Nanako Chung

Application Security

Brendan Dolan-Gavitt

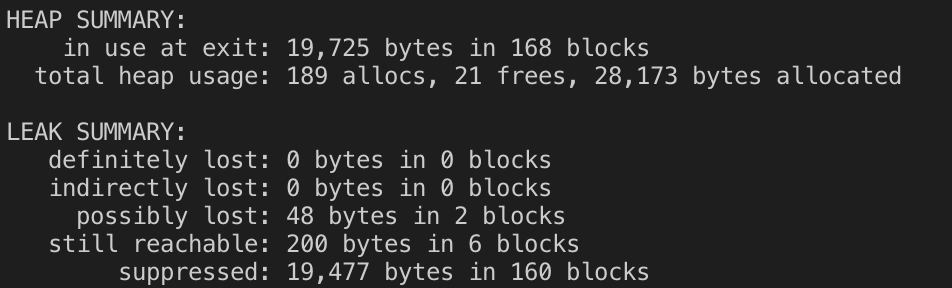
September 22nd, 2019

REPORT 1

• Exactly how the program works:

The spell.c file contains four functions that work to check the spelling of content in a given file. One function takes a "dictionary" file (a file with many words that are spelled correctly) and stores it in a hashtable (i.e. load\_dictionary). Another function takes a file, reads in the content line by line, iterates over each word, calls another function to clean the word in case there are punctuation marks and such attached to it, and assesses whether it is misspelled or not by calling the third function, which takes the word and tries to find it in the dictionary/hashtable.

• The output of Valgrind



• What bugs you expect may exist in your code & why those bugs might occur

* The dictionary may not have as much coverage as we trust it to because it does not list all the possible words (e.g. "shoemaking" is not in the dictionary)
* The dictionary may not have the correct spelling of words because there may be mistakes or alternative spellings (e.g. "colour" vs. "color")
* Words with punctuation marks and special characters may lead to faultiness, especially if there is more than one of these kinds of characters or a mix of both "'" and punctuation marks and special characters
* Numeric strings may have commas and decimals in them, which is valid!
* Some strings with punctuation in the middle are technically "spelled right" such as websites.

• What steps you took to mitigate or fix those bugs

* For the dictionary files, it was tricky and almost pointless to find a solution because there can never be 100% coverage of the words in the English (and non-English) language...
* In my code, I looked at strings with any wacky punctuation or special character, trimmed them using my trim() function to cut off any non-alphanumeric characters at the beginning and end, and then let it go through the detector to see if it was actually misspelled. Thus, all of the unexpected characters in the ASCII table from !-~ and Ç-■ are handled.
* Numeric strings are handled as well. We trim() the string, then count up the digits, commas, and periods, then do a handy math conditional to check if the string is a valid number.

• All of the bugs found by your tests

• All of the bugs found by fuzzing

• How the bugs were fixed

• How similar bugs can be avoided in the future