Project 2 Preliminary Design

General Plan of Attack

For this project, I will follow a development style known as test-driven development. As the name suggests, I will initially write unit tests for functionality I intend to have or add into the kernel. Unit tests will correspond with the required system calls.

Locking

I plan to use reader-writer locks for all system calls, where reset, block, and unblock system calls will be writers, and query system calls will be readers.

Storing Data

One of the important characteristics of these systems is that their data will be queried every time a socket or file is created or accessed.

Firewall

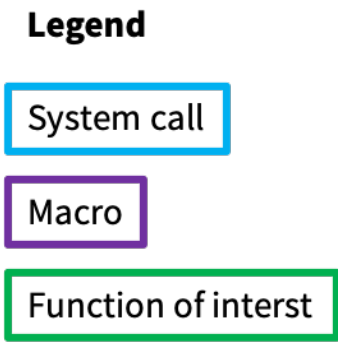
For managing and storing firewall entries, I plan to keep a red-black tree of custom structs where the key of each node is the port number. I will use the Linux provided implementation in **include/linux/rbtree.h**. Red-black trees seem to make the most sense here because port numbers can easily be used here as node keys, so implementing various functions for the tree will not be overly complex. Although a hash table could improve upon the $O(lgn)$ runtime of various operations, depending on the size of the table, the amount of kernel memory in use at any given moment would be fairly large considering the 2^{16} ports available to block.

File Access Control

For managing and storing blocked files, I plan to keep a (chaining) hash table of custom structs where the file path (which starts from **/**) will be used as input to the hash function. I will use the Linux provided implementation in **include/linux/hashtable.h**. I decided on a hash table, after comparing them with B+ trees and radix trees (Patricia tries), because of their inherent ability to categorize various types of data (i.e. string representations of file paths) and near-constant operation times.

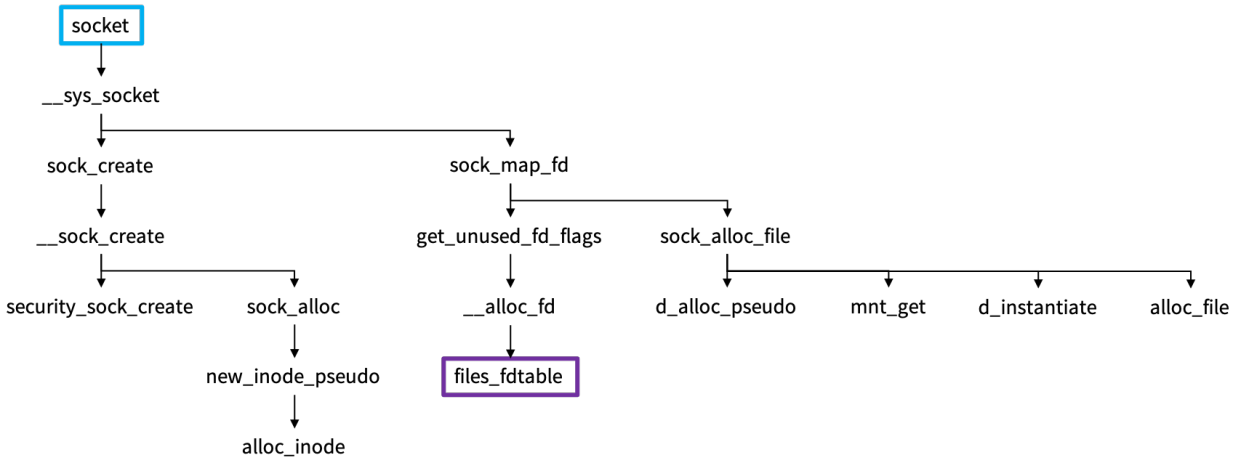
Kernel Modifications

Below, I have traced out various system calls that I believe to be relevant to each of the project's requirements. The legend is as follows:



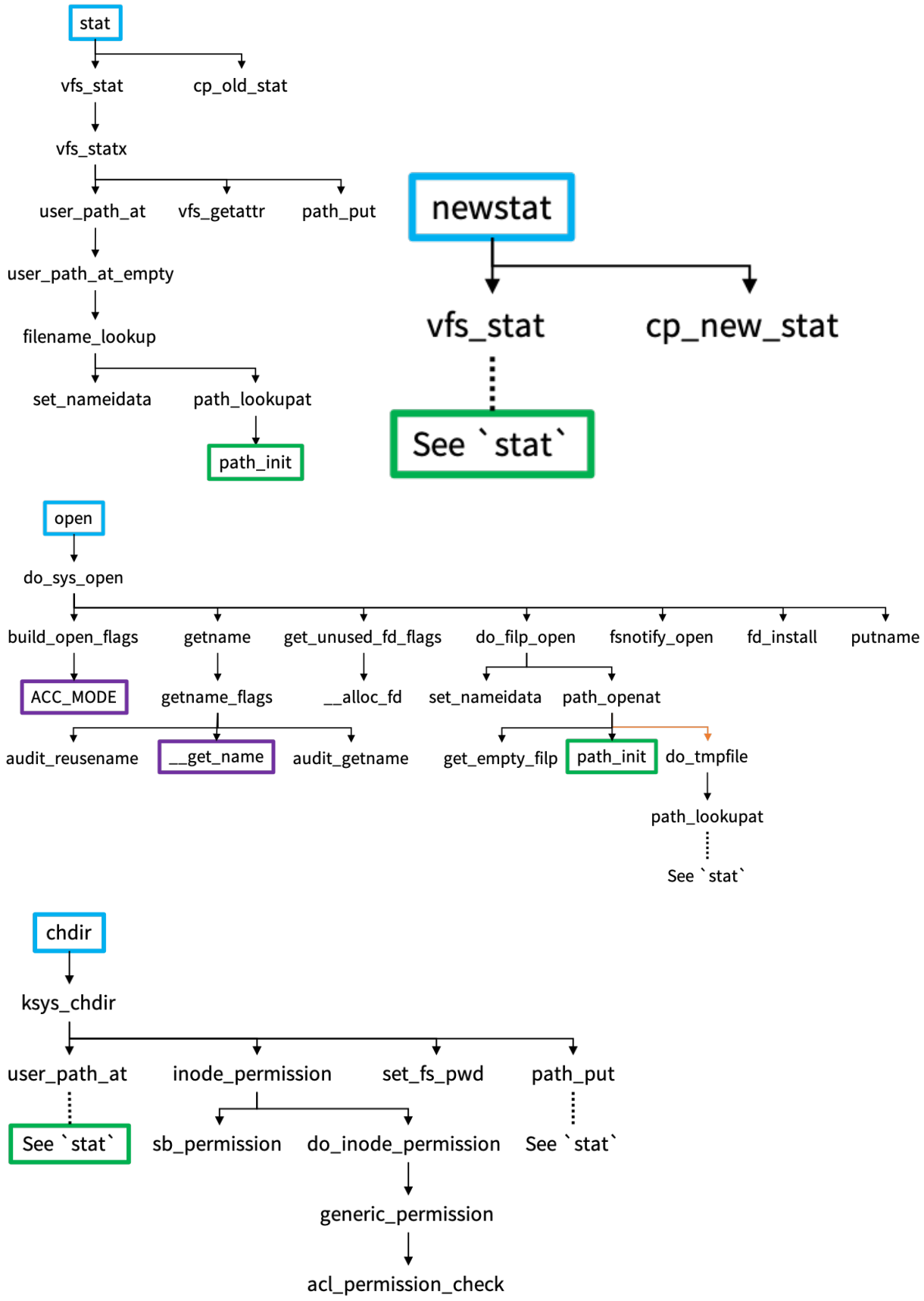
Firewall

The socket system call (in **net/socket.c**) is of most interest for the firewall, since it handles the creation of a socket, the very action that would need to be blocked.



File Access Control

The function of interest for file access control is `path_init` in **fs/namei.c**. This function appears as a place of convergence for many system calls that deal with the filesystem.


stat
vfs_stat
cp_old_stat
vfs_statx
user_path_at
vfs_getattr
path_put
user_path_at_empty
filename_lookup
set_nameidata
path_lookupat
path_init
newstat
vfs_stat
cp_new_stat
See `stat`
open
do_sys_open
build_open_flags
getname
get_unused_fd_flags
do_filp_open
fsnotify_open
fd_install
putname
ACC_MODE
getname_flags
__alloc_fd
set_nameidata
path_openat
audit_reusename
__get_name
audit_getname
get_empty_filp
path_init
do_tmpfile
path_lookupat
See `stat`
chdir
ksys_chdir
user_path_at
inode_permission
set_fs_pwd
path_put
See `stat`
sb_permission
do_inode_permission
See `stat`
generic_permission
acl_permission_check

Conclusion

Please, any advice you are at liberty to share with me will be helpful. It took me multiple hours to simply realize I was looking at the wrong version of kernel source code in the Linux Cross Reference, let alone understanding how they work and spending time researching data structures in the kernel and how to use them. Thank you.

Sources

The following sources were useful in the design process and for understanding various kernel functionality:

- Sockets and Networking
  - https://www.tutorialspoint.com/unix_sockets/index.htm
  - http://man7.org/linux/man-pages/man2/bind.2.html
- Filesystem
  - https://www.howtoforge.com/linux-stat-command/
  - http://www.tutorialspoint.com/unix_system_calls/chdir.htm
  - https://eklitzke.org/path-max-is-tricky
- Kernel Data Structures
  - http://luisbg.blogalia.com/historias/74062
  - https://en.wikipedia.org/wiki/Radix_tree
  - http://www.voidcn.com/article/p-fjdzcnrj-boo.html
  - https://lwn.net/Articles/175432/
  - http://events17.linuxfoundation.org/sites/events/files/slides/LinuxConNA2016%20-%20Radix%20Tree.pdf
  - https://kernelnewbies.org/FAQ/Hashtables
- Miscellaneous
  - https://kernelnewbies.org/FAQ/LikelyUnlikely