CSE 5474 — Software Security Spring 2022

BabyFMT

Due April 2nd, 2022

1 Overview

In this CTF assignment, you are given 11 challenges with varied difficulties (from `babyfmt_level1` to `babyfmt_level11`). `babyfmt_level11` is bonus challenge. These challenges require you to either overwrite the return address of the vulnerable function to hijack its control flow, or directly change some function address (e.g., canary checking) by using string format vulnerability. In this challenge, you need to use stack overflow, or string format vulnerability, or both. ASLR is enabled in all challenges. (The required information is provided in a few challenges to make exploitation easier).

To exploit format string vulnerability, we recommend you to use pwntool. Pwntool provides fmtstr and fmtstr_payload functions. You can refer its manual at `https://docs.pwntools.com/en/stable/fmtstr.html`.

- `babyfmt_level1-2` (5 pts). They are simple challenges on string format vulnerabilities. You can overwrite the function address by exploiting %n in format string specifier.
- `babyfmt_level3` (7 pts). In this challenge, it takes two inputs. You can exploit the format string vulnerability in the second input. The address of the buffer is provided. You can inject shellcode to the buffer and overwrite some address (you need to figure it out) to the buffer address that contains your injected shellcode.
- `babyfmt_level4` (7 pts). Similar to baby_level3, it takes two inputs, but the the canary is enabled. You need to bypass canary.
- `babyfmt_level5` (7 pts). In this challenge, it takes one input. It should execute the `win` function, which uses a variable to control how to read `flag`. You need to change the value of the variable properly.
- `babyfmt_level6-7` (7 pts). In these challenges, stack overflow is required to get the flag. To get the flag, you need to execute `win` function. However, position independent executable (PIE) is enabled. Hence, the function address will be changed every time you run it. The program gives you a hint how to solve it.
- `babyfmt_level8` (9 pts). The baby_level8 requires you to use ROP. The `win` function is still there, but it takes two parameters, and then you need to pass parameter properly with ROP gadgets.
- `babyfmt_level9` (9 pts). In this challenge, it executes `strcpy` to copy an input. There is no any information about the address of the buffer. You need to figure out how to execute the buffer. Hint: Check the arguments of the strcpy function.

- `babyfmt_level10` (11 pts). Level 10 is similar to the level 10 of the previous CTF. However, it only provides address of different function. You need to calculate the addresses of system and setuid functions to get the flag.
- `babyfmt_level11` (bonus 13 pts). This is the most difficult challenge on in this CTF. There is no hint on baby_level11 challenge since it is bonus challenge.

2 Format string vulnerability

The format string vulnerability exists when the below pattern occurs. The basic idea of the format string vulnerability is using %n.

```
read(0, buf);
printf(buf);
```

When the below printf executes, the local variable `s` will change to 13. Because the number of "The value of " consists of 13 characters.

```
int s;
printf("The value of %n", &s);
```

Without a local variable, it will store the value to where %p is indicating. For example, when the `buf` is "cc%59$p", assume that the result of the printf is "cc0x1000". The value 0x1000 comes from the stack so you can read the value of the stack by increasing the number. When the %p of the buf replaces with %n and the printf executes, the value of 0x1000 will change to 2.

You can use the pwntool library function named `FmtStr`.

```
def exec_fmt(payload):
    r = process("program name")
    e = r.elf
    payload = payload.replace(b"END", b"HEYHEY")
    r.clean()
    r.sendline(payload)
    r.shutdown(direction="send")
    data = r.recvuntil(b"HEYHEY", timeout=0.2)
    data = data.replace(b"HEYHEY", b"END")
    return data

autofmt = FmtStr(exec_fmt)
print(autofmt.offset)
print(autofmt.padlen)
```

The above code snippet is the tutorial how to use the `FmtStr`. After this command, you can use the `fmtstr_payload` function to craft payload.

3 Deliverables

The flags you captured, and please submit them in `https://cse5474.osuseclab.com`, and the writeup describing how you solve these challenges. **Please also attach your code in your write up, or package it separately**.