Report for Computer Security Quiz 1

20161105 박수근

1. objdump

> What is the address (*without leading zeroes*) of the 'rodate' in the dynamic section in the provided file?"

Command

```
$ objdump -x 1 | grep rodata
```

`rodata` is the name of section, so I thought it is located on symbol table with its address. The `-x` option of `objdump` let me know the contents of all headers and the symbol table is contained in those datas. So I used objdump with `-x` option, and use `grep` command to find the line that contain `rodata`.

2. strings

> Find the flag in the given executable with the strings tool!

Command

```
$ strings 2 | grep flag
```

I used `strings` tool to check ASCII character data in the executable. And I expected that the answer is located with the word 'flag', so I used `grep` command to find the line with the word 'flag'.

3. GDB, change variable

> Inspect this binary with GDB. More specifically, change the values (a, b) in the main function to print out the flag!

Command

```
$ chmod 777 ./3    # give permission to execute
$ gdb ./3
```

```
(gdb) b main
(gdb) run
(gdb) n
(gdb) set variable a = 1
(gdb) set variable b = 1
(gdb) set variable b = 0
```

In ther first time, there is no permission to execute in the given file. So, in order to execute the file with `gdb` tool, I use `chmod` command with argument `777`.

In the process of debugging the executable file with the `gdb` tool, I first set breakpoint for the `main` function and used `run` command to get to the `main` function point. In the `main` function, I used the `n` command to execute statements in turn.

The given program started the `main` function at line 5, moved through lines 7, 9 and ended after moving to lines 48, 49. Therefore, I thought there would be a point between line 9 and line 48 to know the flag, and I thought that changing the values of variable `a` and `b`, which suggested in the hint, would move to a blocked branch. So I ran the commands `set variable a = 1` and `set variable b = 1` on line 9 to change the value of the variable, and then moved to line 11 by using `n`.

There were a lot of loops in the program. When I used `n` to move to the next statement, I thought that the process were stuck in a loop if the line number kept repeating. So in each that time, I changed the value of the variable `b` to 0 or 1, so that I could go out of the loop. After passing all the loops and deviating to 42,43 lines, I observed the flag value.

4. Demo

> Find the flag by decompiler (e.g., ghidra) reverse-engineering (e.g., using objdump) the given executable!

I solved his problem after the end of the Quiz time.

Command

```
$ chmod 777 ./4    # give permission to execute
$ gdb ./4
```

```
(gdb) disas main
(gdb) b *main+84            # A line before return main function
(gdb) x/bs 0x55555555a760   # Address of flag
```

To solve this problem, I disassemed the `main` function of this executable using the `disas` command. I set the break point at the last statement of the `main` function because the program ends immediately after the `run` command. After that, we ran the program again with the `run` command and checked the `flag` with the `x/bs` command, and I confirmed that data was added to the `flag` array. The answer is `i_love_cse467`.