Data and Bandwidth Limiter + Firewall

Suryaansh Jain, Kartheek Tamanna, Rutv Kocheta

Abstract

The firewall is a program that monitors the amount of data used by each client. If the data limit is reached, the firewall will block all traffic from that client. The firewall will also monitor the bandwidth used by each client. If the bandwidth limit is reached, the firewall will throttle that client. The firewall also blocks certain types of traffic (eg: DNS, HTTP, etc.) and certain websites/IP addresses (eg: ad sites).

Features

- run on physical devices instead of VMs (can use WiFi login system)
- Set a cap on the amount of data that can be used (eg: 1GB/day per device).
- Set a maximum bandwidth limit (eg: 30Mbps per user).
- Allow/block traffic from certain ports (eg: 80, 443, 22, etc).
- Allow/block certain types of traffic (eg: DNS, HTTP, etc).
- Block certain IP addresses (eg: 1.1.1.1).
- maintain traffic logs

Future Extensions

- add a login system so that the limit will be per user and not per device
- add a user-friendly interface for admins
- Use logs to generate reports and plots

Design Details

- We have in total 3 threads:
  - The first thread gets packets from the NFQUEUE and decides their fate.
  - The second thread resets the map that stores the speed every time interval.
  - The third resets the map that stores the total data used in the day.

Macros

- NFQUEUE_NUM states the queue number which is used.
- Packet contents are stored in a string buffer of size "BUFFER_SIZE".
- DATA_LIMIT is the limit for the total data a user can use in a day in **bytes**.
- DATA_RESET_INTERVAL is the period of time after which the DATA_LIMIT resets, in **seconds**.
- SPEED_LIMIT is the limit on the bandwidth of a user, in **kilobytes per second**.
- SPEED_RESET_INTERVAL is the period of time after which SPEED_LIMIT resets, in **microseconds**.
- BYTES_PER_INTERVAL is the formula for the number of bytes per interval that the user sends.

Functions

- `void load_blacklist_files()` This function loads the ip addresses from the .txt file into the set blacklisted_ip, and the TCP and UDP ports into the sets blacklisted_tcp and blacklisted_udp. We use a string buffer of length 16 to read the ip addresses in the fgets function.
- `bool check_local(unsigned long ip)` This function checks whether the ip address originates from the local network or not.
- `bool check_valid(unsigned long ip)` This function checks whether the packet originates from internal communications in the network, we allow such packets without any restriction.
- `int callback(struct nfq_q_handle *qh, struct nfgenmsg *nfmsg, struct nfq_data *nfa, void *data)` In this function, we first check if the source and destination address are in the local network. If so, we allow such packets without any restriction. We then determine whether the packet is from a local address or is going towards a local address. The packet is then blocked depending on whether it originates or is headed to a blacklisted ip address. We acquire locks before we update the data and bandwidth maps. Each map are updated by adding the size of the packet. Next, we check whether the protocol is TCP or UDP, and we block the packet depending on whether the respective TCP or UDP port is a blacklisted port.
- `void packet_main()` This function is used for initializing the queue and adding packets to it.
- `void clear_map_tot()` This function clears the total data limit for a user after the DATA_RESET_INTERVAL, after acquiring a lock.
- `void clear_map_speed()` This function clears the bandwidth limit for a user after the SPEED_RESET_INTERVAL, after acquiring a lock.