Edward Tischler
COP4600
0647
11/17/15
Term Project Part IV

Affirmation that the work submitted is my own and that the Honor Code was neither bent nor broken:

Edward Tischler .

Learning Experience and Summary:

I believe the educational objective of this assignment was to teach us how to create a simple syscall in a foreign operating system. In this assignment, I learned how syscalls work in the OpenBSD operating system and then how to implement one myself. First, I began with creating the cipher algorithm in C. I less believe that this was a major educational objective for us to learn and rather I believe it was meant to make the point that it is best to perfect one's algorithm in speedier environments before trying to implement them in the kernel where compile times can be time consuming. After getting my algorithm working I began to study how to add my syscall to syscalls.master. I thought this part of the assignment was rather easy. I simply copied the structure given to us by Dave in his two syscalls to determine how my syscall should be implemented. Once completed I moved onto the hardest part of the assignment which was implementing my syscall in cop4600.c. I studied Dave's syscalls and found them very helpful in learning how to implement my syscall. I first edited Dave's syscall to determine how everything worked before attempting to implement mine. Through meddling with his syscalls I later learned that *retval was actually the return value from the syscall and not return(0). I also learned that the string in user space must be copied into kernel space before it could be used through using a struct with void *v. Lastly, from meddling with his code I learned that uprintf was used to print from the kernel. With this knowledge I copied and pasted my code with changing all my printf statements to uprintf statements. In addition, I used the template from Dave's second syscall to learn how to store the string from user space as a local variable in kernel space using copyinstr(). Another one of the more difficult parts of this assignment was to learn how to overwrite the original string. By using the man pages associated with copyinstr() I was able to learn that copyoutstr() can be used to overwrite the reference to text so that the user level string will now equal the kernel level string. Other than that, little editing was necessary to my original c code algorithm to have it then work in kernel space. As a result I began to build the kernel. Building the kernel was also in my opinion pretty easy. I was nervous of this because I thought it might take about 5 minutes to compile. But due to my harddrive being SSD the compile times of the kernel were maybe a minute. With a few adjustments and about 5 kernel compiles I was finally finished. I then created a test file that was based off the file given to us. With this I made a prime example showing the syscall working with and without necessary truncation. Part4.txt gave garbage output for some of the text which I found frustrating but the results of the functions weren't garbage so I figured my output was then correct. Overall, I would say this assignment is easier than previous Term Project assignments. I believe this is due to the previous syscall examples given to us by Dave. I also believe I am becoming better at solving these OS related problems.

Edward Tischler
COP4600
0647
11/17/15
Term Project Part IV

Does the program compile without errors?

- Yes, the program compiles without errors.

Does the program compile without warnings?

- Yes, the program compiles without warnings.

Does the program run without crashing?

- Yes, the program runs without crashing.

Describe how you tested the program.

- I tested my program by making sure I can encrypt my strings using the syscall by printing them to the screen. I also printed the return value of the syscall to see that it was correctly returning the length of the new string. I also then reversed my algorithm by using the syscall with –lkey and -nkey to make sure the original string was printed with the correct return value as well. I lastly made sure that a string of more than 1025 characters got truncated with the 1025th character being the null terminator.

Describe the ways in which the program does not meet assignment specifications.

- None

Describe all known and suspected bugs.

- None

Does the program run correctly?

- Yes