AppSecAssignment1.1

Part 1

Did the following, create Github account, create remote repository, link Travis CI to my repository , Git Clone remote repository to my Local Repository, create and edit Travis CI file, Setup SSH and GPG for both side of the Local and Remote Repository, download AppSecAssignment1.1 and push it to my repository to see if Travis CI run successfully.

Part 2

Downloading the files from AppSecAssignment1.1 and run the make command to build the program for giftcardexamplewriter and giftcardreader. Build failed because developer forgot to add a string header and declared a prototype function detected by the compiler.

```
giftcardreader.c:248:17: warning: implicit declaration of function 'memcpy' [-Wimplicit-function-declaration]
  248 |                 memcpy(gcp_ptr→message, ptr, 32);
```

```
giftcardreader.c:101:33: warning: implicit declaration of function 'get_gift_card_value' [-Wimplicit-function-declaration]
  101 |   printf("  Total value: %d\n\n",get_gift_card_value(thisone));
```

After resolving these issues, ran the make command and the program build and ran successfully.

```
kali@kali:~/Desktop/BOGBGC$ ./giftcardreader 1 examplefile.gft
   Merchant ID: GiftCardz.com
   Customer ID: DuaneGreenes Store 1451
   Num records: 1
      record_type: amount_change
      amount_added: 2000
      signature: [ insert crypto signature here ]
  Total value: 2000
```

Crash1.gft: While analyzing giftcardreader.c file, I noticed some interesting function and variables that make the program crash.

There is a t in front of the gft content which represents 116 that was set by examplegc.num_bytes.

```
116 74 164 &#116; t
kali@kali:~/Desktop/BOGBGC$ cat examplefile.gft
tiftCardz.com                DuaneGreenes Store 1451        ,@[ insert crypto signature here ]kali@kali:~/Desktop/BOGBG
```

```
// terrible thing to do.)
void setupgc() {
        examplegc.num_bytes = 116;
        examplegc.gift_card_data = (void *) &examplegcd;
        examplegcd.merchant_id = "GiftCardz.com                ";
        examplegcd.customer_id = "DuaneGreenes Store 1451      ";
        examplegcd.number_of_gift_card_records = 1;
```

It seems that 116 or t converted by the text program is needed for giftreader program to allocated memory for fread to use to constrain the boundaries within the files. Since fread contains a pointer for pointing to the beginning of the file to the end.

```
        fread(ptr, ret_val→num_bytes, 1, input_fd);
```

Looking at the types of variables num_bytes being passed especially fread and malloc functions we can find potential crashes especially passing negative values to num_bytes. It seems that when passing a negative within a malloc argument that takes unsigned size_t, it can cause issue when using fread function. Since size_t represents an unsigned integer it can't passes in negative values so any bit wise that represent -1 will create a complimentary of the value displaying a large integer number. This can cause Fread to read outside the file causing an access violation which the OS will reject.

Christopher Desir

```
root@ubuntu:/home/chrisd/Desktop/CGGY9163HW1.1Versions/V1.0# ./giftcardreader 1 crash1.gft
Segmentation fault
```

To resolve this issue, doing a memory check will fix the code by adding an if condition to check if the value is negative, once detected with can return null to end the program before it crashes.

Crash2.gft, was not able to create a crash. Difficult on what type of values I can inside the gft file. I tried giving long inputs to giving it nothing at all but the program ran successfully. To dive deeper I try to investigate the the code using GDB and Rader2 to find something to manipulate the data but failed miserably.

Hang.gft

Analyzing the code and using GDB I notice two while loops that can cause my program to crash. I tried the one of the loops within the struct type function by lowering num_bytes value to 0 to activate another pass through which worked but only did it once.

The other while loop is used is within the animate function and had to call it by examplegcd.number_of_gift_card_records to 3, within the writer program. By changing it to three, within print_gift_card_info, the else if (gcrd_ptr->type_of_record == 3 will be called to execute animate. Using GDB, I notice variable PC was incrementing by += program constantly until pc is greater than program + 256.

Inside case 0x06 there is a ptr call pc which is incrementing memory location. While its increment it will stop until pc is high enough the the program location. This can be exploited if we go back to memory location before its increment. I was able to pass 253 in a char which represent -3 which can cause a decrement in memory location.

By implanting the following parameters, it will increment and decrement every time pc does not meet the condition. To fix this issue, change pc += (char)arg1 to (unsigned char) work because unsigned type does not accept negative numbers when the value is stored.

```
root@ubuntu:/home/chrisd/Desktop/CGGY9163HW1.1Versions/V1.0# ./giftcardreader 1 hang.gft
   Merchant ID: GiftCardz.com
   Customer ID: DuaneGreenes Store 1451
   Num records: 1
      record_type: animated message
      message: ♦
  [running embedded program]
```

Finishing patching the gft files update the make file files to test cases can be read as well for my local machine and for Travis CI when test casus are pushed into the remote repository.

Part 3

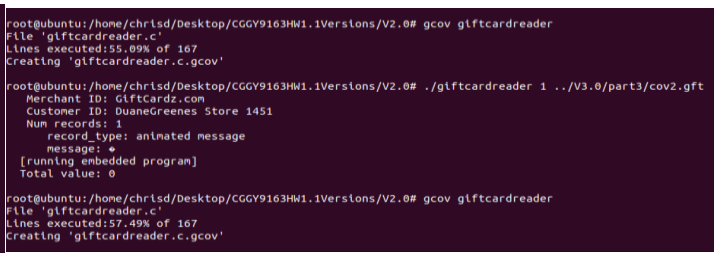
Installed gcov a coverage tool that allows to see what lines were covered and able to observed lines that wasn't covered that may cause vulnerabilities if it does reach those lines depending on the input delta. To setup the coverage validation, we must recompile the giftcardreader.c program with a –coverage flag and started running the input files (crash1.gft, hang.gft).

After setting up gcov I tested the coverage of my first crash1.gft that had a negative on numbytes written on the file, after that created and generated html using lcov.

While running crash1.gft it reported that 8.0% lines was being covered. The program covered less then what it should because it detected a negative num_bytes and ended the program without causing a segmentation fault. When covering hang.gft ,reports showed the program covered 52.69% of the code since it skip the condition that check negative value and running mostly the following function, the struct this_gift_card *gift_card_reader, void print_gift_card_info, and void animate.

To cover more lines of the code, set examplegcd.number_of_gift_card_records value to 2 in the examplewriter program where it will be written in a gft file. By doing changing the value to a one to a two it covers 2.4% more lines of code, the program the coverline within within the print_gift_card_info passed by the else if (gcrd_ptr->type_of_record == 2) {, since we set set examplegcd.number_of_gift_card_records value to 2. Within the if statement it call 2 print function.

For the next item added to cover more line is instantiate the struct gift_card_program as crash and set the variable program to 0x05 in the examplewriter.c file. This call will cover cases within the animate function when the gft file is generated and read by the gft card program. The original hang.gft was doing the same idea but made case to cause it to hang, now we can set program to different cases statement within animate to cover more lines of code. Since we covered a different case 2.7% of the code was covered.



Installing and running fuzzier was pretty was a lot of easy to create test cases to make the program crash. I left the fuzzer for 2 hours and the files it generated. After the fuzzer generates crash and hang testcases. Tested the generate files and notice there was a lot false positive within the hang folder and the crash folder. Only found few crashes but no hangs. The way I work around is used the example writer as an input and made generate hang files which worked successfully.

Using GDB to analyze the causes, was able to find issues pinpoint within the giftcardreader program using the fuzzer 1 and 2 gft file.

Fuzzer1.gft : Like the hang gft file, there was a different issue within the case within animate to cause to cause the program to hang. The cause of the loop occurred in case 0x010 which has a signed char that a negative value may be pass causing pc not being increment till the point the condition is match. Like the previous issue fix the hang by replacing (char)arg1 to (unsigned char) arg1 which the program stopped hanging.

Fuzzer2.gft : Pinpoint the crashes in GDB on the line 107, printf(" Total value: %d\n\n",get_gift_card_value(thisone)); I assume that thisone is causing a access violation to other process since the program finish reading the terminal and then crashes. I tried to not use this one and created a stack variable within the local function to add up ret with gcac_ptr->amount_added but fail. 11/4/2002 Update, after using gdb, noticed that animated is being called and just being returned. One of the variables that can cause an access violation is the pointer pc which is constantly being added within the while stop until its higher memory location than the program +56 statement. To fix this issue before the program break, I set pc to 0 to kind of reset the variable stopping it to access unwanted memory.

Updated my make file and pushed it to the remote repository for travis.ci to test my added coverage and fuzzer test cases. So far still working on the following files, which are crash2.gft.