Workshop in Information Security – HW 3

User Space Management Interface

An action is done based on the argument passed by the user.
These actions allow us to send and receive data to and from the kernel of the FW.
In most cases (apart from the show_log) the function opens and reads/writes from/to a sysfs device file.
In some of them, some parsing is done – for example: in load_rules – we parse the data we got from the file in the path we got as an argument and in show_rules – we parse the data we got from the sysfs device file.

aux.c

This file contains parsing functions:

- **ip_str_to_hl**: The string ip is parsed to unsigned int – which is send to the kernel, as this is how the kernel stores ip addresses.
- **prefix_size_to_mask**: The prefix size is parsed to mask (i.e 24 is parsed to 255.255.255.0 in unsigned int).

The following also have the "inverse" parsing functions (i.e reason_int_to_str has reason_str_to_int):

- **reason_int_to_str:** The reason int is parsed to the reason string according to the the values in the fw.h file. This is done in order to print the logs.
- **protocol_int_to_str:** The protocol int is parsed to the protocol string according to the values in the fw.h file. This is done in order to print the logs and the rules.
- **direction_int_to_str:** The direction int is parsed to the direction string according to the values in the fw.h file. This is done in order to print the logs and the rules.
- **ack_int_to_str:** The ack int is parsed to the ack string according to the values in the fw.h file. This is done in order to print the rules.
- **action_int_to_str:** The action int is parsed to the action string according to the values in the fw.h file. This is done in order to print the rules and logs.

main.c

- **activate:** 1 is written to the fw_rules/active sysfs device file.
- **deactivate:** 0 is written to the fw_rules/active sysfs device file.
- **clear_log:** 0 is written to the fw_log/log_clear sysfs device file.
- **clear_rules:** 0 is written to the fw_rules/clear_rules sysfs device file.
- **load_rules:** The file is opened and read, line by line. Each line is parsed by the format specified in the assignment's instructions, where reason, protocol, direction, ack, action, and ip are parsed using the parsing functions in aux. Each line is sent to the fw_rules /load_rules sysfs device file.
- **show_rules:** Rules are read from fw_rules/show_rules sysfs device file. They are then parsed according to the expected format and using the parsing functions in aux.
  They are then printed.
- **show_log**: First rules size is read from fw_log/log_size sysfs device file in order to know which buffer size to allocate. Then the whole log is read into the buffer from fw_log/show_log. It is then parsed by the expected format and using the parsing functions in aux, and then printed.

Kernel Space Management Interface

There are two sysfs devices: fw_rules and fw_log. Both are under the same class: fw.

We have read and open functions implementation for show_log.

For all the other user/kernel actions (such as show_rules, rules_size) there are sysfs device files that have display/modify implementation (sometimes one of those and sometimes both).

There is a hook function at the NF_FORWARD stage – where all the magic happens: a packet gets there – All the data is "extracted" and tested. If the FW is active, then we check if we have a rule for such a packet. If we do – we decide according to it.

There is also a logging system with similar log lines being consolidated.

fw.c

- **module Initialization:**
    - The localhost rule is loaded into the rules (using a function that will be explained later).
    - The nf_hook_ops's fields are filled as in HW1 in order for the hook function to get the packets in the FORWARD stage.
    - Both fw_rules and fw_log sysfs devices are registered under the same class.
    - Each device is created.
    - A device file is created for each "action" according to the assignments instructions – both the actions that should be supported by the kernel and actions that should be supported by the user.
- **module cleanup:**
    - The logs string is freed.
    - The logs dynamic array is freed.
    - All the sysfs device files are freed.
    - The devices are destroyed.
    - The class is destroyed.
    - The chardev is unregistered.
    - The hook is unregistered.
- **fops:** fw_log functions as a character device for user's show_log request. The read and open functions are implemented in fw_log.c and are to be explained later in this file.
- **hook function**:
    - All the data is "extracted" from the packet like taught in class.
    - If the packet is of TCP protocol – we check wether it is a XMAS packet – and if so – we add it to logs and drop it.
    - If the FW is inactive we add the packet to logs and accept it. (adding to logs will be explained later)
    - Otherwise – we go over the list of rules and compare the packet to each rule – if we find a match – we add it to the logs and the verdict is the rule's verdict. (comparing a packet to a rule will be explained later)
    - If no matching rule was found – we add it to the logs and we accept it**.**

fw_rules.c

- The rules are kept in a static array of size 50, which is the maximum according to the assignment's instructions.
- There is an actual size indicator: rules_size.
- There is an active indicatoe: active.

- **make_localhost_rule:** Creates the default Localhost rule as instructed in the assignment. When clearing the rules this rule is never deleted.
- **prefix_size_to_mask**: Same as in aux. Used only in make_localhost_rule.
- **ip_str_to_hl:** Same as in aux. Used only in make_localhost_rule.
- **num_rules:** Returns the number of rules.
- **Is_fw_active:** Returns 1 iff the FW is active.
- **check_ip:** Compares an ip to a rule ip by mask – if they are equal we return 0, otherwise we return 1.
- **compare_to_rule:** A packet is compared to a rule by its parameters – if it fits (i.e. colums are the same or rule column is "ANY") we return 0, otherwise we return 1.
- **display_rules_size:** fw_rules/rules_size sysfs device file display function – rules_size is displayed.
- **display_active**: fw_rules/active sysfs device file display function – active is displayed.
- **modify_active:** fw_rules/active sysfs device file modify function – active is altered to 0 or 1 accordingly.
- **modify_clear_rules:** fw_rules/clear_rules device file modify function – if 0 is sent (and this is what the user sends to clear the rules) then rules_size is changed to 1 (as Localhost should always be there by the assignment's instructions). I did not nullify all the rest of the rules since whenever I go over the rules I go over them up to rules_size so at this point all of the rest of the rules are considered "garbage".
- **modify_load_rules:** fw_rules/load_rules device file modify function – a single rule is read from the user and loaded into the rules array at the next position, according to the expected format (parsing is made). rules_size is incremented by one.
- **display_show_rules:** fw_rules/show_rules device file display function – a string containing all rules is created and sent to the user.

<u>fw_log.c</u>

- The log is kept in a dynamic array.
- There is an actual log size indicator: log_size.

- **clear_log:** log_size is set to 0 and logs dynamic array is freed.
- **compare_single_log_row:** Log rows are compared by their fields – returns 0 if similar and 1 if not.
- **add_new_log_row:** If logs are empty – allocates new space, otherwise reallocates bigger space. Then pushes the values to the next log row and increments log_size.
- **add_to_logs:** Checks if the ip is localhost's ip – then we don't log it. Then goes over the logs – if a match is found – the counter is incremented and the timestamp is updated. If no match is found – a new log row is created.
- **create_log_str:** log_str buffer is allocated according to the log_size and the fact that each log takes around 70 bytes of space. Then, logs_str is "built" going over the logs dynamic array and it's fields.
- **my_open_for_show_logs:** The string is created, the length is set and the buffer is initialized at the start of the string.
- **my_read_for_show_logs:** The string is sent to user, where each time, the buffer is incremented by the amount of sent bytes (as the string may be very long and we might not be able to send all of it at once).
- **ip_str_to_hl_2:** same as in aux. Used in order not to log the Localhost communication.