Lab #1. Warm-up Exercise

Prof. Jaeseung Choi

Dept. of Computer Science and Engineering

Sogang University

About the Labs

- **Lab assignments will count for 25% of the total score**
  - You will be asked to find, analyze and exploit vulnerabilities in target programs

$18\% \times \frac{9}{10}$

(22.5/25%)

- **We will have three lab assignments**
  - **Lab #1:** Warm-up exercise (5%) $\rightarrow$ 4.5%
  - **Lab #2:** Buffer overflow exercise (10%) $\rightarrow$ 9%
  - **Lab #3:** Advanced exploits and other vulnerabilities (10%) $\rightarrow$ 9%
- **Today: Lab #1 (Warm-up exercise)**
  - Related to *"Chapter 2. Introduction to Software Vulnerability"*
  - Will get you familiar with the skeleton code and problem style
- **Remind: you'd better practice using `Putty` and `vim`**
  - In the lab exam, you are not allowed to use `Visual Studio`, etc.

General Information

- **Check "Lab #1" in *Assignment* tab of *Cyber Campus***
  - Skeleton code (**`Lab1.tgz`**) is attached in the post
  - Deadline: **9/26** Thursday 23:59
  - Submission will be accepted in that post, too
  - Late submission deadline: **9/28** Saturday 23:59 **(-20% penalty)**
  - Delay penalty is applied uniformly **(not problem by problem)**
- **Please read the instructions in this slide carefully**
  - This slide is step-by-step tutorial for the lab
  - It also contains important submission guidelines
    - If you do not follow the guidelines, you will get penalty

Skeleton Code Structure

- **Copy `Lab1.tgz` into CSPRO server and decompress it**
  - You **must connect to** `csproN.sogang.ac.kr` **(N = 2, 3, or 7)**
  - Don’t decompress-and-copy; copy-and-decompress
- **`1-1~1-3` : Each directory contains a problem to solve**
- **`check.py` : Self-grading script (explained later)**
- **`config` : Used by grading script (you don’t have to care)**

```
jschoi@cspro2:~$ tar -xzf Lab1.tgz
jschoi@cspro2:~$ ls Lab1
1-1  1-2  1-3  check.py  config
```

Problem Directory (Example: 1-1)

- **bank.c : Source code of the target program to exploit**
- **bank.bin : Compiled binary of the target program** (취약점가지고 있다)
- **secret.txt : Your goal is to read the content of this file**
  - Assume that you cannot directly read **secret.txt**
  - You must exploit **bank.bin** and make it print **secret.txt**

이거를 출력해야 목표

- **exploit-bank.py : You will write your code here**

```
jschoi@cspro2:~$ cd Lab1/1-1/
jschoi@cspro2:~/Lab1/1-1$ ls
bank.bin  bank.c  exploit-bank.py  secret.txt
```

Target Program

- **You can execute the target program and interact with it**
  - Analyze the provided source code carefully
  - By providing unexpected inputs, you can make it malfunction
  - Fool the program to make it print the content of secret file

```
jschoi@cspro2:~/Lab1/1-1$ ./bank.bin
=============================
[SYSTEM] Your balance = 1000
[SYSTEM] What is your choice?
1. Send money to Alice
2. Read secret file
3. Quit
(Enter 1~3): 2
[ERROR] Only the VIP user can read the secret file
...
```

Your input

Writing Exploit Code

- **Next, translate your actions into the form of code**
  - Fill in the **`exploit-bank.py`** script (skeleton code is given)
  - Using **`Pwntools`** library, you can interact with a program easily
    - You can create an object with **`process(…)`** and call methods
    - To avoid subtle issues, use **`bytes`** type instead of **`str`** type (put the **`b`** prefix in front of a string like **`"blah"`**)

```python
from pwn import *

def exploit():
    p = process("./bank.bin")
    # Read in the menu messages.
    for i in range(6):
        print(p.recvline())
    print(p.recvuntil(b"(Enter 1~3): "))
    p.sendline(b"1") # Choose "1. Send money"
```

string

Methods in `Pwntools` Library

- **There are various methods you can use to interact with the target program**
  - **`recvline():`** read program output until a newline (**`\n`**) is met
  - **`recvuntil(s):`** read until string (**`bytes`** type) **`s`** is met
  - **`recv(n):`** read up to **`n`** bytes
  - **`send(s):`** send string (**`bytes`** type) **`s`** to the program
  - **`sendline(s):`** send **`s`** and newline (**`\n`**) to the program
  - … and many more in the reference document*

*** https://docs.pwntools.com/en/stable/tubes.html#pwnlib.tubes.tube.tube**

Self-grading Your Exploit

- **You can run `check.py` to test if your exploit code can successfully print out the content of `secret.txt`**
  - **"`./check.py`"** will check the exploits for problems one by one
  - Symbols in the result have the following meanings
    - `'O'`: Success, `'X'`: Fail, `'T'`: Timeout, `'E'`: Exception

```
jschoi@cspro2:~/Lab1/$ ls
1-1  1-2  1-3  check.py  config
jschoi@cspro2:~/Lab1/$ ./check.py
[*] 1-1 : O
[*] 1-2 : X
[*] 1-3 : X
```

Don’t do this

- **You may feel tempted to hard-code the string stored in `secret.txt` or directly access it from your exploit code**
  - Of course, that’s not the intention of this lab
  - Even if you pass `check.py`, you will get **0 point** in real grading

```python
def exploit():
    # Maybe I can do this?
    print("Secret file content is: f0ae07cd")

    # Or something like this?
    f = open("secret.txt")
    print(f.read())
```

Hints

- **The point of this lab assignment is to think in the shoes of an adversary (hacker)**
  - Try to think of a creative inputs that can break the program
- **The problems are closely related to the materials in *"Chapter 2. Introduction to Software Vulnerability“***
  - So review that lecture note before you start this lab
- **Reference that may help you in problem 1-2**
  - https://www.gnu.org/software/bash/manual/html_node/Shell-Commands.html
  - If you have no clue at all, skim through this webpage

Problem Information

- **Three problems in total**
  - Problem 1-1: **30 pt.**
  - Problem 1-2: **30 pt.**
  - Problem 1-3: **40 pt.**
- **You'll get the point for each problem if the exploit works**
  - **No partial point for non-working exploit**
- **For Lab #1, analyzing the source code is enough**
  - Don't need any code analysis at assembly-level

Submission Guideline

- **You should submit the three exploit script files**
  - Problem 1-1: `exploit-bank.py`
  - Problem 1-2: `exploit-list.py`
  - Problem 1-3: `exploit-logger.py`
- **No report required for Lab #1**
- **Submission format**
  - Upload these files directly to *Cyber Campus* (**do not zip them**)
  - **Do not change the file name** (e.g., adding any prefix or suffix)
  - If your submission format is wrong, you will get **-20% penalty**