Assignment 02: Fuzzing

Angelo Porcella, Emery Call, Jasmine Vang, Prakriti Baral

October 7th, 2024

1 Introduction

Our group was able to produce three unique crashes. Each crash is reproducible with the input provided by the fuzzer. With the crash reports, we are able to analyze the inputs for similarities. Additionally, we are able to use the inputs from the crash report to try to identify the crash source.

2 Crash Analysis

2.1 SEGV x2

HW/SEGV.png

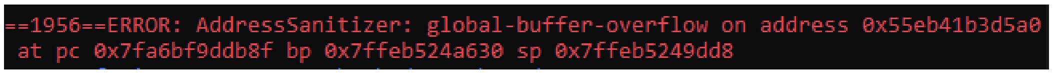
The SEGV crash occurs when there is a read memory access, meaning that memory is accessed that may not exists, the memory is not allocated, or the memory address is invalid.

SEGV is the most common crash that we were able to achieve, and happened quite often. One of the instances of creating the SEGV crash is to modify the email addresses username or domain to be larger than 22 characters long. Modifying either the username or the domain both cause the SEGV crash. An additional way to cause the SEGV crash is to have large amounts of characters in the message body.

We found that there were two variants of this crash, one occuring in the trace at voidsmtpd+0x439b of print_list and the other at voidsmtpd+0x4388 of print_list. The voidsmtpd+0x439b variant was triggered far less in our fuzzer than the one voidsmtpd+0x4388 variant. Seeing as they were both so close in the binary, had the same error message, and seemed to be triggered in a similar way we opted to count them as the same crash type.

With the crash reports for the SEGV crashes, it is shown that certain parameters have a character limit. Once the character limit is exceeded, the SEGV crash takes place.

2.2 Global Buffer Overflow



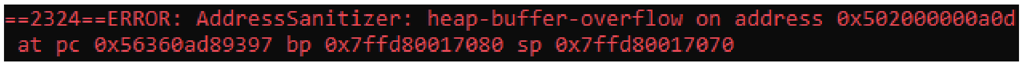
```
==1956==ERROR: AddressSanitizer: global-buffer-overflow on address 0x55eb41b3d5a0
 at pc 0x7fa6bf9ddb8f bp 0x7ffeb524a630 sp 0x7ffeb5249dd8
```

A global-buffer-overflow (GBO) occurs when the program writes or reads more data than allowed in a global or static buffer. In some instances the GBO output that caused a crash, could be modified to create a SEGV crash. It is possible to modify SEGV inputs to create a GBO.

One instance of a GBO occurs when there are several messages with random ordered inputs and valid ordered inputs given to the server. Included in the random input must be periods and the ”HELO” prompt. Removing all periods or the random ”HELO” prompts cause the GBO to cease to exists. Additionally, at least two random string inputs that are accepted by the client need to be at least 290 characters long.

Given the information for a successful GBO, it can be deduced that the GBO occurs when both random inputs exists and valid messages are created. The information outside of the body and random data strings do not need to be inordinately long. A GBO can still occur if the data outside of the body is within expected lengths.

2.3 Heap Buffer Overflow



```
==2324==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x502000000a0d
 at pc 0x56360ad89397 bp 0x7ffd80017080 sp 0x7ffd80017070
```

Heap Buffer Overflow (HBO) occurs when a program writes or reads more data than it has allocated on the heap, causing it to access memory beyond the bounds of the allocated buffer. In some instances, modifying the input that achieved a HBO would change the crash to SEGV. This leads to the potential of taking SEGV crashes and modifying the input to cause an HBO.

The fuzzer was able to cause several HBO crashes, the following analyzes two instances, HBO1 and HBO2.

When analyzing HBO1 it was found that the output contained up to six successive instances of ”RCPT TO”. In the crash report, if the successive instances of ”RCPT TO” was reduced to five instances, the HBO1 crash no longer occurred. One bizarre behavior, was that when adding six successive instances of ”RCPT TO” to HBO2s crash input, it did not cause an HBO crash.

It was found that in HBO2 the message strings contained more characters than HBO1. In the crash report, the typical client output of HBO2 was at least 100 lines more than the client output of HBO1. If lines are removed from the successful crash output of HBO2, the crash no longer remained.

Both crashes contain random input at random instances. It is likely that both crashes are caused by a succession of specific memory being used. For HBO1 there may be a correlation between where ”RCPT TO” is stored, or accessed in relation to other parameters. In HBO2 crashes, it is likely that other parameters cause an HBO2 crash. It appears that there may be a succession of memory allocation that writes or reads in a specific order to cause each individual HBO.