Previous Lab Review

Race condition competition question.

Answer:

1.2 -> A -> 1.1/2.1 -> B

Task 4: sticky bit protection

Return to libc

Kailiang

Task 1 : exploit vulnerability

Understand how $esp, $ebp move in function frame

push $ebp

mov $esp, $ebp

…

leave

ret

Task 1 : : exploit vulnerability

Draw the function frame in your report to show how return-to-libc attack happen.

Where to put?

1. system()
2. exit()
3. /bin/sh

Task 1 : exploit vulnerability

Debug program

1. Get the return address
2. What is the address of system() and exit()
3. DO NOT debug set-root-uid program

Task 1 : exploit vulnerability

Environment Variable

1. system() call need argument
2. argument reference store in function frame
3. Export MYSHELL=/bin/sh

Task 1 : exploit vulnerability

Get Environment Variable Address

```
void main ()
{

        char *shell = getenv("MYSHELL");

        if(shell)
                printf("%x\n", (unsigned int)shell);

}
```

Task 1 : exploit vulnerability

Filename length will affect the address of environment variable.

1. In your report, you need to explain to me how does filename length affect environment variable address. Show me your **debug result**.

Task 1 : exploit vulnerability

Generate badfile

```
/* You need to decide the addresses and
   the values for X, Y, Z. The order of the following
   three statements does not imply the order of X, Y, Z.
   Actually, we intentionally scrambled the order. */
*(long *) &buf[X] = some address ;   //  "/bin/sh"
*(long *) &buf[Y] = some address ;   //  system()
*(long *) &buf[Z] = some address ;   //  exit()
```

Task 1 : exploit vulnerability

Launch Attack

```
$ gcc -o exploit exploit.c
$./exploit          // create the badfile
$./retlib           // launch the attack by running the vulnerable program
# <---- You've got a root shell!
```

Task 1 : exploit vulnerability

In your report…

1. Write down how you get system(), exit(),environment variable address
2. Answer the question in lab description

Task 2 : Address Randomization

```
$ su root
  Password: (enter root password)
# /sbin/sysctl -w kernel.randomize_va_space=2
```

1. Repeat task 1
2. Show me your observation and explain it

Task 3 : Stack Guard

```
$ su root
  Password (enter root password)
# gcc -z noexecstack  -o retlib retlib.c
# chmod 4755 retlib
# exit
```

1. Repeat task 1
2. Show me your observation and explain it

Grade criteria

- Task1 : 60%
- Task2 : 20%
- Task3 : 20%