COSE451 소프트웨어보안

Assignment0

Stage0_Debug_Me 분석

이정민/컴퓨터학과/2023320060/고려대학교

Stage0_Debug_Me 분석

주어진 과제는 Stage0_Debug_Me 파일을 gdb-peda를 사용하여 분석한 후 주어진 프로그램이 “CORRECT!”를 출력하도록 하는 문자열을 찾는 것이 목표이다. 이를 위해 우선 disas 명령어를 사용하여 main함수를 분석하였다.

```
gdb-peda$ disas main
Dump of assembler code for function main:
   0x08049228 <+0>:     push   ebp
   0x08049229 <+1>:     mov    ebp,esp
   0x0804922b <+3>:     sub    esp,0x34
   0x0804922e <+6>:     call   0x80491d4 <epilogue>
   0x08049233 <+11>:    lea    eax,[ebp-0x32]
   0x08049236 <+14>:    push   eax
   0x08049237 <+15>:    push   0x804a1dc
   0x0804923c <+20>:    call   0x8049060 <__isoc99_scanf@plt>
   0x08049241 <+25>:    add    esp,0x8
   0x08049244 <+28>:    lea    eax,[ebp-0x32]
   0x08049247 <+31>:    push   eax
   0x08049248 <+32>:    call   0x8049186 <check>
   0x0804924d <+37>:    add    esp,0x4
   0x08049250 <+40>:    test   eax,eax
   0x08049252 <+42>:    je     0x8049263 <main+59>
   0x08049254 <+44>:    push   0x804a1df
   0x08049259 <+49>:    call   0x8049050 <puts@plt>
   0x0804925e <+54>:    add    esp,0x4
   0x08049261 <+57>:    jmp    0x8049270 <main+72>
   0x08049263 <+59>:    push   0x804a1e8
   0x08049268 <+64>:    call   0x8049050 <puts@plt>
   0x0804926d <+69>:    add    esp,0x4
   0x08049270 <+72>:    mov    eax,0x0
   0x08049275 <+77>:    leave
   0x08049276 <+78>:    ret
End of assembler dump.
```

그 결과로 main 함수에서는 입력을 scanf 함수를 사용하여 사용자로부터 입력 받는다는 사실과 이를 check 함수로 넘긴다는 것을 알아냈다. 이를 통해 disas check 명령어를 이용하여 check 함수를 분석해야 한다는 결론을 낼 수 있었다.

```
gdb-peda$ disas check
Dump of assembler code for function check:
   0x08049186 <+0>:     push   ebp
   0x08049187 <+1>:     mov    ebp,esp
   0x08049189 <+3>:     sub    esp,0x4
   0x0804918c <+6>:     mov    DWORD PTR [ebp-0x4],0x0
   0x08049193 <+13>:    jmp    0x80491c7 <check+65>
   0x08049195 <+15>:    mov    edx,DWORD PTR [ebp-0x4]
   0x08049198 <+18>:    mov    eax,DWORD PTR [ebp+0x8]
   0x0804919b <+21>:    add    eax,edx
   0x0804919d <+23>:    movzx  eax,BYTE PTR [eax]
   0x080491a0 <+26>:    movsx  eax,al
   0x080491a3 <+29>:    mov    edx,DWORD PTR [ebp-0x4]
```

```
   0x080491a6 <+32>:    add    edx,0x804c040
   0x080491ac <+38>:    movzx  edx,BYTE PTR [edx]
   0x080491af <+41>:    xor    edx,0x11
   0x080491b2 <+44>:    movzx  edx,dl
   0x080491b5 <+47>:    sub    edx,0x3
   0x080491b8 <+50>:    cmp    eax,edx
   0x080491ba <+52>:    je     0x80491c3 <check+61>
   0x080491bc <+54>:    mov    eax,0x0
   0x080491c1 <+59>:    jmp    0x80491d2 <check+76>
   0x080491c3 <+61>:    add    DWORD PTR [ebp-0x4],0x1
   0x080491c7 <+65>:    cmp    DWORD PTR [ebp-0x4],0x1d
   0x080491cb <+69>:    jle    0x8049195 <check+15>
   0x080491cd <+71>:    mov    eax,0x1
   0x080491d2 <+76>:    leave
   0x080491d3 <+77>:    ret
End of assembler dump.
```

분석된 어셈블리어를 통해서 check 함수의 중요한 몇 가지 정보를 추론할 수 있었다.

```
0x0804918c <+6>:     mov    DWORD PTR [ebp-0x4],0x0
0x08049193 <+13>:    jmp    0x80491c7 <check+65>
...
0x080491c7 <+65>:    cmp    DWORD PTR [ebp-0x4],0x1d
0x080491cb <+69>:    jle    0x8049195 <check+15>
```

해당 부분을 통해서 check 함수는 총 30번의 반복과정을 통해서 입력값과 정답 문자열을 검사한다는 것을 알 수 있었다. 또한 마지막 0x080491cb <+69>를 통해서 반복문을 통해 이루어지는 과정은 0x08049195 <+15> 부터 0x080491c3 <+61> 임을 알 수 있었다. 그래서 다음 목표는 반복문 안에서 어떤 일이 이루어지는 지를 분석해서 'x/' 명령어를 사용해 정답 문자열을 추론하는 것으로 설정하였다.

```
0x08049195 <+15>:    mov    edx,DWORD PTR [ebp-0x4]
0x08049198 <+18>:    mov    eax,DWORD PTR [ebp+0x8]
0x0804919b <+21>:    add    eax,edx
0x0804919d <+23>:    movzx  eax,BYTE PTR [eax]
0x080491a0 <+26>:    movsx  eax,al
0x080491a3 <+29>:    mov    edx,DWORD PTR [ebp-0x4]
0x080491a6 <+32>:    add    edx,0x804c040
0x080491ac <+38>:    movzx  edx,BYTE PTR [edx]
0x080491af <+41>:    xor    edx,0x11
0x080491b2 <+44>:    movzx  edx,dl
0x080491b5 <+47>:    sub    edx,0x3
0x080491b8 <+50>:    cmp    eax,edx
0x080491ba <+52>:    je     0x80491c3 <check+61>
0x080491bc <+54>:    mov    eax,0x0
0x080491c1 <+59>:    jmp    0x80491d2 <check+76>
0x080491c3 <+61>:    add    DWORD PTR [ebp-0x4],0x1
```

반복문 안에서는 eax에 입력문자열을 한글자씩 가져온 후에 이를 비교하여 정답 문자열의 해당 인덱스의 값과 같다면 je 명령어를 통해서 반복문을 계속하고, 아니라면 0x080491bc <+54>: mov eax,0x0 으로 return 0을 하여 반복을 바로 종료한다. 이때 특이한 점은 해당 프로그램은 입력된 문자열과 정답 문자열이 담겨있을 것으로 추정되는 edx의 값에 0x11 xor 한 값을 비교한다는 것이다. 사용자가 입력한 문자열을 input으로 가정한다면 c언어를 역추론 해보았을 때 아래처럼 될 것이라고 생각하였다.

```
int check(char *input) {
    for (int i = 0; i <= 29; i++) {
        unsigned char from_data = *(0x804c040 + i);
        unsigned char target = (from_data ^ 0x11) - 3;

        if ((unsigned char)input[i] != target)
            return 0;
    }
    return 1;
}
```

결과적으로 check 함수는 30바이트 크기의 문자열을 인자로 받은 후 해당 문자열이 정답 문자열로 의심되는 data 배열로부터 특정한 조건인 (data[i] ^ 0x11) – 3과 같음을 검사함을 알 수 있었다. 이를 통해, 다음으로 해야할 것은 data문자열을 읽은 후 그로부터 input값을 추론하는 것임을 알 수 있었다.

‘x/30 0x804c040’를 통해서 edx에 저장되어 있는 값을 확인하였다. 그 결과 아래의 결과를 얻을 수 있었다.

```
gdb-peda$ x/30bx 0x804c040
0x804c040 <cmp>:    0x57    0x43    0x47    0x59    0x26    0x29    0x25    0x6f
0x804c048 <cmp+8>:  0x5a    0x75    0x62    0x62    0x6d    0x73    0x7a    0x26
0x804c050 <cmp+16>: 0x62    0x62    0x6d    0x73    0x47    0x47    0x42    0x73
0x804c058 <cmp+24>: 0x7e    0x7d    0x78    0x79    0x35    0x91
```

위의 결과를 앞서 특정한 조건인 (data[i] ^ 0x11) – 3을 고려하여 정답 문자열을 알아내고자 하였다. 이를 위해 간단한 python 코드를 사용하였고, 사용한 코드는 아래와 같다.

```
data = [
    0x57, 0x43, 0x47, 0x59, 0x26, 0x29, 0x25, 0x6f,
    0x5a, 0x75, 0x62, 0x62, 0x6d, 0x73, 0x7a, 0x26,
    0x62, 0x62, 0x6d, 0x73, 0x47, 0x47, 0x42, 0x73,
    0x7e, 0x7d, 0x78, 0x79, 0x35, 0x91
]

result = ''.join([chr((b ^ 0x11) - 3) for b in data])
print(result)
```

그 결과로 얻는 문자열은 다음과 같다.

```
PS C:\Users\a2349\OneDrive\바탕 화면\KoreaUniv\Course\25-1\소프트웨어보안\과제0> & C:/Users/a2349/AppDat
/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/a2349/OneDrive/바탕 화면/KoreaUniv/Course/25-1/소프
트웨어보안/과제0/reversing.py"
COSE451{Happy_h4ppy_SSP_life!}
```

해당 문자열을 r 명령어를 통해 프로그램 실행 후 입력하여 “CORRECT!”를 출력해냈다!

```
gdb-peda$ r
Starting program: /home/cose-451/Stage0_Debug_Me/Stage0_Debug_Me
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
--------------------------------------------------------------------------
Youngjae and Seogyeong are graduate students in Computer Science at Korea University.
They have been enjoying their happy... graduate life under the guidance of the King God
 Emperor Professor Seunghoon Woo.
Then one day, professor asked them a question.
What input should I put in this code to get 'CORRECT!'?
--------------------------------------------------------------------------

COSE451{Happy_h4ppy_SSP_life!}
CORRECT!
[Inferior 1 (process 87) exited normally]
```