This lab was very challenging yet insightful. While implementing the cwlite LSM, I focused on setting up the filtering and creating the authorization in standard with cwlite principles. I feel the filtering mechanic was created properly making the debugfs file structure when sample ran.

The main parts of the authorization was implemented in the has_perm function. The function was the backbone of the cwlite model's integrity protection, and I felt confident in the implementation.

Despite the successes, I encountered significant challenges when trying to test my code. The main issue was getting it to compile, facing many syntax and logic errors like seen below.

```
shahnewaz@shahnewaz-laptop:~/Desktop/p2/p2-user$ gcc -o cwlite_test lemmonscwlite_test.c cwl
cwlite.c  cwlite.h  cwlite.o  cwl_test
shahnewaz@shahnewaz-laptop:~/Desktop/p2/p2-user$ gcc -o cwlite_test lemmonscwlite_test.c cwlite.c
lemmonscwlite_test.c:7: error: expected ‘)’ before ‘char’
lemmonscwlite_test.c:30: error: expected declaration specifiers or ‘...’ before string constant
lemmonscwlite_test.c:30: warning: data definition has no type or storage class
lemmonscwlite_test.c:30: error: conflicting types for ‘printf’
lemmonscwlite_test.c:30: note: a parameter list with an ellipsis can’t match an empty parameter n
ame list declaration
lemmonscwlite_test.c:31: error: expected identifier or ‘(’ before ‘}’ token
shahnewaz@shahnewaz-laptop:~/Desktop/p2/p2-user$ gcc -o cwlite_test lemmonscwlite_test.c cwlite.c
lemmonscwlite_test.c: In function ‘test_file_access’:
lemmonscwlite_test.c:19: error: expected ‘)’ before ‘;’ token
lemmonscwlite_test.c:29: error: expected ‘,’ or ‘;’ before ‘}’ token
lemmonscwlite_test.c:48: error: expected declaration or statement at end of input
```

The compilation issues prevented me from running comprehensive tests to verify my implementation's correctness. Setting up the proper testing environment with the correct kernel version proven to be a lot harder than I initially anticipated when implementing sample.c.

While I was not able to fully test the implementation, I am confident in several aspects of the code. The basic structure of the LSM in sample.c is in place, and im pretty sure the debugfs file for cwlite filtering is set up correctly. The logic for cwlite authorization in has perm, though untested, follows the principles of the cwlite integrity model.

sample.c

```
static int has_perm(u32 ssid_full, u32 osid, u32 ops)
{
        u32 cwl = 0xf0000000 & ssid_full;
        u32 ssid = 0xffffffff & ssid_full;
#if 0
        if (ssid && osid)
                printk(KERN_WARNING "%s: 0x%x:0x%x:0x%x:0x%x\n",
                       __FUNCTION__, ssid, cwl, osid, ops);
#endif
        /* YOUR CODE: CW-Lite Authorization Rules */
        if (ssid && osid)
                if(ssid == SAMPLE_TRUSTED){
                        return 0;
                }
                else if(ssid == SAMPLE_UNTRUSTED){
                        if(osid == SAMPLE_TRUSTED){
                                return -EACCES;
                        }
                        else if (osid == SAMPLE_UNTRUSTED){
                                if(cwl){
                                        return 0;
                                } else {
                                        return -EACCES;
                                }
                        }
                }
        }
        /* Other processes - allow */
        else return 0;
```

Moving forward, resolving the compilation issues in the test code is crucial to verify code implementation. Further testing is surely needed to ensure that the code is adhering to the cwlite model under various (harder to conceptualize) scenarios. This project highlighted the complexity of kernel module development and the importance of a robust testing strategy when working with low-level system components.