Part 1 – Github Setup

In order to generate signed commits I first generated a GPG keypair. This was used with git in order to sign all commits for this project. Additionally, I had to add my GPG public key to the Github interface to ensure that my commits would show as 'verified.' Afterwards, I prepared my Github actions configuration. I configured it so it will acquire the files needed from my repository and then call the makefile to compile and run the programs including the test cases.

Part 2 – Bugs

The next step was to examine the source code of the gift card reader program. As noted in the instructions, there are comments leading to indications of potential bugs.

The first issue I noticed is the value of the 'num_bytes' variable, which is the first item read from the gift card, is used to allocate memory for a pointer. There were no checks conducted prior to this operation. I generated a gift card with a negative value for this variable which resulted in a segmentation fault. To fix this issue, I added an if statement prior to the memory allocation to determine if the gift card contains a variable 'num_bytes' with a negative value. If it does, then the program will print an error and fail gracefully. It would be better for the program to end without performing the proper task than crash entirely or potentially manipulate memory maliciously.

The next two issues I discovered were involving the animate function in the card reader. I noticed that as part of the included program on the gift card two arguments would be passed as part of the '0x04' operation which manipulated the registers. Again, there were no checks on these values. I tested passing different values into this operation until I received a segmentation fault. I believe this overwrites the return pointer of the reader. As a fix, I changed it so that the register can only be changed if the value is less than what caused this crash.

The third problem I found, also with the animate function, is with the '0x10' operation. I noticed that the arg1 variable is cast to a char after being declared an unsigned char. The pointer 'pc' is also an unsigned char. If arg1 is signed then I could pass a value to it causing it to loop backwards through negative numbers. At the end of the switch 'pc' is incremented by 3. I passed a hex value to arg1 of '-3' which would be countered by the following incrementation. This would result in an infinite loop. In order to get to this point, however, I had to ensure that the zero flag was true. To do this, I passed arguments to the '0x06' case to flip that switch. Upon the following instruction, the infinite loop occurred. To solve this problem, I cast the arg1 variable back to an unsigned char preventing this exploit.