

Final Paper Proposal

Colin Samplawski, Malcolm Reid, and Keith Funkhouser

CS 736 - Fall 2016

1 Introduction

Nearly every function from commonly used C libraries returns some value. In most cases, these values indicate something that is relevant to the functions operations, such as a file descriptor, number of bytes processed, or a pointer to some data. Additionally, most functions can return some predefined value that indicates that an error occurred during the function's execution. Unfortunately, many programmers have the bad habit of not checking these return values for such error conditions. In our work, we plan to analysis a suite of commonly used Linux applications and see if any do not correctly handle errors returned by commonly used system/library calls.

2 Motivation

In their paper (cite), Miller et al. modified the `malloc` library function to return `NULL` with some probability indicting that memory allocation has failed. They then tested a variety of Linux utilities using this version of `malloc` to test if they correctly handled this error case. Surprisingly, the majority of these utilities did not correctly handle this case and crashed in a variety of ways. We plan to expand on this idea and test significantly more system/library calls in a similar way.

3 Method

We do not expect our process to be terribly difficult to implement. Our main challenge is designing a way to wrap the system/library calls of interest. We need to be intercept calls in a way that is invisible and non-distributive to the application being tested. To solve this problem, Miller et al. extracted the binary of the call of interest (in their case `malloc`) and used a binary rewriter to rename `malloc` to `_malloc`. They could then write a new function called `malloc` which would be called by the applications. This new version returned an error value with some probability or just passed the call along to `_malloc`. We have come up with a similar solution that we believe is slightly more elegant.

Linux systems have a built in environment variable called `LD_PRELOAD` which allows users to load a shared library before starting an application. Most importantly, these preloaded libraries take precedence over any other libraries loaded by the application. Therefore if this preloaded library contains a function named

`open`, any calls to `open` by the application will invoke the preloaded `open` and not the version found in the C library(?).

4 A Small Example

5 Schedule

| Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
|--------------------------------------|----------|------------------------------------|--|--|---------------------------------|---------------------------------|
| Oct 24th | Oct 25th | Oct 26th | Oct 27th | Oct 28th | Oct 29th | Oct 30th |
| Oct 31st No class | Nov 1st | Nov 2nd No class | Nov 3rd | Nov 4th No class | Nov 5th | Nov 6th |
| Nov 7th | Nov 8th | Nov 9th | Nov 10th | Nov 11th | Nov 12th | Nov 13th |
| Nov 14th No class | Nov 15th | Nov 16th No class | Nov 17th | Nov 18th | Nov 19th | Nov 20th |
| Nov 21st | Nov 22nd | Nov 23rd | Nov 24th Thanksgiving recess | Nov 25th Thanksgiving recess | Nov 26th Thanksgiving recess | Nov 27th Thanksgiving recess |
| Nov 28th | Nov 29th | Nov 30th | Dec 1st | Dec 2nd | Dec 3rd | Dec 4th |
| Dec 5th No class | Dec 6th | Dec 7th No class | Dec 8th | Dec 9th Paper draft due to referees | Dec 10th | Dec 11th |
| Dec 12th | Dec 13th | Dec 14th | Dec 15th Paper reviews back to author | Dec 16th | Dec 17th | Dec 18th |
| Dec 19th Final project papers due | Dec 20th | Dec 21st Project Poster Session | Dec 22nd | Dec 23rd | Dec 24th | Dec 25th |

| System Calls | libc calls | Pthread calls |
|--|---|---|
| close creat dup fork ioctl mkdir mmap open pipe read write | free kmalloc malloc memcpy printf strto* | pthread_cond_init pthread_create pthread_mutex_init |