Secret Message Decryption Bug Report

By: Sriyuth Sagi
Date: 2/22/2018

Overview:

To keep track of the different types of bugs present in a program, which reads an encrypted file and output decrypted text based on specific arguments and operations and analyze the solutions to fix these bugs.

Scope:

Each section of the code shows a tendency towards certain types of errors so a report potentially gives a good indicator of the types of errors that appear while using the various types of data so it is better to explore the program through the operations rather than the individual bugs. Furthermore, all the bugs are reproducible through specific command line arguments so the report will continue to be effective.

Development Environment:

Operating System: MacOS High Sierra Version 10.13.3
Computer and Hardware: MacBook Pro (15-inch, 2016), Processor 2.7 GHz Intel Core i7, Memory 16GB 2133 MHz LPDDR3
Compiler: LLVM version 8.0.0 (clang-800.0.42.1)
IDE: Xcode 9.2,Build version 9C40b
Memory Debugger: DrMemory-MacOS-1.11.0-2

Arithmetic Operations:

Bug ID 00001: Assertion Failed Error Message at Line 132

Summary: Getting an Assertion Failure message while running the arithmetic operation.
Error Type: Runtime Error.
Tools Used: lldb debugger
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --arithmetic-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up and the corresponding screenshot.

```
Assertion failed: (ntns(mnuwcf,znokdz,jy_na_,5,znokdz) == 5), function yengm, file
/Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 132.
```

```
116     // set up some variables
117     int htxjxx = 10;
118     int fedg = 46;
119     int jy_na_ = 4;
120     int wdper = jy_na_ - fedg; // -42
121     int ognpw = fedg - 3*htxjxx + 5*jy_na_; //  32
122     int mnuwcf = 2*fedg + 2*jy_na_; //  100
123     int znokdz = ognpw - (fedg / jy_na_) + wdper + 20; // -1
124     int aqsmt = (mnuwcf / jy_na_) / htxjxx; //  3
125     int lkfe = (wdper / aqsmt) / 7; // -2
126     int gtckbw = znokdz + lkfe; // -3
127     int hwqpp = (mnuwcf / ognpw) - aqsmt; // -1
128     int btwtt = mnuwcf + 2*wdper; // 16
129     int wzcm = znokdz + lkfe + hwqpp + gtckbw; // -8
130     float sthalw = htxjxx / mnuwcf; // 0.1
131
132     //*********************************************************************
133     // The remainder of this function has NO bugs
134     // DON'T EDIT ANYTHING IN THIS FUNCTION AFTER THIS LINE
135
136     // 100 / -1 / 4 / 5 / -1 = 5
137     std::cout << "Multidivide: " << ntns(mnuwcf, znokdz, jy_na_, 5, znokdz)
138     << " (expected 5)." << std::endl;
```

```
htxjxx = (int) 10
fedg = (int) 46
jy_na_ = (int) 4
wdper = (int) -42
ognpw = (int) 36
mnuwcf = (int) 100
znokdz = (int) 3
aqsmt = (int) 2
lkfe = (int) -3
gtckbw = (int) 0
hwqpp = (int) 0
btwtt = (int) 16
wzcm = (int) 0
sthalw = (float) 0
usyd = (float) 0.000000000000000000000000000000000000000459149455
```

Fix Description:

- It is possible to use the lldb debugger to find that the variables, ognpw, aqsmt, hwqpp, wzcm and sthalw, being calculated did not match their desired values.

- The functions were adjusted to find the values of the first four accordingly using only constants.

- Among these, the first four were due to arithmetic errors while the last was a computational error due to not having a float among the variables being divided. Which resulted in the division returning an integer rather than a float.
  - Float was casted onto one of the variables being manipulated to give sthalw and it allowed the value to sthalw to become a float which matched the intended value.

Additional Comments/Similar Errors:

- A similar bug to the one found in sthlaw can be found at line 157 where it causes an assertion failure because of the ntns which is another place where the division results in integers.

File Operations:

Bug ID 00002: Logical Error at Line 43

Summary: Getting a logical error and exiting the file operations function early due to misuse of the equality operator.
Error Type: Runtime Error.
Tools Used:
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --file-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.

```
Usage: /Users/sriyuthsagi/Library/Developer/Xcode/DerivedData/test-
aunpkkrglxassvgzztnqbnxctvgr/Build/Products/Debug/test operations infile outfile
Couldn't start operations.
```

Fix Description:

- The program was returning true at the if statement where it was supposed to return false and therefore exited the file operations function early.
  - The error was fixed by changing the equality operator (==) to an inequality operator (!=) which reversed the logical argument made by the function.

Additional Comments/Similar Errors:

- Another similar error can be found on line 55 inside another if statement though instead of the use addition of an inequality operator, this statement only needs the addition of an ! before the argument inside the if statement.

Array Operations:

Bug ID 00003: Memory Allocation Error Message at Line 195

Summary: There was a message suggesting the presence of a bad pointer or memory location present in the function blq__
Error Type: Runtime Error.
Tools Used:
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --array-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.

```
Thread 1: EXC_BAD_ACCESS (code=1, address=0xfffffffffffff80).
```

Fix Description:

- The message EXC_BAD_ACCESS suggests the presence of a corrupted pointer or bad memory location which is possible due to the presence of an array which is used in the same manner as a normal double within the function.
    - In order to lower the risks of corrupted memory harming the rest of the code, the array hqahy was converted to a non-array variable and because of this it is necessary to add pass by reference values inside the modf function and removed the * signs from the return so there is no longer any risk of memory errors from the array and it is not necessary to delete it.
- There was also a logical error present on lines 398 and 404 where assignment operators were used inside the if statement instead of equality operators.
    - This was fixed by changing the assignment operators to equality operators and removing one of the sets of parenthesis.
- There was another major logic error on line 402 where the function checked if gimijh is the hypotenuse. However, the function failed to check if jmxok is the hypotenuse instead.
    - This error was solved by adding an abs() statement so it will return a value whether gimijh is greater or jmxok is greater and test both.
- There was also the program's only build time error in this function as it had return statements inside the if statements but it had no return if the two if statements failed.
    - This was solved by adding a return -1; at the end of the function.

Additional Comments/Similar Errors:

- The reason this bug is ID 00003 instead of 00001 even though it has a build time error is because the build time error was solved before testing any of the other bugs and chose to order bugs in terms of their runtime in the function.
- The variable hqahy was changed to a float type so it better matches the rest of the function variable types and provides a more accurate value in the return. This meant the modf functions had to be changed to modff functions to take into account the new float type.

Vector Operations:

Bug ID 00004: Memory Allocation Error Message at Line 662

Summary: There was a message suggesting the presence of a bad pointer or memory location present in the function uzhrqy from the for loop.
Error Type: Runtime Error.
Tools Used:
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --vector-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.
Thread 1: EXC_BAD_ACCESS (code=1, address=0x50040c22c)
Fix Description:

- The program was calling on a negative index during the first iteration of the for loop on line 491 and was therefore returning a failure.
    - This error was fixed by changing the initialized value of m__e to 1 from 0 so it will never call outside the possible index.
- The program also had a memory error on line 493 as it tried to call an array index past the maximum indexed value of the array.
    - Calling the .size() function will return the total length of the array but does not take into account the fact that indexing starts from 0 so it is necessary to add a -1 inside the index.

Additional Comments/Similar Errors:

- Another bug that will lead to an assertion failure is also located in the uzhrqy and is due to the vector being entered into the function through a pass by value instead of a pass by reference and can be fixed by adding a pass by reference (&) in both the function prototypes for uzhrqy for the vector dambeq.

Bug ID 00005: Assertion Failed Error Message at Line 688

Summary: There was an error message coming from an assertion failure while running the vector operation.
Error Type: Runtime Error.
Tools Used: lldb debugger
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --vector-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.

```
Assertion failed: (gamer == 4), function aelex, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 688.
```

Fix Description:

- The value of gamer was being displayed as a random value because, while it was defined, the value of gamer was never initialized so calling on it will display a random assigned int type value.
    - This error is relatively simple to fix as it only requires the initialization of the int value to 0 to keep track as a counter.

Additional Comments/Similar Errors:

Bug ID 00006: Assertion Failed Error Message and Memory Leak at Line 715

Summary: There was an error message coming from an assertion failure as a result of zggw, while running the vector operation.
Error Type: Runtime Error and Memory Leak.
Tools Used: lldb debugger and Dr. Memory
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --vector-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.

```
Assertion failed: (zggw(fxaur, qoyafo)), function aelex, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 715.
```

Fix Description:

- The function zggw failed every assert statement showing that the values it was returning were opposite of those that were intended which suggests the presence of a logic error. Dr. Memory also shows a memory leak inside the zggw function.
    - The way to fix this function was to return values that were the opposites of what it had done till now and the most efficient method to do this was to convert the greater than (>) in the if statement inside zggw to a less than (<). However, this will not solve the memory leak there so it is better to swap all the true and false statements and add an { else return false;} to the end of the if statement.

Additional Comments/Similar Errors:

Bug ID 00007: Presence of an Infinite Loop at Line 748

Summary: The for loop at line 748 iterated through values infinitely and resulted in a crash.
Error Type: Runtime Error.
Tools Used:
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --vector-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.
Comparison of unsigned expression >= 0 is always true
Fix Description:

- The for loop would loop infinitely because the stop condition is for the value of enbexn to fall below 0 which is not possible with the unsigned int type uint.
    - The way to fix this function was to change the type of the variable enbexn from uint to int.

Additional Comments/Similar Errors:

- There is another instance of an infinite loop present in the code at line 737, but unlike this error which was a result of the stop condition, the other error was a result of the iterator for the loop being incorrect and could be fixed by changing izwupw+1 to izwupw++.

List Operations:

Bug ID 00008: Assertion Failed Error Message at Line 506

Summary: There was an error message coming from an assertion failure while running the list operation.
Error Type: Runtime Error.
Tools Used: lldb debugger
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --list-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.
Assertion failed: (smlls.back() == 'z'), function ihwoi, file /Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/test/test/main.cpp, line 506.
Fix Description:

- The values within smlls are backwards as compared to their intended order according to the lldb debugger.
    - The way to fix this is to swap the greater than and less than statements on lines 506 and 509 and also swap the positions of A and Z on line 509.

Additional Comments/Similar Errors:

Bug ID 0009: Incorrect Output from List Operations

Summary: The value being returned by the function ihwoi to the main() is incorrect due to a variety of semantic problems which have resulted from different forms of logic errors.
Error Type: Runtime Error.
Tools Used: lldb debugger
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --list-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the error message that shows up.
elderberry quart nectarine orange zwetschge pomegranate durian grape yellow squash fig iodine strawberry tangerine jujube lemon mango cherry uglyfruit apple watermelon kiwi –398054121 letters did not ever appear in the fruit names.
List bugs are NOT FIXED
Fix Description:

- (logic error – line 521) The if statement on the line is checking if either of the two are not divisible by the iterator value whereas it is supposed to be checking if either of the two values are divisible.
    - This can be fixed by changing the inequality operators (!=) to equality operators (==).

- (logic error – line 522) When an element from the list is erased, it is still going to the next element in the list and skipping over one which means that values in the list will end up being missed.
    - This can be fixed by adding -- to the iterator value so it will not go directly to the next value.
- (logic error – line 568) There is another error in this erase statement in which a ++ was added to the iterator in the erase statement which will similarly start skipping values in the list.
    - This can be fixed by removing the ++ from the iterator value so it will not go directly to the next value.
- There is another important error on line 588 where the int chwsn is defined but not initialized which will lead to a random value being assigned.
    - This can be fixed by initializing the value to 0.
- (logic error – line 589) The iterator on this line is initialized to smlls.end() however, this is not possible as assigning to end() is the point the list is over and therefore would not hold a value.
    - This can be fixed by adding -- to the initializing value so it will be assigned.

Additional Comments/Similar Errors:

All Operations:

Bug ID 00010: Memory Leaks at Lines 303

Summary: There is a memory leak present at the line as the program comes back from the file operations
Error Type: Memory Leak.
Tools Used: Dr. Memory
Steps to reproduce: Open the command line window and execute the below commands

- g++ main.cpp –o decrypt.exe –Wall
- ./decrypt.exe --all-operations encrypted_message.txt secret_message_output.txt

Error output: Below is the screenshot from Dr. Memory.

```
~~Dr.M~~
~~Dr.M~~ Error #1: POSSIBLE LEAK 179 direct bytes 0x00517250-0x00517303 + 0 indirect bytes
~~Dr.M~~ # 0 replace_operator_new_array
~~Dr.M~~ # 1 xlo_s                                  [/Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/hw4/hw4/main.cpp:71]
~~Dr.M~~ # 2 main                                   [/Users/sriyuthsagi/Dropbox/Data Structures/Homeworks/hw4/hw4/main.cpp:302]
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       0 unique,     0 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       1 unique,     1 total,    179 byte(s) of possible leak(s)
```

Fix Description:

- When executing the program with Dr. Memory, it is returning a leak at the location that is the same size as the memory that was read from the file which shows that the file, which was read in through file_buffer, was never deleted.
    - The leak can be fixed by including delete [] file_buffer; statements following the use of the variable file_buffer to ensure that none of the memory present is leaked.

Additional Comments/Similar Errors:

Conclusion:

Based on the above bugs, it can be concluded that most of the memory leaks could have been avoided by deallocating all memory locations when they are no longer needed in the program while the present runtime errors were largely a result of errors in logic which could be avoided with memory diagrams and careful planning in advance before the program is actually written out.