Assignment 2: Computer Engineering Case Study – Game Memory Hacking

Available: Friday October 25, 2024.
Due: Sunday November 10th @ 11:55pm (Submit via the Brightspace submission system)

Programming aspects to get familiarized with:

- Pointers,
- Dynamic memory allocation,
- Multidimensional arrays,
- Subprograms/Functions.

Problem Statement:

In the field of computer engineering, pointers and data manipulation are key concepts, especially when applied to real-world scenarios like gaming. In many video games, players can face obstacles such as not having enough health or money, which can make the game challenging or even frustrating. To overcome these difficulties, some tools, like Cheat Engine, have been developed. These tools work by accessing and changing the game's data directly in its memory. This involves using pointers to locate and modify the exact place in the computer's memory where this game data is stored, such as the values for health and money. In this assignment, you are challenged to engineer a game hacking program in C++ that manipulates and controls real-time data and UI elements of a given game binary.

Due to the technical nature of creating a game memory hacking program, which falls under the category of potentially malicious software, it is imperative to operate within a controlled environment. Compiling and running such software on common operating systems like Windows or Mac could lead to the program being flagged as a virus by security software, hence we need to run it in Ubuntu Linux Operating system.

Step-by-Step Guide for Game Hacking Setup

If you have a Linux machine, you don't need any additional setups. However, if you have either Windows or Mac, you have to compile your code on NYU Linux servers as follows.

You can watch the full setup procedure here: https://www.youtube.com/watch?v=AfK-zg5xAZ4

1. Install VS Code:
   VS Code (https://code.visualstudio.com) is a popular code editor. Install it by visiting the linked page and following the instructions for your operating system.

2. Connect to NYU Abu Dhabi VPN:
   You need to connect to the NYU Abu Dhabi VPN to access the computer resources remotely, if you are off campus. Make sure you have installed the NYU VPN client and logged in with your NYU credentials.

3. Open Terminal:

Windows: Press `Windows Key + R`, type `powershell`, and press `Enter`.
Mac: Press `Command + Space` and search "Terminal".

4. SSH into NYU Server:
   In the terminal, type the following command to connect to the remote server (replace pk2269 with your NYU net ID):

```
ssh pk2269@10.230.10.7 -p 4410
```

You will need your NYU password to log in.

Repeat this step with a **new terminal window**. One for running the game, one for running and compiling the hacker program.

5. Download Required Files and Libraries:
   Once connected, download the necessary files and libraries by typing:

```
wget https://github.com/Pi-31415/CPE/raw/main/CPE.zip && unzip CPE.zip
```

This command downloads and extracts the libraries required for game hacking.
You can list the files you have with the following command at any point.

```
ls
```

6. Create and Write Code in **`hacker.cpp`** **on your Desktop**:
   Windows:
   Right-click on your desktop, select **New** > **Text Document**, and rename it to `hacker.cpp`.
   Mac:
   Open **VS Code,** create a new file, and save it as `hacker.cpp` on your desktop.

   Open `hacker.cpp` in **VS Code**:
   Windows: Right-click the file and select **Open with Code**.
   Mac: Right-click the file, choose **Open With**, and select **Visual Studio Code**.

   Add the following initial code to `hacker.cpp`:

```
#include <iostream>
#include <fstream>
#include "hacking_functions.h"
```

   These libraries are necessary for input/output and to use pre-written hacking functions.

Check the contents of `hacking_functions.cpp` at this link (https://github.com/Pi-31415/CPE/blob/main/CPE/hacking_functions.cpp).

7. Upload **`hacker.cpp`** to Linux Server (you need a separate terminal for uploading the source code):

   In a new terminal, to upload the file to the server, type this command in your terminal (replace pk2269 with your NYU net ID):

```
scp -P 4410 "$HOME/Desktop/hacker.cpp" pk2269@10.230.10.7:/home/pk2269/
```

   This command transfers the `hacker.cpp` file from your computer to the remote server.

8. Run the Game:

   Before compiling your hacker program, run the game by typing:

```
./game
```

   This ensures the game is running before you try to hack it.

9. Compile and Run Your Hacker Program:

   After editing and saving `hacker.cpp` in **VS Code**, upload it to the server using the previous `scp` command.

   To compile your program, type

```
g++ -o hacker hacker.cpp hacking_functions.cpp -std=c++11
```

   Then, run the program with the following command (you need sudo):

```
sudo ./hacker
```

   You should have **three terminal windows** open:

   1. **Game Window**: Running the game.
   2. **Compiling Window**: For compiling and running your hacker program.
   3. **Uploading Window**: For uploading the code after each edit.

You need to upload and compile it (Step 7 to 9) every time you make a change to the code.

If your code is correct, you should get this result – Full Game Hacking Demo (https://www.youtube.com/watch?v=QI1qnqD5p9k&ab_channel=PiKo).

The Game:

Once the game is run, it will output a file called **gameinfo.txt**. This contains the required information you need to hack the game.

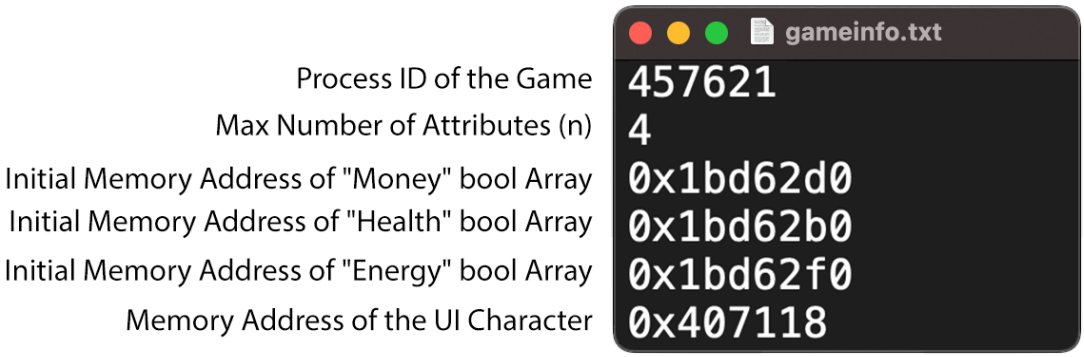

Figure 1: Contents of `gameinfo.txt` file, which the game generates on run-time.

The game includes 2 types of data - the game statistics (Boolean) and the UI Character (char). The Boolean data are implemented as arrays in the game for 3 different attributes (Money, Health and Energy). The maximum number of these attributes is determined at run-time, so you have to read the second line in **gameinfo.txt** to obtain this info. Since the data type is Boolean, the next memory address is obtained by just adding 1, once you know the initial address of the array.

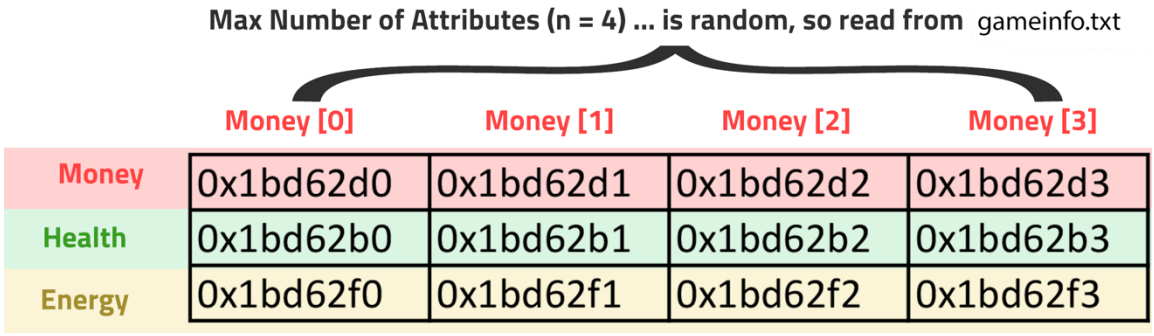

Figure 2: Example Memory Layout of the Game (Addresses may differ).

The game also has a character for UI (default is *) which stores what character to display the game attributes with. The address of this char is the last one in gameinfo.txt. Your second task is to overwrite this char and hack the UI of the game from * to something else on keyboard, that the user can choose.

Develop a software solution, which will do the following:

- Must read from "`gameinfo.txt`", which contains critical runtime information including the process ID and memory addresses for game variables like money, health, and energy.
- Include the library necessary for memory manipulation by adding the following line at the beginning of your hacker.cpp file, like this:

```
#include "hacking_functions.h"
```

- Read the contents of gameinfo.txt, then **dynamically** allocate a 2D array to store the game values read from gameinfo.txt.
- Implement **UI Hacking Mode**, where you identify and modify the character to display the game attributes.
- Implement **Data Hacking Mode** to alter game statistics. Display current values, allow user input for new values, and write these changes into the game's memory. Pay attention to the max number of Attributes (n). For instance, if the health is currently 1 (one *), and the max n is 8, the user should be able to enter the new health value anywhere between 0 to 8 and have that number of asterisk (*) displayed in the health bar in the game program. Note that the game program updates itself, and automatically detects whether your hack is successful or not).
- At the start of your program, after reading the value of pid (process ID) from gameinfo.txt, call the function `attachToProcess(pid)` from the library. This function attaches your program to the game process, allowing you to modify its memory.
- For Integers, any modifications to the contents of any integer address can be done using `modifyValue(pid, address, new_value).` If you need to retrieve any existing data from that address, use the function `readValue(pid, address).`
- For UI Character, any modifications to the contents of any char address can be done using `writeUIChar(int target_pid, long address, char new_char).` If you need to retrieve any existing data from that address, use the function `readUIChar(int target_pid, long address).`
- Implement a user-friendly menu to switch between the hacking modes.
- Ensure the program can gracefully attach to and detach from the game process using the provided library functions.
- You are not allowed to use advanced data structures for developing your solution.
- Make sure your code is well commented.
- Add any additional subprograms if you need to.
- You can make additional assumptions in case you feel any information is missing. Make sure to clearly state them.

Submission

Your submission **must** include an assignment report (Word Document or PDF) and a C++ source code (.cpp file). The report must follow the five steps model for software development (as discussed in class):

a) Step 1: Problem Identification and Statement (5 points)
b) Step 2: Gathering Information (10 points)
c) Step 3: Test Cases and algorithm (35 points)
d) Step 4: Code or implementation (35 points)
e) Step 5: Test and Verification (a minimum of 4 test cases) (15 points)