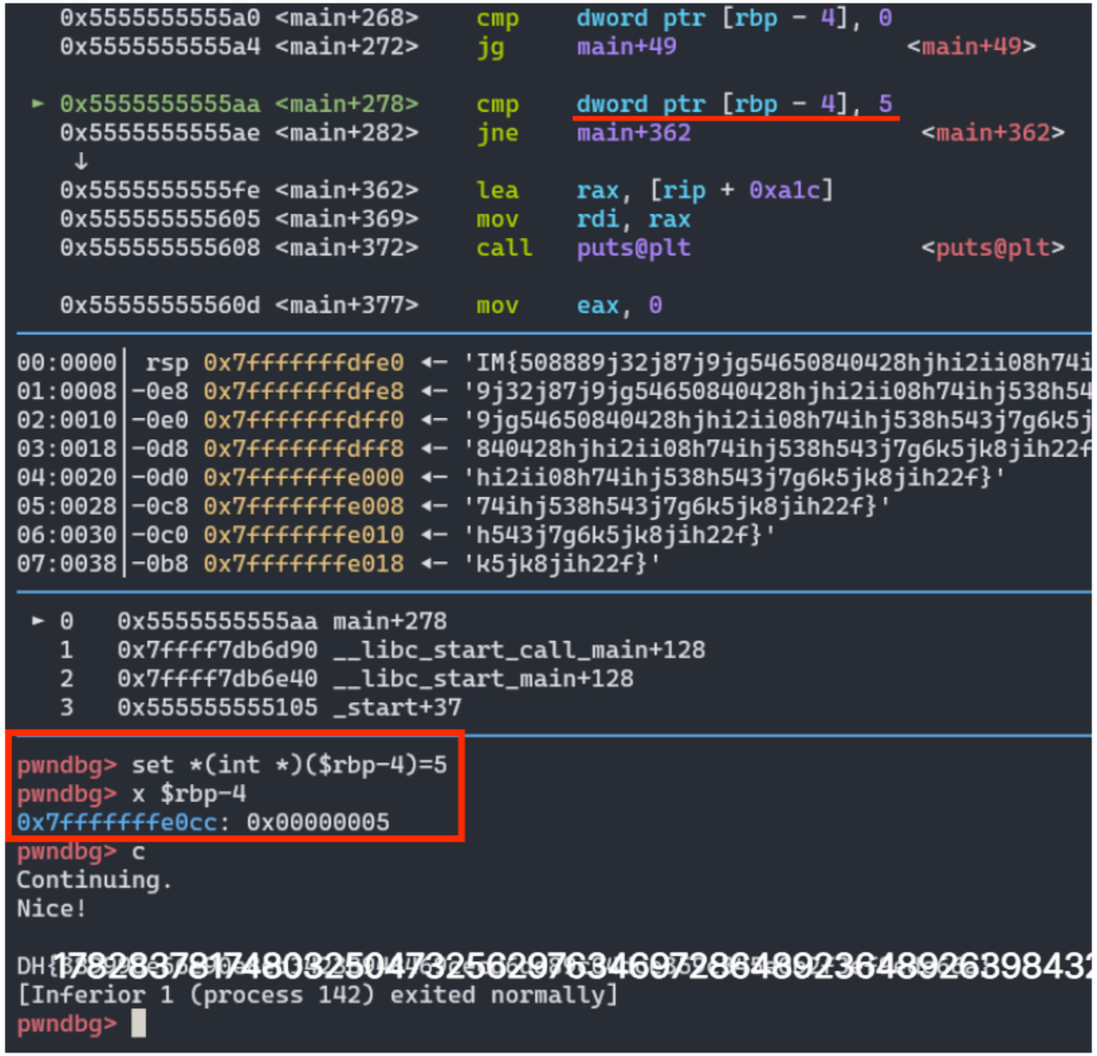
05

Training : Small Counter

첫 분기점을 지난 후 원하는 스택의 값을 변경 (명령어 set)

스택의 값이 바뀌었는지 확인 (명령어 x)

continue (명령어 c) : 분기점 통과, flag 획득.



06

Training : Check Function Argument

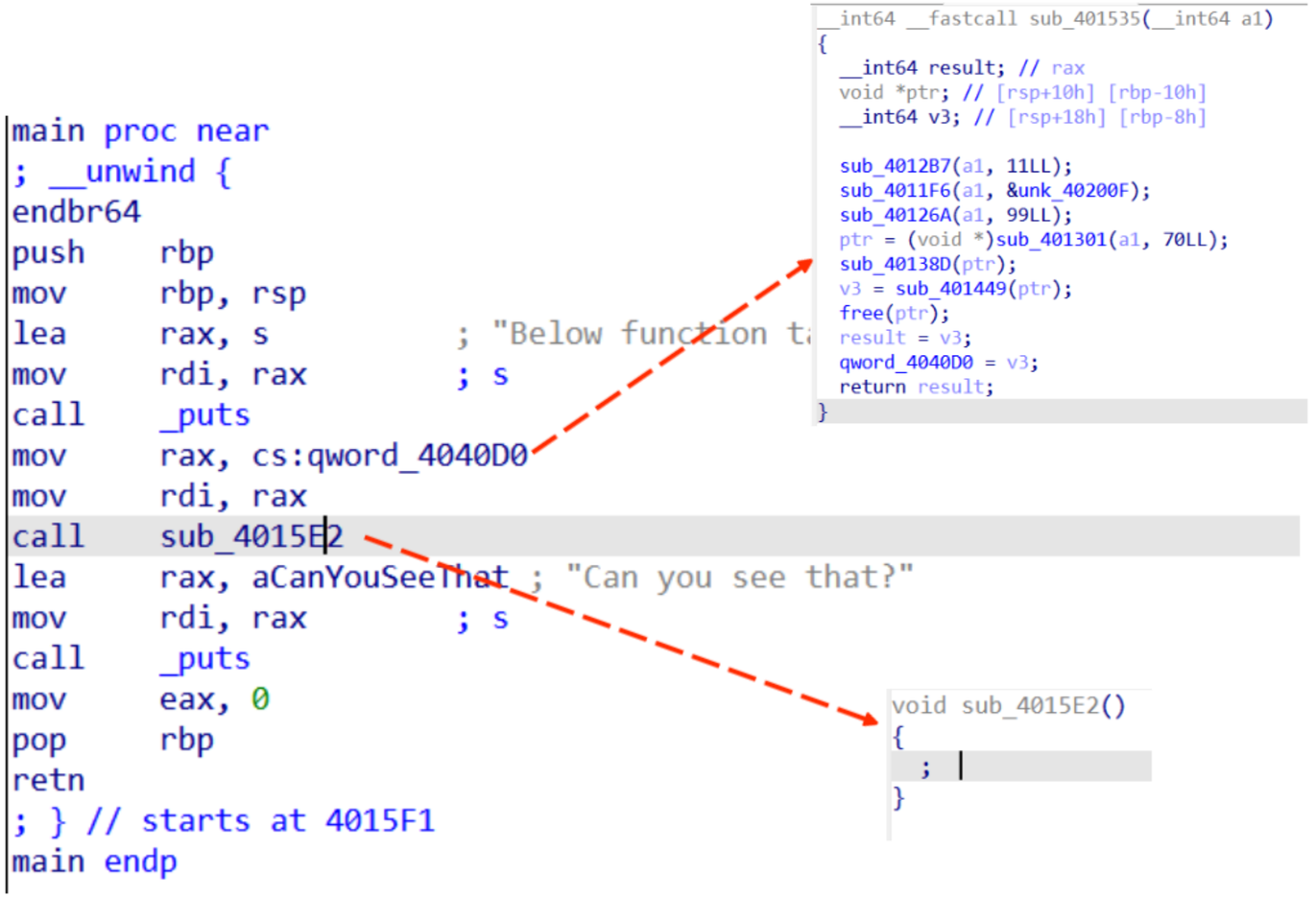
실행 : 아래의 함수가 인자값으로 flag를 갖고 있다고 한다. 출력은 해주지 않는다.

```
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/Check_Funtion_Argument$ ./check_function_argument
Below function takes the flag as an argument :)
Can you see that?
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/Check_Funtion_Argument$
```

06

Training : Check Function Argument

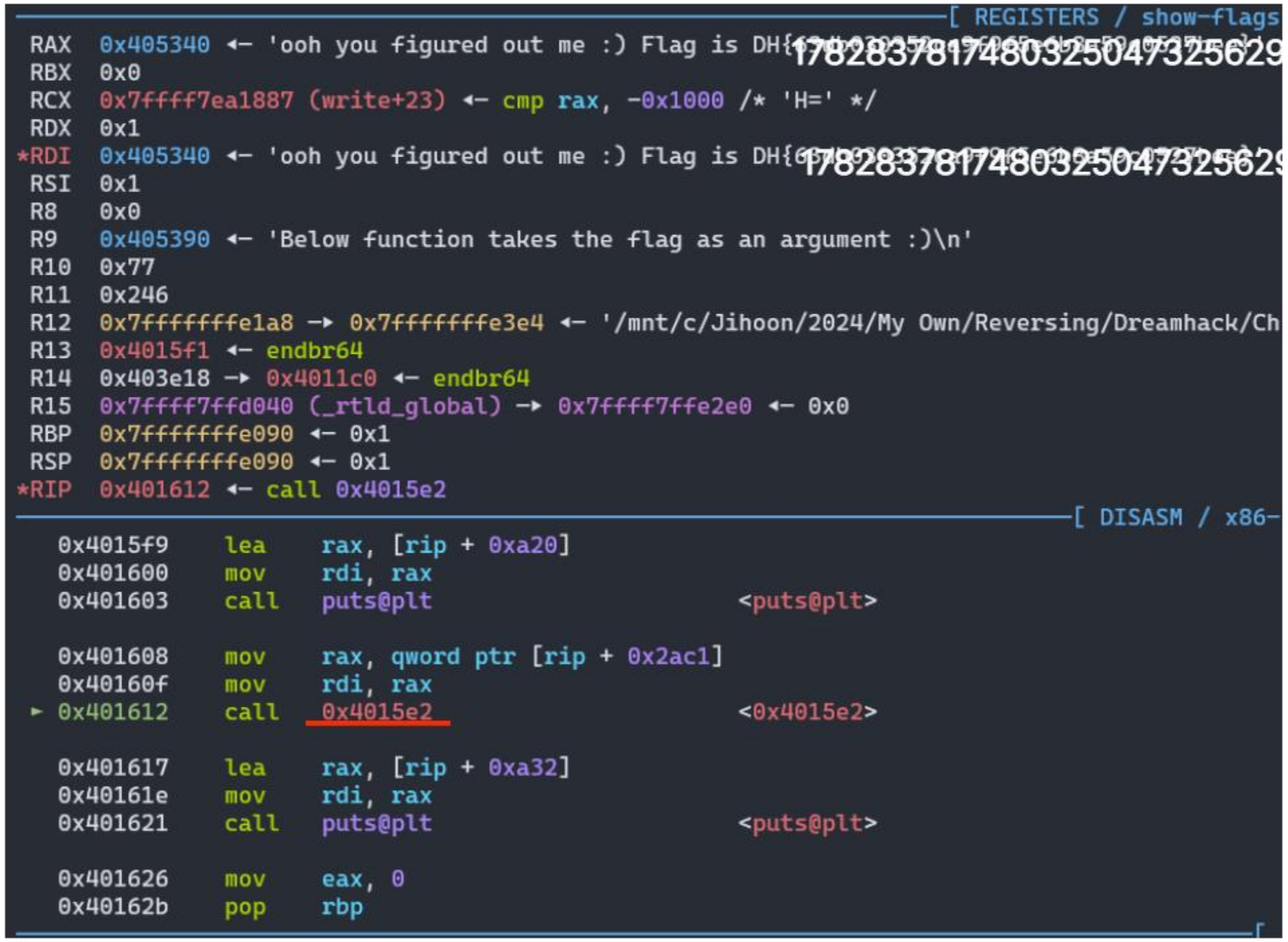
정적분석(IDA)



06

Training : Check Function Argument

동적분석(PwnDbg) : rdi = destination index



07

Training : Check Return Value

실행 : 이전 문제와 비슷하지만 인자값이 아닌 리턴값에 flag가 있는듯 하다.

```
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/Check_Return_Value$ ./check_return_value
OK. I will return flag.
I have returned the flag :)
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/Check_Return_Value$
```

07

Training : Check Return Value

정적분석(IDA)

```
main proc near
; __unwind {
endbr64
push    rbp
mov     rbp, rsp
lea     rax, s          ; "OK. I
mov     rdi, rax        ; s
call    _puts
lea     rax, unk_404080
mov     rdi, rax
call    sub_40152B
lea     rax, aIHaveReturnedT ; "I have returned the flag :)"
mov     rdi, rax        ; s
call    _puts
mov     eax, 0
pop     rbp
retn
; } // starts at 4015B6
main endp
```

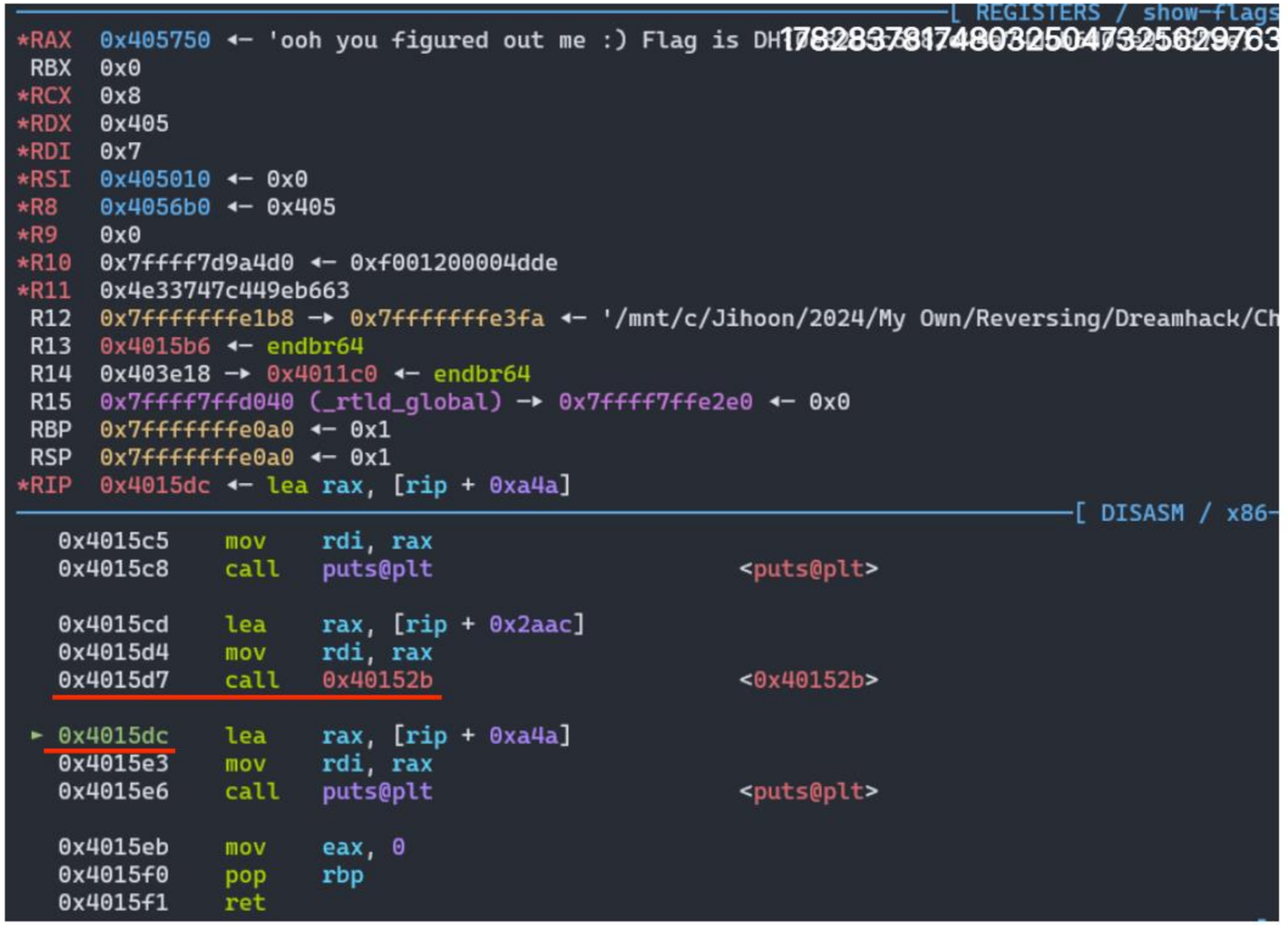
```
__int64 __fastcall sub_40152B(__int64 a1)
{
  void *ptr; // [rsp+10h] [rbp-10h]
  __int64 v3; // [rsp+18h] [rbp-8h]

  sub_4012B7(a1, 44LL);
  sub_4011F6(a1, &unk_40200B);
  sub_40126A(a1, 77LL);
  ptr = (void *)sub_401301(a1, 70LL);
  sub_401388(ptr);
  v3 = sub_401444(ptr);
  free(ptr);
  return v3;
}
```

07

Training : Check Return Value

동적분석(PwnDbg) : rax : 함수의 반환값을 저장



08

Training : ez_rev

실행 : 입력값을 넣으면.. 비웃음 당한다.

```
hoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/ez_rev$ ./prob
Input: 1234
KKKKKKKKKKKKKhoodscp@LAPTOP-0F5PGRC9:~/Dreamhack/ez_rev$
```

08

Training : ez_rev

정적분석(IDA) : 수많은 함수들 이후에 strcmp

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // edx
  int v4; // ecx
  int v5; // r8d
  int v6; // r9d
  __int64 v7; // rdx
  char _0[40]; // [rsp+0h] [rbp+0h] BYREF
  unsigned __int64 vars28; // [rsp+28h] [rbp+28h]

  vars28 = __readfsqword(0x28u);
  _printf_chk(1LL, "Input: ", envp);
  _isoc99_scanf((unsigned int)"%26s", (unsigned int)_0, v3, v4, v5, v6, _0[0]);
  shift_right(_0, 3LL);
  xor_with_key(_0, off_4E50F0);
  shift_left(_0, 3LL);
  xor_with_key(_0, off_4E50F0);
  shift_right(_0, 3LL);
  if ( !strcmp("|l|GHyRrsfwxmsIrietznhIhj", _0) )
    _printf_chk(1LL, "Correct!", v7);
  else
    _printf_chk(1LL, "KKKKKKKKKKKK", v7);
  return 0;
}
```

08

Training : ez_rev

정적분석(IDA) : 여러 함수들.. 일일이 분석할 수도 있겠지만?

```
__int64 __fastcall shift_right(_BYTE *a1, int a2)
{
  __int64 result; // rax
  int v3; // r12d
  __int64 v4; // r14
  char v5; // bl
  int v6; // [rsp+Ch] [rbp-4Ch]
  __int64 v7; // [rsp+10h] [rbp-48h]
  __int64 v8; // [rsp+18h] [rbp-40h]

  result = j_strlen_ifunc();
  v6 = result;
  if ( a2 > 0 )
  {
    result = (int)result;
    v3 = 0;
    v4 = (unsigned int)(result - 2);
    v7 = (int)result - 1LL;
    v8 = (int)result - 2LL - v4;
    do
    {
      while ( 1 )
      {
        v5 = a1[v7];
        if ( v6 <= 1 )
          break;
        ++v3;
        result = j_memmove(&a1[v7 - v4], &a1[v8], v4 + 1);
        *a1 = v5;
        if ( v3 == a2 )
          return result;
      }
      ++v3;
      *a1 = v5;
```

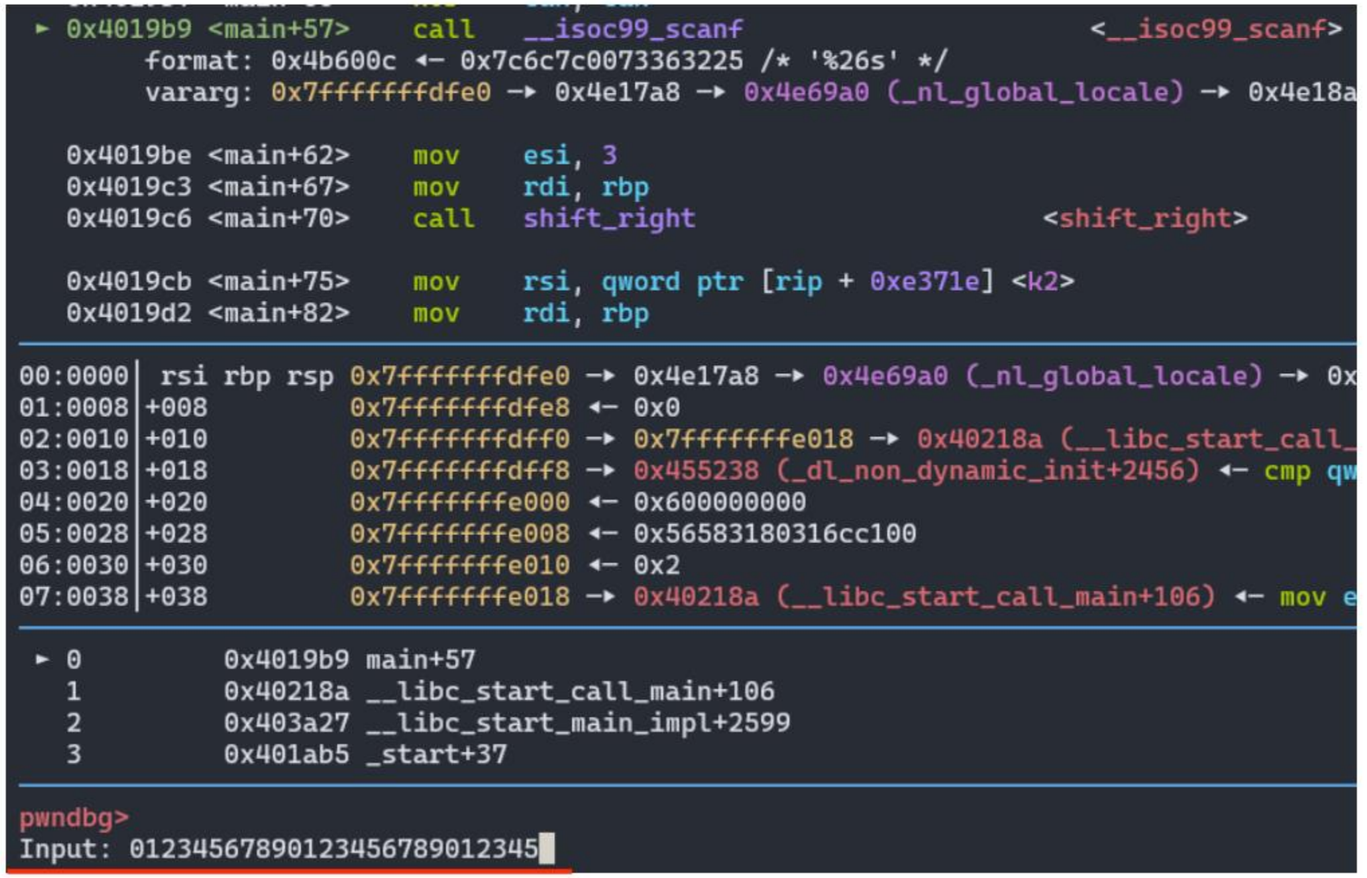
```
__int64 __fastcall xor_with_key(__int64 a1, __int64 a2)
{
  int v3; // r12d
  __int64 result; // rax
  int v5; // esi
  __int64 v6; // rcx

  v3 = ((__int64 (*)(void))j_strlen_ifunc)();
  result = j_strlen_ifunc(a2);
  if ( v3 > 0 )
  {
    v5 = result;
    v6 = 0LL;
    do
    {
      *(_BYTE *)(a1 + v6) ^= *(_BYTE *)(a2 + (int)v6 % v5);
      result = v6++;
    }
    while ( result != v3 - 1 );
  }
  return result;
}
```

08

Training : ez_rev

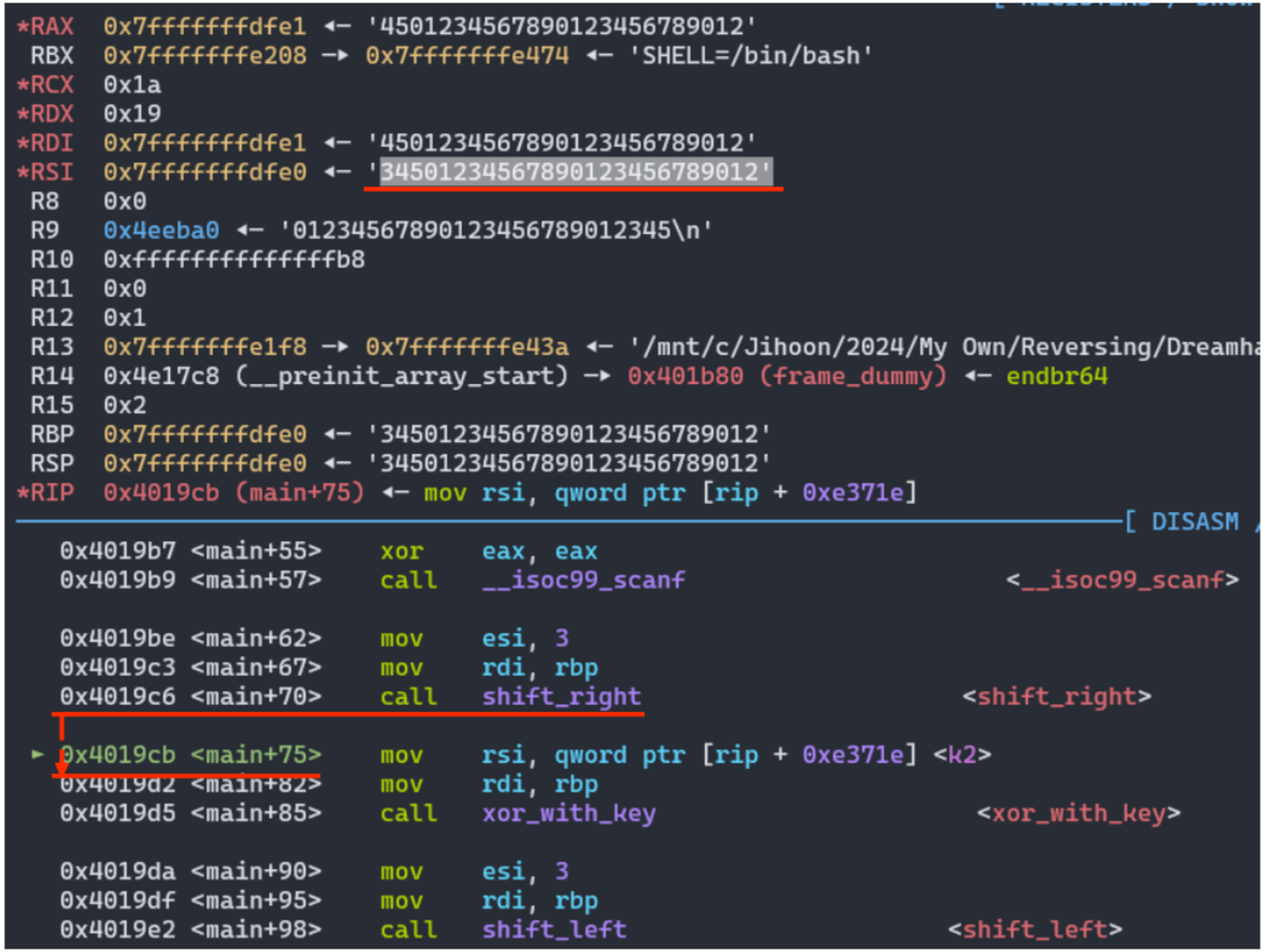
동적분석(PwnDbg) : 입력



08

Training : ez_rev

동적분석(PwnDbg) : shift_right 함수 기능



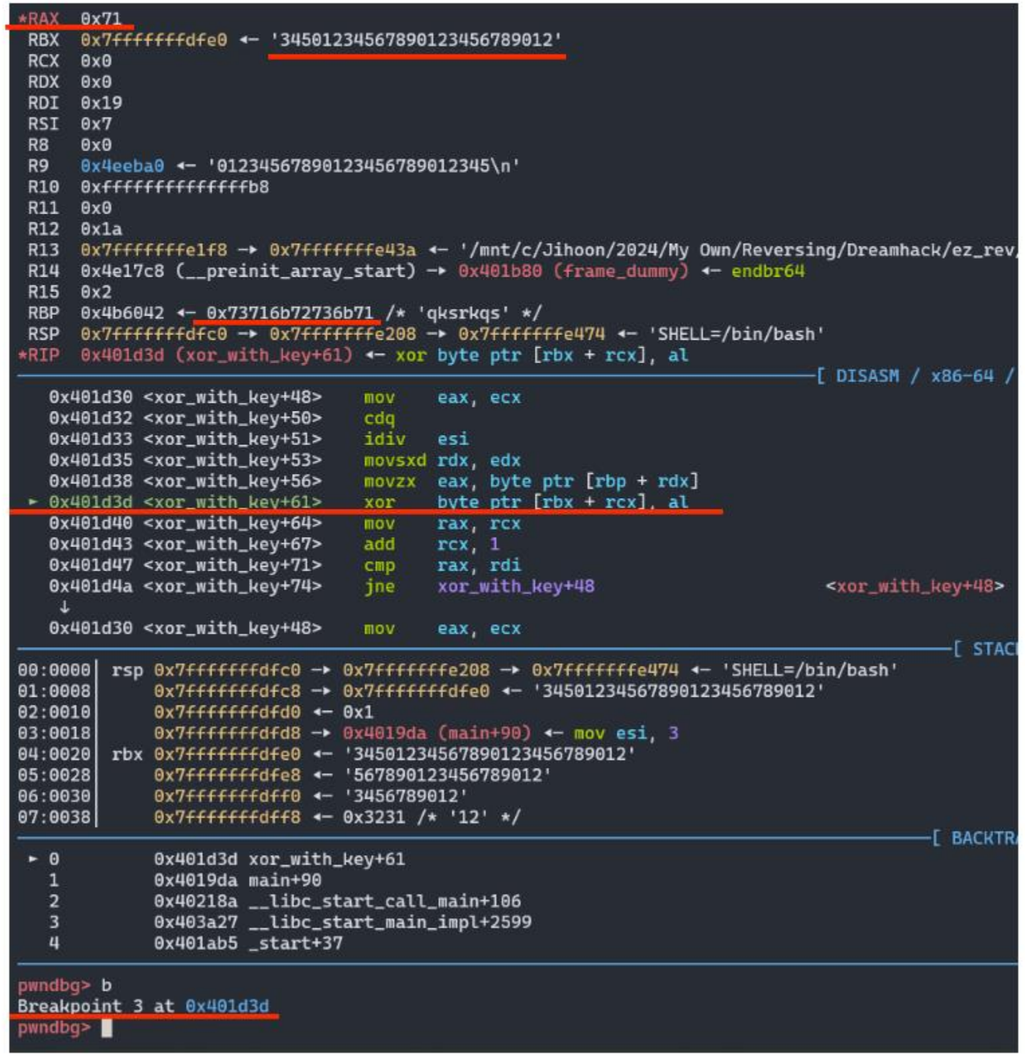
08

Training : ez_rev

동적분석(PwnDbg) : xor_key 함수 기능

xor 연산 부분에 BP 후 분석 :

71, 6b, 73, 72, 6b, 71, 73 순으로 xor 연산



08

Training : ez_rev

동적분석(PwnDbg) : shift_left 함수 기능

```
  0x401d4d <xor_with_key+77>    pop    rbp
  0x401d4e <xor_with_key+78>    pop    r12
  0x401d50 <xor_with_key+80>    ret
   ↓
  0x4019da <main+90>            mov    esi, 3
  0x4019df <main+95>            mov    rdi, rbp
► 0x4019e2 <main+98>            call   shift_left                    <shift_left>
      rdi: 0x7fffffffdfe0 ◂— 'B_FBZC@E^EESHC@Y@F^GDIRCCY'
      rsi: 0x3
      rdx: 0x4
      rcx: 0x1a

  0x4019e7 <main+103>           mov    rsi, qword ptr [rip + 0xe3702] <k2>
  0x4019ee <main+110>           mov    rdi, rbp
  0x4019f1 <main+113>           call   xor_with_key                    <xor_with_key>

  0x4019f6 <main+118>           mov    esi, 3
  0x4019fb <main+123>           mov    rdi, rbp
```

```
  0x4019da <main+90>      mov    esi, 3
  0x4019df <main+95>      mov    rdi, rbp
  0x4019e2 <main+98>      call   shift_left                    <shift_left>

  0x4019e7 <main+103>     mov    rsi, qword ptr [rip + 0xe3702] <k2>
  0x4019ee <main+110>     mov    rdi, rbp
► 0x4019f1 <main+113>     call   xor_with_key                    <xor_with_key>
      rdi: 0x7fffffffdfe0 ◂— 'BZC@E^EESHC@Y@F^GDIRCCYB_F'
      rsi: 0x4b6042 ◂— 0x73716b72736b71 /* 'qksrkqs' */
      rdx: 0x19
      rcx: 0x1a

  0x4019f6 <main+118>     mov    esi, 3
  0x4019fb <main+123>     mov    rdi, rbp
  0x4019fe <main+126>     call   shift_right                   <shift_right>

  0x401a03 <main+131>     mov    ecx, 0x1a
  0x401a08 <main+136>     lea    rsi, [rip + 0xb4602]
```

08

Training : ez_rev

역연산 코드 작성

```
shift_right(_0, 3LL);
xor_with_key(_0, off_4E50F0);
shift_left(_0, 3LL);
xor_with_key(_0, off_4E50F0);
shift_right(_0, 3LL);
if ( !strcmp("[l|GHyRrsfwxmsIrietznhIhj", _0) )
  _printf_chk(1LL, "Correct!", v7);
```

```
char arr[] = {"GHyRrsfwxmsIrietznhIhj|l|"}; // shift 후 첫번째 xor 을 위한 값

int xor_key[] = {0x71,0x6b,0x73,0x72,0x6b,0x71,0x73};
// xor 에 사용되는 키


for (int i=0; i < 25; i++)
{
  printf("0x%x, ", arr[i] ^ xor_key[i%7]);
}
```

```
0x36, 0x23, 0xa, 0x20, 0x19, 0x2, 0x15, 0x6, 0x13, 0x1e, 0x1, 0x22, 0x3, 0x1a, 0x14, 0x1f, 0x9, 0x1c, 0x3, 0x38, 0x
1b, 0x1b, 0x17, 0x1f, 0xe,
```

08

Training : ez_rev

역연산 코드 작성

```
shift_right(_0, 3LL);
xor_with_key(_0, off_4E50F0);
shift_left(_0, 3LL);
xor_with_key(_0, off_4E50F0);
shift_right(_0, 3LL);
if ( !strcmp("|l|GHyRrsfwxmsIrietznhIhj", _0) )
  _printf_chk(1LL, "Correct!", v7);
```

```
int arrr[] = { 0x17, 0x1f, 0xe, 0x36, 0x23, 0xa, 0x20, 0x19, 0x2, 0x15, 0x6, 0x13, 0x1e, 0x1,
0x22, 0x3, 0x1a, 0x14, 0x1f, 0x9, 0x1c, 0x3, 0x38, 0x1b, 0x1b};
// shift, xor, shift 후 xor 를 위한 값

for (int i=0; i < 25; i++)
{
  printf("%c", arrr[i] ^ xor_key[i%7]);
}
```

Thank you!

Q & A

PwnDbg 는 거들뿐.. IDA가 최고다..