Software Vulnerabilities: Exploitation and Mitigation

—

Lab 6

Prof. Jacques Klein & Pedro Jesús Ruiz Jiménez
(inspired from Prof. Alexandre Bartel's course)

6 Heap Overflow (35 P.)

In this lab you will exploit a heap buffer overflow to execute arbitrary code. In this lab, the target architecture is x86 in 32-bit mode and the way parameters are passed to a function, the registers available, etc. differs slightly from x86_64.

6.1 Setup Labs Environment

6.1.1 Install tools

Install *QEMU* by following the instructions here.

6.1.2 Download resources

Download the Debian 2.2 virtual image here (user:root, password:svem).

6.2 Launch Emulated Environment

On your host machine, go to the directory where `Debian2.qcow2` is located. Then, create the virtual machine by using the following command:

```
$ qemu-system-i386 -hda Debian2.qcow2 -m 1024
```

Note that in this version of debian, the network does not work with qemu, so you will not be able to share a directory between the host and the guest. However, do not worry since:

1. The C program is already available
2. The only file you will have to create in the for this lab is the input file containing 24 bytes.

6.3 Vulnerable Code

The goal of this lab, is to execute the `executeme` function by exploiting a heap overflow vulnerability.

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

void executeme() {
    int a = 2;
    if (a > 42) {
        printf("Not reachable\n");
    }
    printf("Congrats, you have reached the end of this lab!\n");
    exit(-1);
}

int main(int argn, char** argv) {
    void* m1;
    void* m2;

    m1 = malloc(10);
    m2 = malloc(10);

    printf("m1: 0x%x\n", m1);
    printf("m1: 0x%x\n", m2);

    strcpy(m1, argv[1]);

    free(m1);
    free(m2);

    return 0;
}
```

Compile this program with the following command:

```
$ gcc -g -N heap.c
```

Question 6.1 Describe options g and N. 2 P.

Question 6.2 Where is the heap overflow vulnerability in the code? Explain. 2 P.

Question 6.3 When will the control flow of the program be redirected? Explain. 2 P.

For the following questions, you might want to look at the /proc/PID/map file to see where the heap and the stack are. In this old version of Debian, the stack and the heap will not be indicated when you look at the file. However, you known that the heap is after the text segment (code) and the stack starts at a high address...

Question 6.4 At what addresses are located m1 and m2? At what addresses are located heap chunks for m1 and m2? Explain the values in the"size" and "prev_size" fields after the first free function has finished and with the input "AAAAAAAAAA". 4 P.

For the next question, you can use the following steps to put a breakpoint on the free function and then print the stack when free is called (input contains the bytes you give to argv[1]):

```
(gdb) bp chunk_free
(gdb) print free
(gdb) bp free
(gdb) run `cat input`
(gdb) x/40gx $esp
```

Question 6.5 At what address on the stack is/are located the return address/es of the "free()" function (this return address is the address of the instruction following the first "call free" instruction in main)? 2 P.

Question 6.6 At what address is located the `executeme` function? 2 P.

For the following question, you need to construct your input from hexadecimal values. You can do it by using the following command:

```
$ printf "\x41\x41\x41\x41" > input
```

Question 6.7 What input do you give to the main function so that the control flow is redirected to the executeme function? Draw a heap representation to explain how you manipulate heap data to achieve the redirection. Describe your input. (Hint 1: the total length of the input should be 6x4 = 24 bytes, no more, no less and must follow the structure we have seen in the lecture) (Hint 2: big endian or little endian?) 10 P.

Question 6.8 The executeme function has a weird condition which is never executed. Why is this useful to the attacker? Explain. Could this heap overflow correctly execute any function? Is the heap marked as executable in this version of Debian from the 2000's? 6 P.

6.4 Defense

Question 6.9 Describe how you would prevent heap buffer overflow exploitations. 5 P.

Note on plagiarism

Plagiarism is the misrepresentation of the work of another as your own. It is a serious infraction. Instances of plagiarism or any other cheating will at the very least result in failure of this course. To avoid plagiarism, always cite the source from which you obtained the text.