Documentation Of Work

Musab 26923, Umar 27132

Part 1:

https://github.com/B-Con/crypto-algorithms/blob/master/

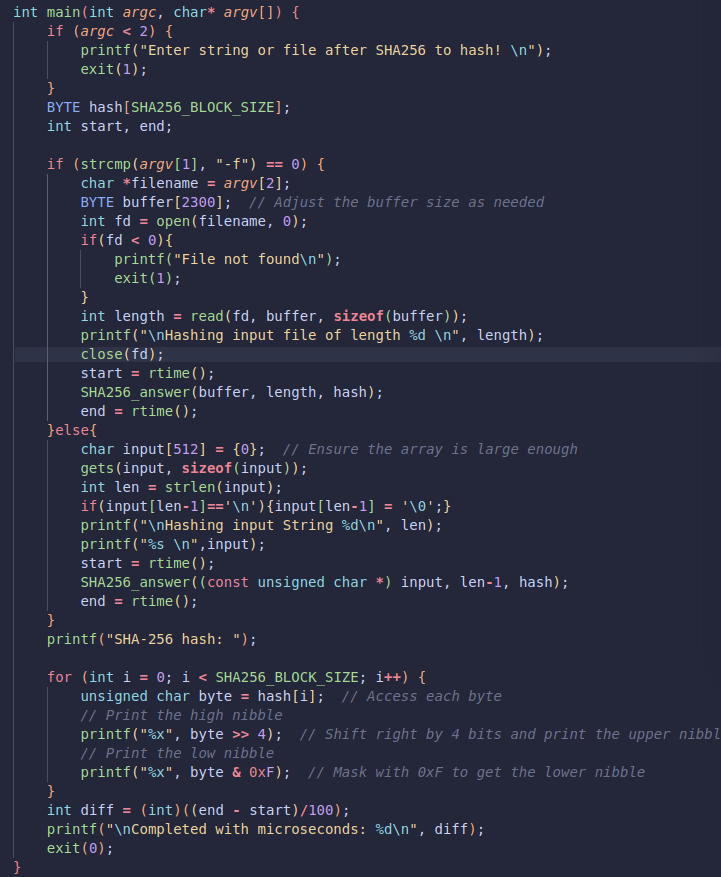
This link contains two of the files we used. sha256.c and its header file.

These files were added to the kernel folder and user folder separately(kernel files were name SHA256.\* and user files were names sha256.\*).

Part 2:

→ User space implementation.

Added a main function to sha256.c where we allowed for comm­and line inputs and file inputs. Sha256.c was then added to UPROGS in the makefile.



Part 3:

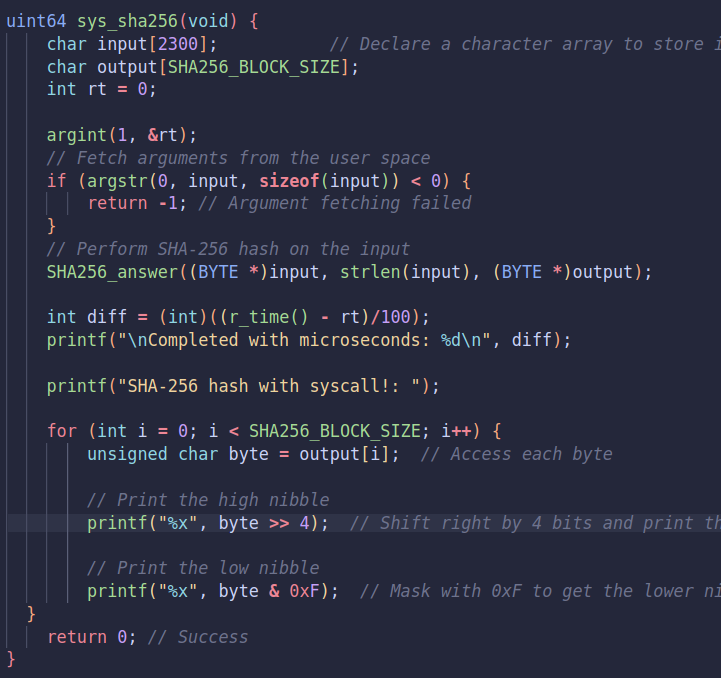
→System call.

Created ‘sys\_sha256’ method in sysproc.c.

Filled out boilerplate required to make a syscall (eg. update syscall.c, syscall.h, usys.pl, user.h, etc.).

Used argstr method to transfer the buffer pointer to the kernel,

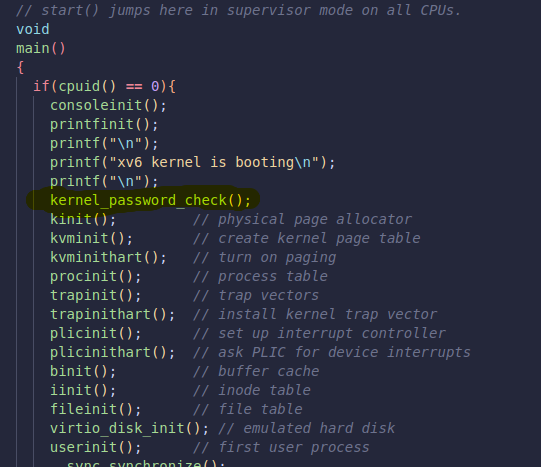
The sys\_sha256 calls the methods defined in SHA256.h(kernel) and print out the result.

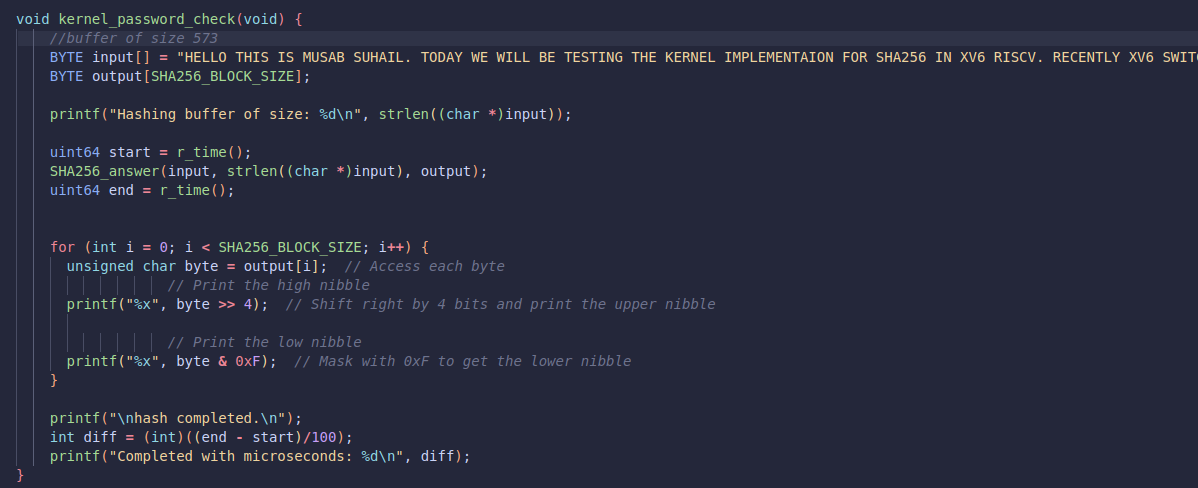


Part 4:

→ Kernel Space.

Set up a hashing of hard coded strings of various lengths through the methods in SHA256.h.





part 5:

→ r\_time syscall.

Use the inline function r\_time() in riscv.h to setup a syscall in sysproc.c, and follow steps taken in part 3 to turn it into a system call. Then divide the difference by 100 to get time diff in microseconds.

Note: Since print statements take particularly more time than any other process. All print statements to be kept outside the hash calculations and time interval.

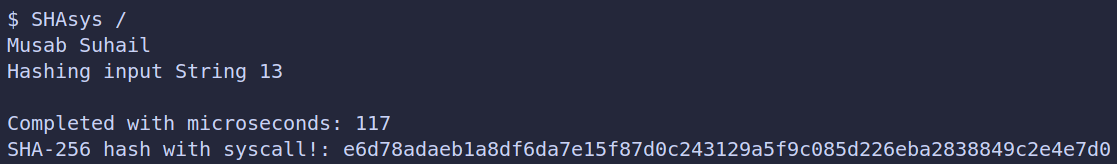
**Manual:**

How to use SHAsys (the syscall implementation):

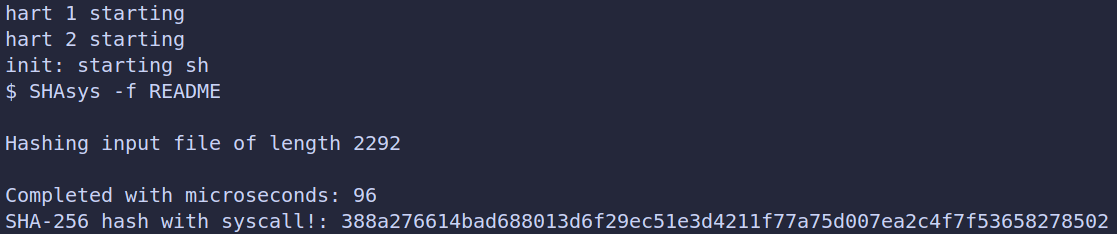
For String:

SHAsys /

<String which fits in the line>



For file:

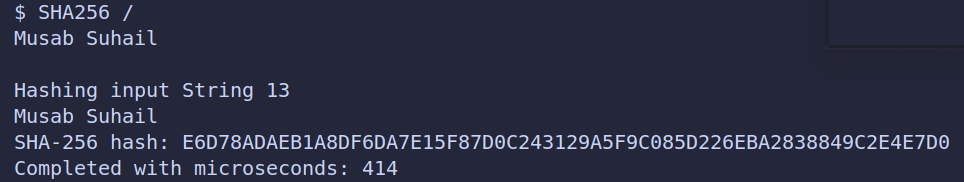
 SHAsys -f <filename>

How to use SHA256 (the user space implementation):

For String:

SHA256 /

<String which fits in the line>



For file:

SHA256 -f <filename>

