CSE4501 – Vulnerability Research: Lab 5

Eric Pereira

Septemer 24th, 2019

Contents

This lab involves utilizing an ASLR defeat to redirect execution to code of our choosing. Lab is worth 30 pts

to build the lab executable, simply execute:

```
$  make
```

This will produce an executable named `aslr_lab`.

Part 1 - Demonstrate EIP control 10pts

The basic vulnerability we are exploiting is an out of bounds array access. In `main()` an array of struct record objects are allocated on the stack. A pointer to this array is passed to several functions that prompt the user to enter a record to access. For example, in `save_record()` we see.

```
printf("Which record do you want to save to?\n");
uint32_t recordnum = read_u32();
struct record *r = &records[recordnum];
```

We then store user provided input to a potentially invalid array member. In `main()`, note that a function pointer is created on the stack, which points to `lose()`. Your goal for this part is to overwrite the function pointer with some arbitrary value, such as `0x41414141` and cause the program to execute it.
Solution:

Part 2 - Exploit Without ASLR 10pts

Once you ensure we have EIP control, the next step involves executing the `win()` function. We will first do this with ASLR disabled. First make sure that ASLR is disabled on your machine. To check, run the following command in the aslr_lab directory

```
$  ./set_aslr
ASLR value is currently:
0
```

If this does not show 0, run

```
$  sudo ./set_aslr 0
```

This should disable ASLR.

Now, determine the load address of the `win()` (via gdb, IDA Pro, etc.) function and exploit the out of bounds access vulnerability to execute it.

Solution:

Part 3 - Exploit with ASLR 10pts

Now that you know you can execute `win()`, turn on ASLR

```
$  sudo ./set_aslr 2
```

This should enable ASLR.

Try exploiting the vulnerability with the base address of `win()` that you discovered in Part 2. This will most likely cause a segmentation fault. This is due to the address of `win()` being randomized by ASLR. Your goal for this lab is to execute `win()` with ASLR on. To do this, you will have to utilize the out of bounds access vulnerability to leak information that will allow you to determine where `win()` is located, then use the out of bounds access to exploit the vulnerability to gain code execution.

Going Further

As an exercise for the reader, and your own edification, see if you can leverage the vulnerability to gain complete arbitrary code execution, either via shellcode or ROP. Is it possible? We don't know, that's your job! But if you succeed, you get 1337 hax0r points and the satisfaction of a job well done.