To modify file attributes, I decided to use the already existing extended attributes. As a result, I did not have to modify any existing structures, but instead use already existing system calls. First, I had to setup my system calls. To accomplish this, I had to add my system call to the system call table located in **/usr/rep/src/reptilian-kernel/arch/x86/entry/syscalls/syscall_64.tbl**.

332 common set_classification sys_get_classification

333 common get_classification sys_set_classification

Afterwards, function protypes were defined in the ./include/linux/syscalls.h file

asmlinkage long sys_set_classification(int fd, int class_level);

asmlinkage long sys_get_classificationl(int fd);

These changes must specifically be made to link my syscalls to the kernel.

Next I developed both of my system calls. I used the virtual file system verison of **getxattr** and **setxattr** to accomplish the creation and retrieval of the classification values. To do this I had to retrieve the **dentry** and **inode** of a file using the **struct fd.** I obtained the **struct fd** from **fdget** using the file descriptor. After doing that the syscall has everything it needs to call either**__vfs_setxattr** or **__vfs_getxattr**.

The **sys_set_classifcation** call is rather straightforward as after receiving that information, it can just make the call to **__vfs_setxattr** with **"user.classification"** as the extended attribute parameter. The library function **int set_classifcation** has to open the file with **O_WRONLY flag**, the FD is then checked. In the case of ordinary files, open will O_**WRONLY** flag will return -1 if the process does not have write permission. Directories act a little differently and can only be opened with **O_RDONLY.** To get around this, first test if **fd == -1**, and then try to open as a directory. This allows my code to work with ordinary files and directories.

The **sys_get_classifcation** call is similar. After retrieving file information the call to **__vfs_getxattr is made** with **"user.classification"** as the extended attribute parameter. Afterwards the return value is checked for **-ENODATA**, in that case the attribute does not exist, so I return 0. The library function **int get_classifcation** has to open the file with **O_RDONLY flag**, the FD is then checked. In the case of ordinary files and directories, open will O_**WRONLY** flag will return -1 if the process does not have read permission.

Testing was done by modifying the provided securitytagtest.c file to include different types of files with different types of permission access. I used **chown** and **chmod** to change access on files. I also made sure to turn the VM off and then test after restart to ensure the attributes were persistent.