Software Vulnerabilities: Exploitation and Mitigation – Lab 7

Alexandre Bartel

The report for the lab should consist of a single pdf file. Please use the following filename:

`lab7_FIRSTNAME_LASTNAME.pdf`

Send the report to alexandre.bartel@uni.lu with the following subject:

`MICS2019SVEM Lab7 FIRSTNAME_LASTNAME`

The deadline is the 21st of April 2019 at 23:59.

1 Lab7 (30 P.)

In this lab you will exploit a heap buffer overflow to execute arbitrary code. Use the Debian 2.2 image available here (user:user, root:root). The default layout is qwerty. You can change to the azerty layout once logged-in by typing "loadkeys fr" (save it in .bashrc to avoid typing it at every login). Note that in this lab, the target architecture is x86 in 32-bit mode. The way parameters are passed to a function, the kinds of registers available, etc. differs slightly from x86_64. Note that in qemu, if you want to open a second terminal, first type (ctrl-alt-2) to enter qemu's console. Then, type (sendkey ctrl-alt-f2). Then, type (ctrl-alt-1) to go back to the new terminal!
Note that you can also use the "screen" program to have multiple screens. Type CTRL-A and then C to create a new screen. Type CTRL-A and then N to switch from one screen to the other.
The useful programs available to you in this lab are: screen, gdb, gcc and hexedit. Unfortunately, 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 in /home/user/; (2) the only file you will have to create in the VM for this lab is the input file "input" containing 24 bytes.

1.1 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 1.1 Describe options *g* and *N*. 2 P.

Question 1.2 Where is the heap overflow vulnerability in the code? Explain. 2 P.

Question 1.3 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 1.4 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 1.5 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 1.6 At what address is located the `executeme` function? 2 P.

For the following question, you can use the `hexedit` tool to edit a file. To save press F2. To truncate at the current position press ESC-T. To quit press CTRL-C.

Question 1.7 What input to 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 achive 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? That is the question) 10 P.

Question 1.8 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.

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 properly cite your sources.