Hw4secws documentation - ori petel

Important remark:

This submission contains only the first part of assignment 4 –
implementation of the connection table and the statefull inspection.

I got 100 for assignments 1-3 and Reuven approved that successful implementation of the first part will guarantees that **I will pass the course.**

As we agreed, after the semester I will ask the dean for late submission of the second part of assignment 4, and assignment 5.

The submission consists of three parts:

- /module – The kernel space code directory.
- /user – The user space code directory.
- /documentation.pdf – Dry documentation (this file)
- /prev_doc.pdf – Dry documentation of the previous assignment
  (necessary for understanding this one)

The implementation is identical to the one specified in the previous documentation (see attached), except the following additions:

Kernel Side Flow

Inspection

The inspection flow has changed since we now first check if the packet is a TCP packet.

If the packet **is not a TCP packet**, we decide the verdict by matching to static rule table.

If packet **is a TCP packet**, we check whether the SYN flag is on or off.

If the SYN flag is on – we check against the static rule table, and create a new connection that represents the packet.

If the SYN flag is of – we check the packet against the dynamic connection table, and decide the verdict by the TCP state machine (I.e. statefull inspection).

Kernel side modules

tracker

This new module plays the following roles:

- Contains the structure "ctable" the represents all the current connections over the fw.
- Provides connections tracking functionality:
  get connection, add connection, remove connection, free
- Enforces validity of tcp connection (by tcp state machine).
- Provides char device functionality for "Showing connection table".

An interesting fact about my implementation is that I chose the design of **one entry for both** c2s and s2c sides of connection.

Every packet passes through the firewall, given the current state, defines the next state of the connection and **the direction of the next packet that we expect to receive**.

In this way (by remembering the expected direction), we are able to enforce TCP validity more efficiently in both space and time.

User-space

I chose to implement the user space program with c language.

In order to produce the executable file "main", you should run the command "make" in /user directory.

Conn handler

This module handles all the connections issues in the user space program:

- Connection_t structure and required enums (that match to the kernel side).
- Conversion of buffer to a connection structure.
- Functions for conversion between connection fields and strings.
- Function for conversion between the connection itself and a string that representing the connection

User (contains the "main" function)

This module preforms the actual reading and writing flow to and from the firewall devices. (exactly as before)