Software Security Lab

Introduction

We chose 3 typical topics introduced in class as the challenges of software security lab, they are:

1. Task 1 – Buffer overflow.
2. Task 2 – Integer overflow.
3. Task 3 – Time-of-check-to-time-of-use.
4. Task 4 – User management application

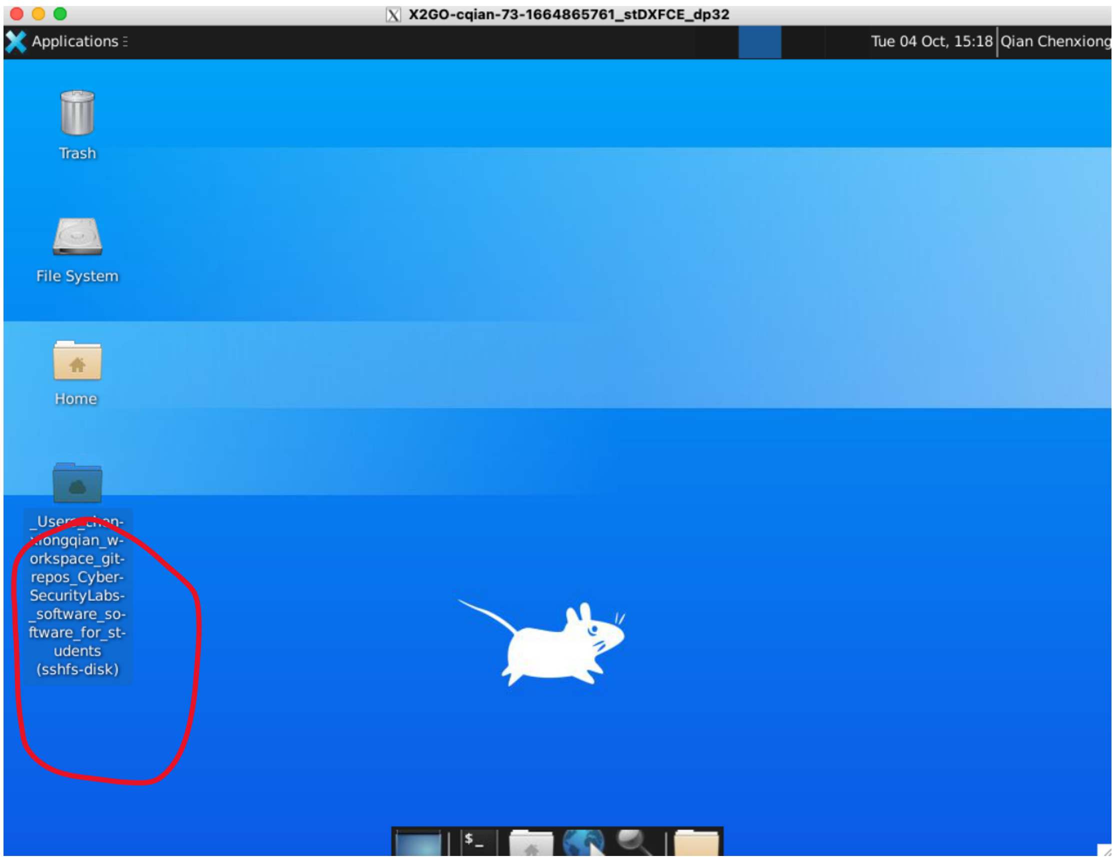
In this assignment, please read the details of each task to figure out the specific requirements. For each task, we provide you with the source code of the program and the binary compiled from the given source code. You are required to exploit the vulnerability hidden in the program to finish each task. The source code is provided to help you understand the program and might not be entirely the same as the source code we used to compile the executable binary. Therefore, you should not recompile the binary from the source code to run. **You are supposed to run the binaries that we compiled on the CS server or your own Unix-based computer.**
To get your account to use the CS server, please visit:
https://intranet.cs.hku.hk/csintranet/newstudent.jsp

Notice

Server Connection and File Sharing

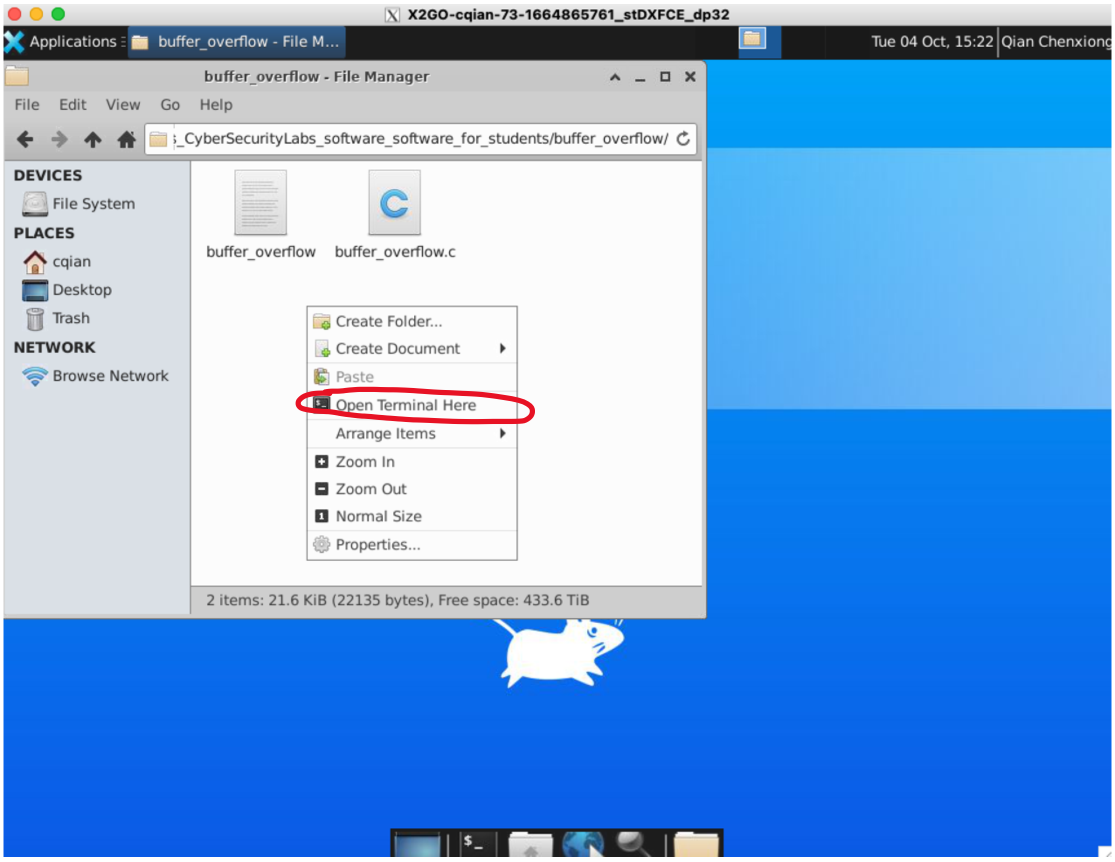
You should connect to the CS server and run the tasks and get screenshots of them. Please refer https://moodle.hku.hk/mod/resource/view.php?id=2694891 to connect the server. After you connect the server, please refer https://intranet.cs.hku.hk/csintranet/contents/technical/howto/x2go/filesharing/x2goclient-file-sharing.jsp to share the files in this assignment with the server.

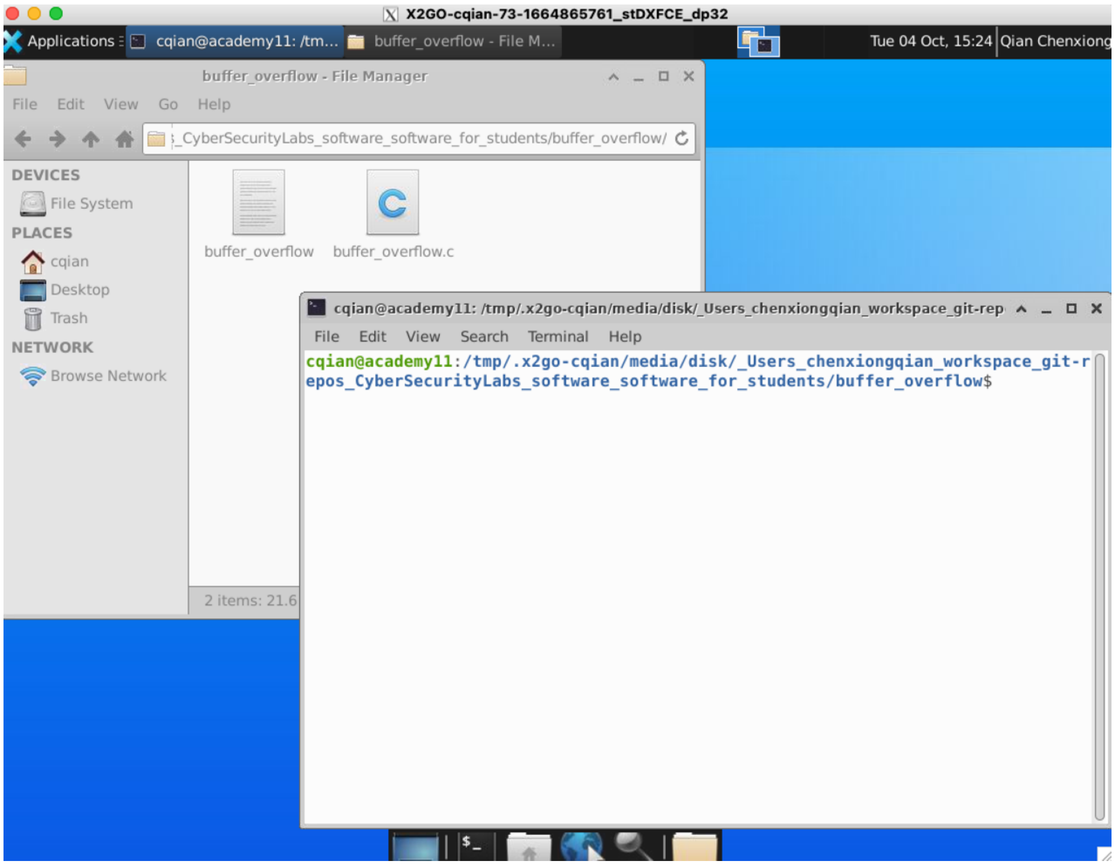
After doing that, you will see a folder appear on the desktop.


X2GO-cqian-73-1664865761_stDXFCE_dp32
Applications
Tue 04 Oct, 15:18
Qian Chenxiong
Trash
File System
Home
_Users_chen-xiongqian_w-orkspace_git-repos_Cyber-SecurityLabs-_software_so-ftware_for_st-udents (sshfs-disk)

Run the programs

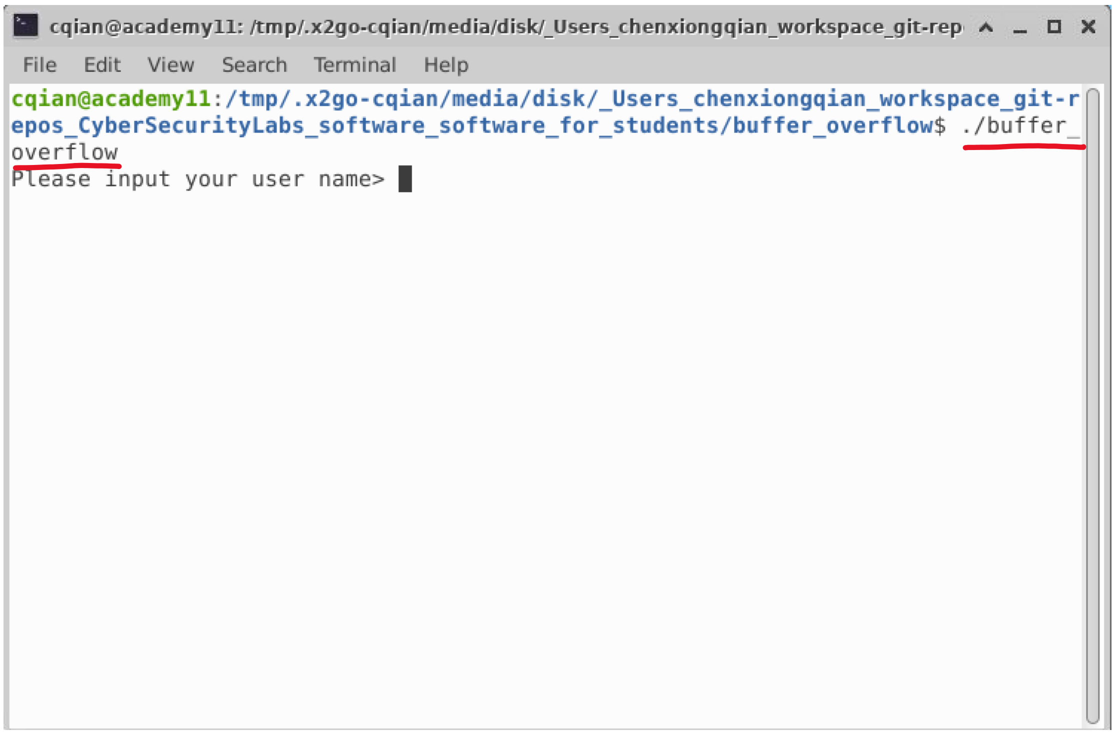
To run the programs, go into each task’ s folder. Then, right click to choose “Open Terminal Here” , which will open a terminal and locate to the particular task’ s folder.




X2GO-cqian-73-1664865761_stDXFCE_dp32
Applications
cqian@academy11: /tm...
buffer_overflow - File M...
Tue 04 Oct, 15:24
Qian Chenxiong
buffer_overflow - File Manager
cqian@academy11: /tmp/.x2go-cqian/media/disk/_Users_chenxiongqian_workspace_git-rep
File
Edit
View
Search
Terminal
Help

```
cqian@academy11:/tmp/.x2go-cqian/media/disk/_Users_chenxiongqian_workspace_git-r
epos_CyberSecurityLabs_software_software_for_students/buffer_overflow$
```

After that, you can type the command in the terminal to run the binary program. E.g., typing the command "./buffer_overflow" will start running the ***buffe_overflow*** program.


cqian@academy11: /tmp/.x2go-cqian/media/disk/_Users_chenxiongqian_workspace_git-rep
File Edit View Search Terminal Help

```
cqian@academy11:/tmp/.x2go-cqian/media/disk/_Users_chenxiongqian_workspace_git-r
epos_CyberSecurityLabs_software_software_for_students/buffer_overflow$ ./buffer_
overflow
Please input your user name> 
```

Submission

Please submit a report **in the format of PDF** to answer the questions listed in each task.

Task 1 – Buffer Overflow (20 Points)

This lab operates as a “login” program. At the very beginning, it will create some stack variables to store the user information:

```
char user_name[48] = {0};
char password[48] = {0};
char flags[48] = {0};
```

It requires you to input user’ s name and password for further use:

```
printf("Please input your user name> ");
scanf("%s", user_name);
printf("Please input your password> ");
scanf("%s", password);
```

After that, the program will check the value of ***user_name*** and the hash value of ***password***. If it is correct, the program will set the ***flags*** to a magic string:

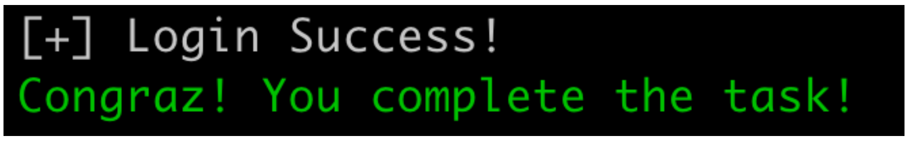
```
#define HASH_OF_CORRECT_PASSWORD 0x4b665b0

if (strncmp(user_name, "CoatOfArms", 10) == 0 &&
    getHash(password, 32) == HASH_OF_CORRECT_PASSWORD) {
    strncpy(flags, "AcCes50k", 8);
}
```

And after that, the program will check the flags, and if it equals to the magic string value (i.e., the ***user_name*** and ***password*** are correct), it will call the ***winner_handler,*** which means you complete the task:

```
if (strncmp(flags, "AcCes50k", 8) == 0) {
    printf("[+] Login Success!\n");
    winner_handler();
}
```

The function ***getHash*** is to get the hash value of a certain string and ***HASH_OF_CORRECT_PASSWORD*** is the hash value of the correct password.

Your goal in this task is to do the login successfully. If you see the following output in the terminal, you complete the task.



```
[+] Login Success!
Congraz! You complete the task!
```

Report (20 Points)

- Please highlight the vulnerable code (**5 points**).
- Please explain how the vulnerability happens. (**5 Points**)
- Please explain the steps that you use to exploit the vulnerabilities to win the game. (**10 Points**)

Please attach or draw figures to make the demonstration clearer.

Task 2 – Integer Overflow (25 Points)

This is a game designed by a newbie game developer, who wants to make it impossible for you to win!

At the beginning of the game, you can choose one of the two characters (hero Mage or hero Slayer) to start your adventure, they have different powers and skills for you to fight with the boss Guardian.

The following shows the detailed properties of Mage, Slayer, and the Guardian.

Mage:

- The initial health and mage points are 400000000 and 500, respectively.
- The Mage has three skills:
  1. Burn. It causes 200000000 damage points to the Guardian and costs 100 mage points.
  2. Potion. It refreshes the mage points to 500 points.
  3. Water Shield. It causes 25 mage points and decrease the damage from Guardian by a factor of 10 for a round.

Slayer:

- The initial health point is 400000000 points, and Slayer does not have any mage points.
- The Slayer has two skills:
  1. Slash. It causes 200000000 damage points to the Guardian.
  2. Deadly Slash. It causes 500000000 damage points to the Guardian and 200000000 damage points to self.

Guardian:

- The initial health point is 2000000000 points, and the Guardian does not have any mage points.
- In each round, the Guardian causes 100000000 damage points to the hero and heals itself with 5000000 health points.

The fight would proceed with rounds. In each round, the Guardian will attack you and heal itself. You are asked to choose one skill to fight against the guardian. After each round, the Guardian’ s health point will be updated, so will the hero’ s health point and mage point. If the HP of Guardian is less or equal than 0, you win the game. On the contrary, if your HP is equal to or less than 0, you lose the game☹.

Your goal in this task is to win the game using both heroes.

Report (25 Points)

- Please find and explain the vulnerability that you exploit to win the game using Mage. (**5 points**)
- Please demonstrate the steps to win the game using Mage with screenshots of each round. (**5 points**)
- Please find and explain the vulnerability that you exploit to win the game using Slayer. (**5 points**)
- Please demonstrate the steps to win the game using Slayer with screenshots of each round. (**5 points**)
- Pease explain how to fix the vulnerabilities. (**5 points**)

Task 3 – Time-of-check-to-time-of-use (15 Points)

Race condition problems often come up when we are dealing with parallel programs. In class we have discussed about the model of Time-of-Check & Time-of-Use, and this task is also built on the TOC & TOU model.

This is also a login program demo similar to task 1.

At first the program will open the files "**./user_name.txt**" and "**./password.txt**" and check the file sizes of them, and if any of the sizes is larger than 32 bytes, it will throw an error and exit:

```
int user_name_fd = open("./user_name.txt", O_RDWR);
if (user_name_fd == -1) {
    perror("open user_name.txt");
    exit(-1);
}

int password_fd = open("./password.txt", O_RDWR);
if (password_fd == -1) {
    perror("open password.txt");
    exit(-1);
}

if (get_file_size("./user_name.txt") > 48 ||
      get_file_size("./password.txt") > 48) {
    printf("File size too large.\n");
    exit(-1);
}
```

After that, it will **sleep for 10 seconds** for no reason 😂:

```
sleep(10);
```

Then, the program starts to read ***user_name*** and ***password*** from the files:

```
int i = 0;
char c;
while (read(user_name_fd, &c, 1) != -1) {
    if (c == '\n') break;
    user_name[i] = c;
    i++;
}
printf("Read %d bytes from user_name.txt\n", i);

i = 0;
```

```
while (read(password_fd, &c, 1) != -1) {
    if (c == '\n') break;
    password[i] = c;
    i++;
}
printf("Read %d bytes from password.txt\n", i);
printf("user_name ==> %s\n", user_name);
printf("password ==> %s\n", password);
```

After that, the program will check the value of ***user_name*** and the hash value of ***password***. If it is correct, the program will set the ***flags*** to a magic string. Next, the program will check the ***flags***, and if it equals to the magic string value (that is, the ***user_name*** and ***password*** are correct), it will call the ***winner_handler,*** which means you complete the task

```
if (strncmp(user_name, "We1s0n", 6) == 0 &&
    getHash(password, 32) == HASH_OF_CORRECT_PASSWORD) {
    strncpy(flags, "AcCes50k", 8);
}

if (strncmp(flags, "AcCes50k", 8) == 0) {
    printf("[+] Login Success!\n");
    winner_handler();
}
```

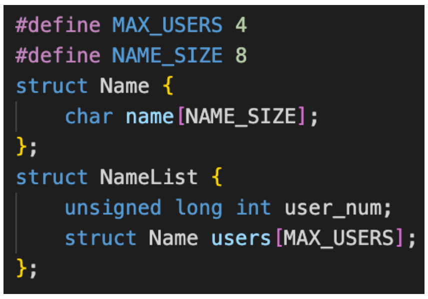
Your goal in task is to use TOCTTOU attack to login successfully.

Report (15 Points)

- Please explain the vulnerability that you exploit to launch the attack. (**5 points**)
- Please take screenshots to demonstrate the steps that you use to win the game. (**5 points**)
- Please propose a patch to fix the code to eliminate the vulnerability. (**5 points**)

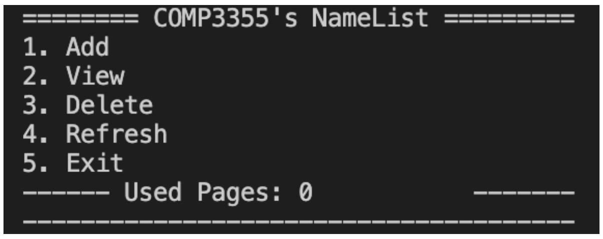
Task 4 – User management application (40 Points)

This is a user management application. You can use it to add or delete usernames. As shown in the given source code:
The `NameList` structure contains two fields: `user_num` is how many usernames are there in the name list; `users` contains the data of each username, which is a `Name` structure.
The `Name` structure contains one field `name`, which is simply an array of data of a username. It is designed to have maximumly 4 users and 8 characters in a name.



```c
#define MAX_USERS 4
#define NAME_SIZE 8
struct Name {
    char name[NAME_SIZE];
};
struct NameList {
    unsigned long int user_num;
    struct Name users[MAX_USERS];
};
```

The application menu shows available options and how many usernames are there currently. Then it starts to take integer inputs representing which options you want to enter:



```
======== COMP3355's NameList =========
1. Add
2. View
3. Delete
4. Refresh
5. Exit
------ Used Pages: 0          --------
--------------------------------------
```

You can add, read, delete usernames, clear and refresh the whole name list, or exit the program by typing the option number (1~5):

- When adding a username, it reads your input and append it to the current available place (`users[user_num]`). The index starts from 0.
    - In the computer' s memory, characters ('ABCD…') are stored as corresponding numbers (in hex, A: 41, B: 42, …).
    - **Our user list takes hex numbers in one line without spaces** as inputs when adding users, and outputs both the hex number and their characters, but some characters are unprintable.
    - The program also properly handled the endianness, you can directly type hex numbers and check what is shown. To know more, search about "ASCII code" and "endianness" .
- When starting the application, it asks you to input a password, if you input the correct password, you can enter the admin mode to get a secret. But the password is randomly generated every time, which you will never know.

Your job is to carefully audit the given code to find vulnerabilities and try to exploit them to get the admin secret by running the given program!

Report (40 Points)

- Please point out two vulnerabilities in the source code. (**10 points**: 5 points for each vulnerability)
- Please use **screenshots** to demonstrate how to trigger the two vulnerabilities and **explain** what abnormal behaviors they would cause (use what you have learned about the memory layout in the course). (**10 points:** 5 points for each vulnerability)
- Please **describe** how to exploit the vulnerabilities to get the secret messages. (**10 points**)
- Please exploit the vulnerabilities to get the secret messages by the running program. Provide **step-by-step screenshots** to demonstrate the exploitation. (**10 points**)