

CSED415: Lab 01 Report

20230499/KimJaeHwan

1. Overview

과제 환경에 접속하면 target 실행파일과 target.c 소스코드가 있다. target.c가 포함되어 컴파일된 target 실행파일이 플래그를 출력하도록 취약점을 공격해야 한다.

2. Analyze target.c

소스 코드를 보거나 target을 실행시켜 본다면, print_flag() 함수를 실행시켜야 함을 알 수 있고, password 변수의 값과 SECRET의 값이 일치해야 함을 알 수 있다. SECRET은 146642라는 값으로 정의되어 있으므로, password의 값을 맞게 변경해야 한다.

이 프로그램에서는 크기가 8인 테이블 배열 변수가 있으며, 입력을 두 번 받아서 첫 번째 입력의 위치에 있는 배열의 원소를 두 번째 값으로 수정한다.

- get_user_input() 함수를 통해 첫 번째 입력을 받아,
- get_elem_ptr(index) 함수를 통해 첫 번째 입력의 위치의 주소를 가져온다.
- get_user_input() 함수를 통해 다시 두 번째 입력을 받아,
- *ptr = value; 구문을 통해 값을 넣는다.

따라서 get_elem_ptr(index) 함수를 통해 password 변수의 주소에 접근할 수 있다면, 원하는 값으로 password 값을 변경할 수 있다.

3. Process memory map

password 변수와 table 배열 변수는 static으로 선언되었으므로, .data 부분에 있다. 코드에서 친절하게 각 변수의 메모리 주소를 출력해 주므로, 한 번 실행시킨 후 나오는 출력의 결과물을 통해 얻은 각 변수의 주소를 이용해 메모리 맵을 그리면 다음과 같다.

.data Section	
0x4040C0	password (int) : 2025
0x4040C4	(Padding)
0x4040E0	table[0] (int) : 29
0x4040E4	table[1] (int) : 71
0x4040E8	table[2] (int) : 95
0x4040EC	table[3] (int) : 90
0x4040F0	table[4] (int) : 2
0x4040F4	table[5] (int) : 75
0x4040F8	table[6] (int) : 62
0x4040FC	table[7] (int) : 3
0x404100	

4. Exploit the vulnerability

앞서 서술한 바와 같이, `get_elem_ptr(index)` 함수에 치명적인 취약점이 있다. 주어진 코드는 다음과 같다.

```
int *get_elem_ptr(int index) {
    if (index < SIZE) {
        return table + index;
    }
    return NULL;
}
```

위 함수에서 반환하는 포인터의 주소는 `table + 4 * input`인데, 테이블의 크기가 8이므로 해당 크기보다 큰 입력에서는 NULL을 반환하며, 함수 바깥에서 NULL인 경우를 적절히 처리하도록 구현되어 있다. 그러나 `index`의 하한은 정해져 있지 않으므로, 음수를 입력으로 넣을 경우 `table`의 주소보다 낮은 주소의 위치를 반환하도록 만들 수 있다. 앞서 실행시켰을 때 `password`의 위치가 `table[0]`의 위치로부터 `-0x20 = 32`만큼 떨어져 있었으므로, `input`에 `-8`을 넣으면 `get_elem_ptr()` 함수가 `password`의 메모리 주소를 반환하도록 만들 수 있다.

5. Security patch

이 함수를 다음과 같이 수정하면 된다. 인덱스의 범위를 알맞게 제한하거나, 인덱스이므로 사용자의 입력으로 `unsigned_int`를 사용하는 것도 방법이다. 다음과 같은 방법들을 통해 수정하면 `get_elem_ptr()` 함수가 배열 외부의 주소를 반환하는 취약점을 막을 수 있다.

```
int *get_elem_ptr(int index) {
    if (0 <= index && index < SIZE) {
        return table + index;
    }
    return NULL;
}
```

```
int *get_elem_ptr(int index) {
    if (index < 0 || index >= SIZE) {
        fprintf(stderr, "Error: Invalid index.\n");
        return NULL;
    }
    return table + index;
}
```

알맞은 위치가 아닌 경우 NULL포인터를 반환한다.

```
int *get_elem_ptr(unsigned int index) {
    if (index < SIZE) {
        return table + index;
    }
    return NULL;
}
```

음수의 입력인 경우 매우 큰 양수로 취급되어 NULL 포인터를 반환할 것이다.

`assert.h` 헤더에서 제공하는 `assert()` 함수를 이용해 예외 처리를 하는 것도 좋은 방법이다.

6. Result

date 명령과 함께 타겟을 실행하여 얻은 플래그를 첨부하였다. 플래그가 시간에 따라 변화하는 것으로 생각되어, 현재 시각과 함께 출력하였다. 플래그는 2025년 2월 28일 오후 6시 00분 30초에 마지막으로 얻었다.

```
csed415-lab01@csed415:~$ ./target <<< "-8 146642" && date
Let's get warmed up! Invoke print_flag() to capture your flag.

----- Current table entries -----
(생략)
-----

[+] Enter the index to modify: [+] Enter a new value for table[-8]:
----- Current table entries -----
(생략)
-----

----- password address and value-----
Addr: 0x4040c0 -> Value: 146642
-----

Great job! :)
This is your flag:

944583A6CFFB89C892AEABE82B57E278CFA4A2197AD63A9A11218192ECA1C51A
E0F66CCE7011C4EED08566997A3F058268F2C355C84C3A7DE08E8DC8C947172B
A0894A94169E7E2A27E44DA6D8BA3A5C8280EDA00E8A51DE63FD0A461A2F0CAC
BFD92AF440AB02D23DDC71C4860050CB041ABC226D3F0936A992F1CEE8A07884
184FE1923CB744CE2D2EE87EEEF1A7EA420013E2C407D0C124C5DEF397D8A898
75089AE84E5BA88FDECC77850A59E8A9D17A27E3188497AF08BA711780ECA411
968ADE98FFE742B8A5751BACEFCF5043881E49D2501579EACEB4F123429F53EF
0D0FA84D94C00699E391966E5F4C7DBEE1D62BB46108FD0CBFD2C73AFF34A6D4
2834F83113B9AC022B986D8987874CA290C56DFCCF468B2BAE433EB0BAC69A99
FBF1B236972B75A5F37CA0C12EB86E6A897C72DCE05BE1BD116F13F7B3EC8615
647FBD873677DFB2E38264C20DA4E25A76FC94B4CA806C601034BC38AC8BC8E4
FED9AC9825907407B668D6E5A02D0A38CD9A76680494C4F547725C22DCE50341
00CA32A56E5E79CA70B21A37B7DFF5CB077FE43AD4B78822E689AC8CCDC1B968
3B851016A5DC57BFD9B3E1C8B5E3649E73EB6014E2F01035583DF79D6C90AC0
18AF0F8B6AE7B69B65DD1F79CAB1CBF4DB1EECB8A721ECEFA79ED946BD2B9A68
7290A4316B7ADD75E4F030CC778CBD6512C8E3087B869FE1CFFCA41BB74D3F3A

Fri Feb 28 06:00:30 PM KST 2025
```