Lab #1.
Warm-up Exercise

Prof. Jaeseung Choi

Dept. of Computer Science and Engineering

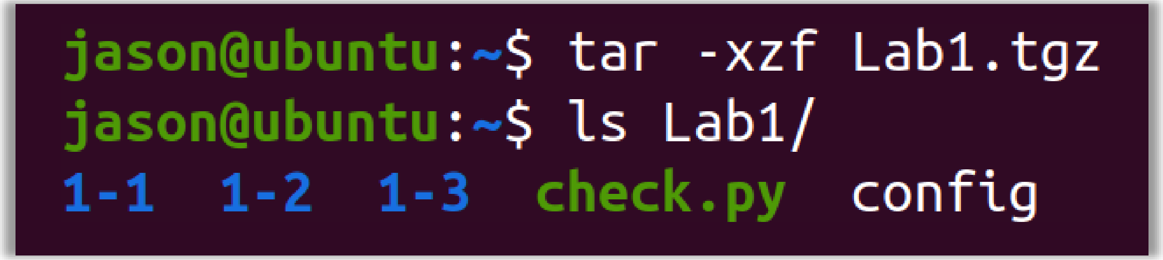
Sogang University

General Information

- **Check "Lab #1" in *Assignment* tab of *Cyber Campus***
  - Skeleton code (`Lab1.tgz`) is attached in the post
  - Deadline: **9/26** Tuesday 23:59
  - Submission will be accepted in that post, too
  - Late submission deadline: **9/28** Thursday 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

Preparing the Environment

- **Copy `Lab1.tgz` into CSPRO server and decompress it**
  - Remember that we have changed the plan to use CSPRO until Lab #5 or Lab #6 (in November)
  - So you **must use CSPRO** (not your own Linux machine)
    - Connect to cspro**5**.sogang.ac.kr (**don’t miss the "5"**)
  - Don’t decompress-and-copy; copy-and-decompress
- **`check.py`: Self-grading script (explained later)**
- **`config`: Used by grading script (you don’t have to care)**

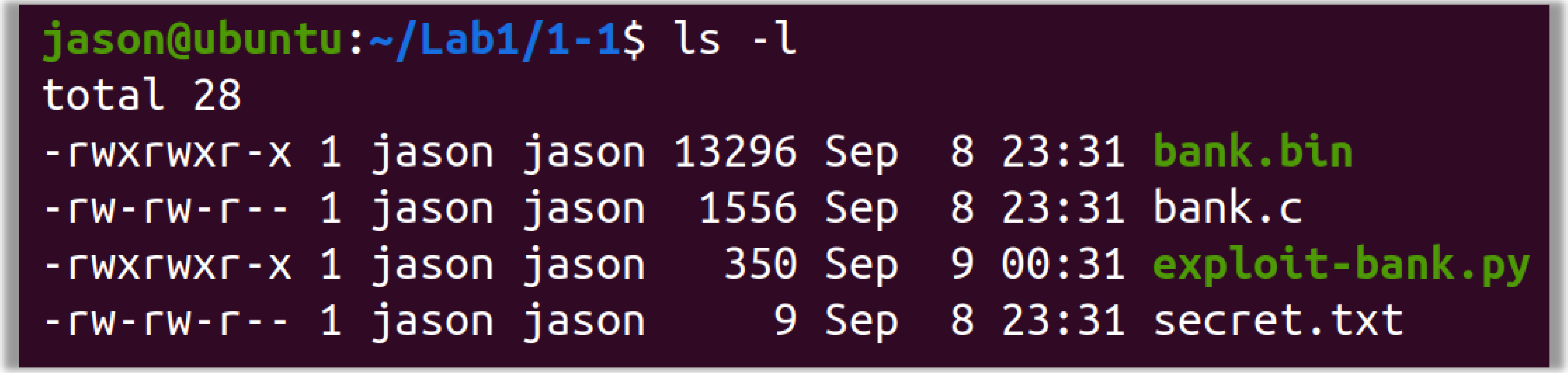


```
jason@ubuntu:~$ tar -xzf Lab1.tgz
jason@ubuntu:~$ ls Lab1/
1-1  1-2  1-3  check.py  config
```

Problem Directory (Example: 1-1)

- **`bank.c` : Source code of the target program to exploit**
- **`bank.bin` : Compiled binary of the target program**
- **`secret.txt` : Your goal is to read this file**
  - Assume that you cannot directly read **`secret.txt`**
  - You must exploit **`bank.bin`** and make it print **`secret.txt`**
- **`exploit-bank.py`: You will write your exploit here**

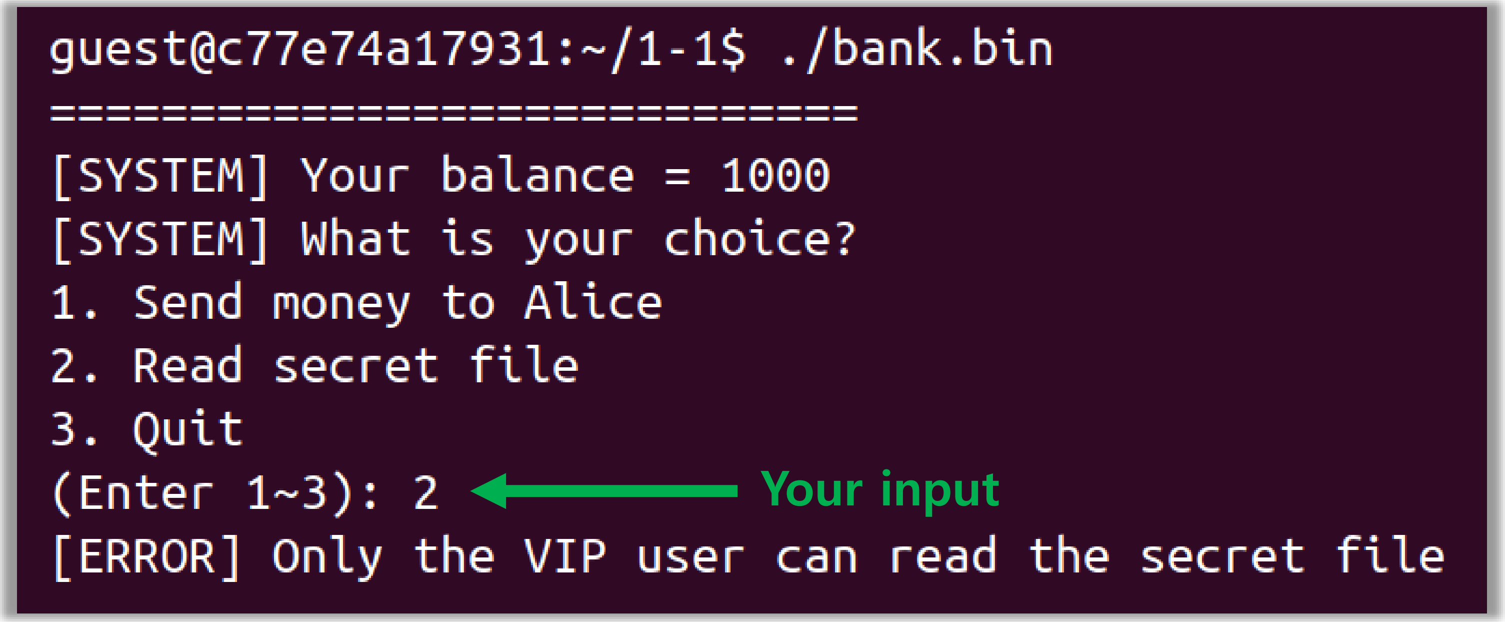
(제출)



```
jason@ubuntu:~/Lab1/1-1$ ls -l
total 28
-rwxrwxr-x 1 jason jason 13296 Sep  8 23:31 bank.bin
-rw-rw-r-- 1 jason jason  1556 Sep  8 23:31 bank.c
-rwxrwxr-x 1 jason jason   350 Sep  9 00:31 exploit-bank.py
-rw-rw-r-- 1 jason jason     9 Sep  8 23:31 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 and obtain the content of secret file



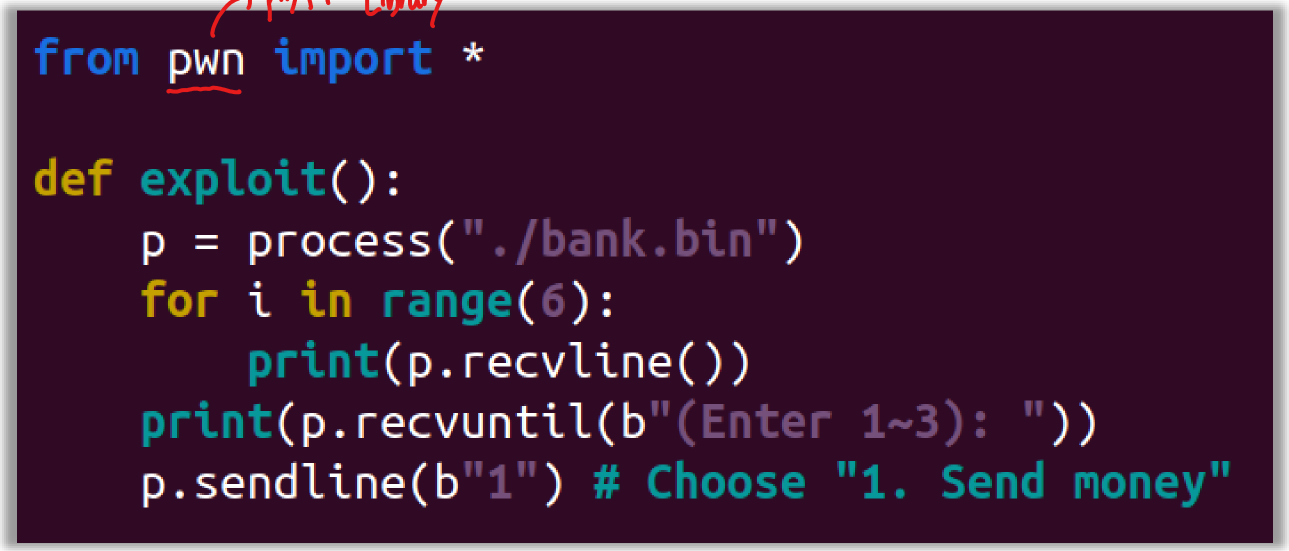
```
guest@c77e74a17931:~/1-1$ ./bank.bin
=============================
[SYSTEM] Your balance = 1000
[SYSTEM] What is your choice?
1. Send money to Alice
2. Read secret file
3. Quit
(Enter 1~3): 2
[ERROR] Only the VIP user can read the secret file
```

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
    - Various methods*: **`recvline, recvuntil, sendline`**
    - To avoid subtle issues, use **`bytes`** type instead of **`str`** type (put the **b** prefix in front of **`"blah"`**)

→ phython Library



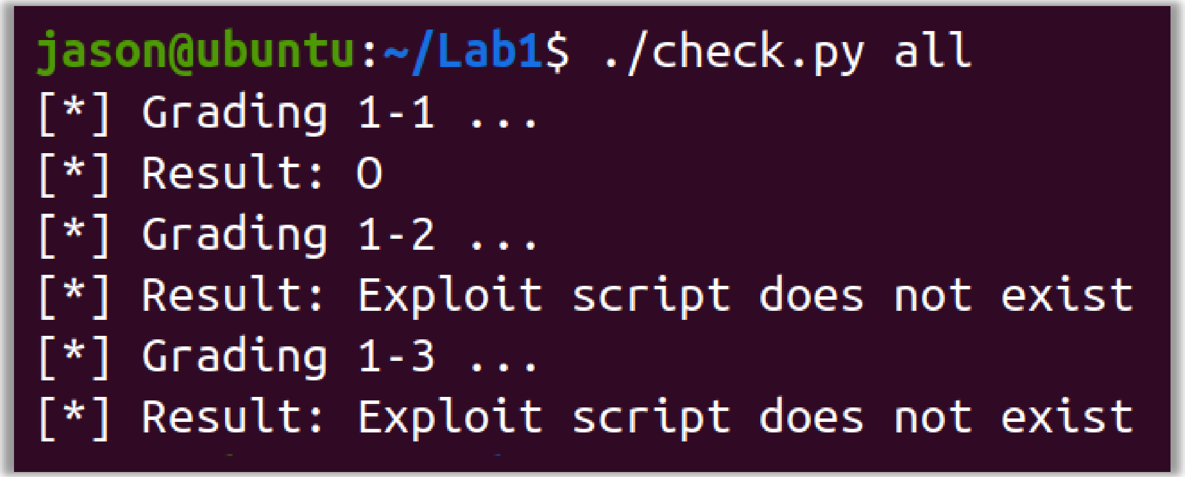
```
from pwn import *

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

* 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`**
  - It assumes that the exploit scripts are stored under **`share/`**
  - `"./check.py 1-1"` will check the exploit for problem 1-1
  - `"./check.py all"` will check the exploits for all the problems
  - Symbols in the result has the following meanings
    - `'O'`: Success, `'X'`: Fail, `'T'`: Timeout, `'E'`: Exception



```
jason@ubuntu:~/Lab1$ ./check.py all
[*] Grading 1-1 ...
[*] Result: O
[*] Grading 1-2 ...
[*] Result: Exploit script does not exist
[*] Grading 1-3 ...
[*] Result: Exploit script does not exist
```

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

```
def exploit():
    # Maybe I can do this?
    print("Secret file content is: f0ae07cd")
    # Or something like this?
    f = open("secret.txt")
    print(f.read())
```

✗

↳ 이런 코드를 어떻게 방어할지?

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**
  - Assembly-level analysis is not required

Submission Guideline

- **You should submit the three exploit script files**
  - Problem 1-1: `exploit-bank.py`
  - Problem 1-2: `exploit-logger.py`
  - Problem 1-3: `exploit-mileage.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**