Chapter 2

Flaw 1: Static Threshold

A static threshold, such as *700.000 bps*, does not take into account natural variations in network traffic. The main limitation of this method is its lack of adaptability. Network conditions are not fixed: legitimate traffic may fluctuate significantly depending on the time of day, the day of the week, or the introduction of new services. A static threshold calibrated for low traffic can lead to a high number of false positives during peaks of legitimate activity. Conversely, a threshold set too high may fail to detect low intensity attacks, producing false negatives and leaving a security gap. To address these limitations, a dynamic threshold mechanism has been designed and implemented. This approach is capable of "learning" normal network behaviour in real time and adapting the definition of anomaly accordingly.

The core of the solution is the adoption of a statistical model that analyses the most recent data flow to establish a baseline for "normal" traffic. This is achieved through two key concepts, which are the sliding window and an application of a technique typically used for anomaly detection [1], called the 3-sigma rule.

The moving window is a technique used to analyze a subset of sequential data. In our context, a history of the last N traffic samples is maintained for each port of each switch, where N corresponds to the `WINDOW_SIZE` and is set to 6. With a sampling interval of *10 seconds*, the window represents the traffic trend of the last *60 seconds* (one minute) but its duration can be freely modified. With each new sample, the most recent value is added to the window and the oldest is removed. This ensures that the statistical model is always based on current

data, allowing the system to adapt to changes in traffic behaviour. As we know, statistical calculations are not reliable without a sufficient number of samples, which is why the system implements a phase in which, for the first 6 monitoring cycles (*60 seconds*), the moving window is still being populated. During this period, the system does not have enough data to calculate a meaningful μ and σ, which are parameters used for the dynamic threshold, so it continues to operate using the basic `static_threshold`. Only when `len(history)==WINDOW_SIZE` does the system switch to calculating, and therefore using, the dynamic threshold. In principle, a variable average could have been used as a first approach, but it was agreed to choose a more robust threshold. In a Gaussian distribution, most values are concentrated around the average. There is a rule, called the *68-95-99.7 rule* [2], which states that approximately *68%* of the data lies within one standard deviation (therefore with *k=1*, one of the common choices in literature. In this way, the dynamic threshold remains close to normal traffic, increasing sensitivity to abnormal peaks without allowing the threshold value to adapt excessively to exceptional values. The aforementioned k is a tolerance coefficient, the choice of which proved to be decisive for the effectiveness of the blocking system. The threshold is defined as:

$$threshold = \mu + k \cdot \sigma \tag{2.1}$$

where μ represent the mean and σ the standard deviation of the traffic measured in the sliding window, respectively. This choice represents a compromise between sensitivity to anomaly detection and the risk of false positives, consistent with the observed characteristics of the monitored network.

The changes focus on controller initialization `__init__` and managing responses to port statistics `_port_stats_reply_handler`. Starting with the first section, the old fixed threshold is retained but renamed for clarity (`static_threshold`). It is no longer the main threshold, but acts as a minimum base value, which is an important detail, given that the dynamic threshold can never fall below this value, preventing the threshold from becoming too low. As mentioned above, a sliding window of size 6 is introduced, thus defining the number of most recent traffic samples that the controller must consider when calculating the dynamic

threshold. In this case, we are actually referring to traffic measurements from the last *60 seconds*, as each sample arrives every `timeInterval=10` seconds. The `self.byte_rate_history` is also defined here, which is a dictionary of dictionaries, where the first-level key is the Datapath Identifier (`dpid`, unique switch identifier), while the second-level key is the port number (`port_no`). The value associated with each port is a list containing the temporal trend of the observed traffic rates (`byte rates`), used as a sliding window.

The theoretical model has been translated into code within the event handler `_port_stats_reply_handler`, which is the function that manages the reception of statistics from Openflow switch ports. It is called whenever a switch sends a reply with its port statistics (`EventOFPPortStatsReply`). Its main purpose is to monitor network traffic, update statistics, calculate dynamic thresholds, and manage the locking and subsequent unlocking of a switch port. After initialising the history structure for each port on each switch, the current value of the observed traffic is entered and, if the sliding window has reached its maximum size, the oldest sample is removed. During comparison, the reference threshold is static by default, while once the window is full (6 samples, as mentioned above), the dynamic threshold is calculated. Another aspect already illustrated is that, in order to guarantee the threshold actually used, the maximum between the dynamic and static thresholds is given. During execution, the system prints information logs reporting the status of the window, which may or may not be full, the calculated mean and standard deviation, and the calculated and adopted threshold.

```
datapath             port     rx-pkts  rx-bytes/s  rx-error  tx-pkts  tx-bytes/s  tx-error
---------------- -------- --------  --------  --------  --------  --------  --------
0000000000000002        1          0       0.00          0          0       0.00          0
SWITCH 2 PORTA 1: Finestra piena. μ=1.16, σ=2.60. Soglia Dinamica Calcolata: 3.77 B/s. Soglia in uso: 700000.00 B/s
0000000000000002        2          0       0.00          0          0       0.00          0
SWITCH 2 PORTA 2: Finestra piena. μ=11.87, σ=17.83. Soglia Dinamica Calcolata: 29.69 B/s. Soglia in uso: 700000.00 B/s
0000000000000002 fffffffe          0       0.00          0          0       0.00          0
SWITCH 2 PORTA 4294967294: Finestra piena. μ=0.00, σ=0.00. Soglia Dinamica Calcolata: 0.00 B/s. Soglia in uso: 700000.00 B/s
```

Figure 2.1: Example of the controller terminal showing the additions made to display the parameters relating to the new feature introduced.

Finally, the incoming and outgoing traffic rates are compared with the current threshold: if either value exceeds it, the alarm counter is incremented, otherwise it is decremented, in order to manage both the activation of the alarm and its subsequent deactivation.

To test the operation of the system, as mentioned, a series of printouts were first added to the log. What was done was to generate fairly aggressive UDP traffic immediately after the initial window had been completely populated, thus setting a fixed threshold to verify both the change in the dynamic threshold and its use in relation to the filling of what can commonly be called a history buffer, and the operations of blocking and subsequent unblocking of the switch port affected by that communication, typically belonging to the first switch on the path.

```
*** Running CLI
*** Starting CLI:
mininet> h3 iperf -s -u -p 5001 &
mininet> h1 iperf -c 10.0.0.3 -u -b 100M -t 60 -p 5001
------------------------------------------------------------
Client connecting to 10.0.0.3, UDP port 5001
Sending 1470 byte datagrams, IPG target: 112.15 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 41696 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-60.0001 sec   750 MBytes   105 Mbits/sec
[  1] Sent 534992 datagrams
[  3] WARNING: did not receive ack of last datagram after 10 tries.
```

Figure 2.2: Generation of UDP traffic using the `iperf` tool, sent from host h1 to host h3. Specifically, the traffic flow has a target bandwidth of *100 Mbps* and is maintained constant for a duration of *60 seconds*.

The system responds appropriately by blocking the switch port affected by the traffic, thus allowing both to note that the threshold has now changed, assuming the value of the calculated dynamic threshold, which is based on recent traffic through that port, and that exceeding it triggers the blocking action as originally configured.

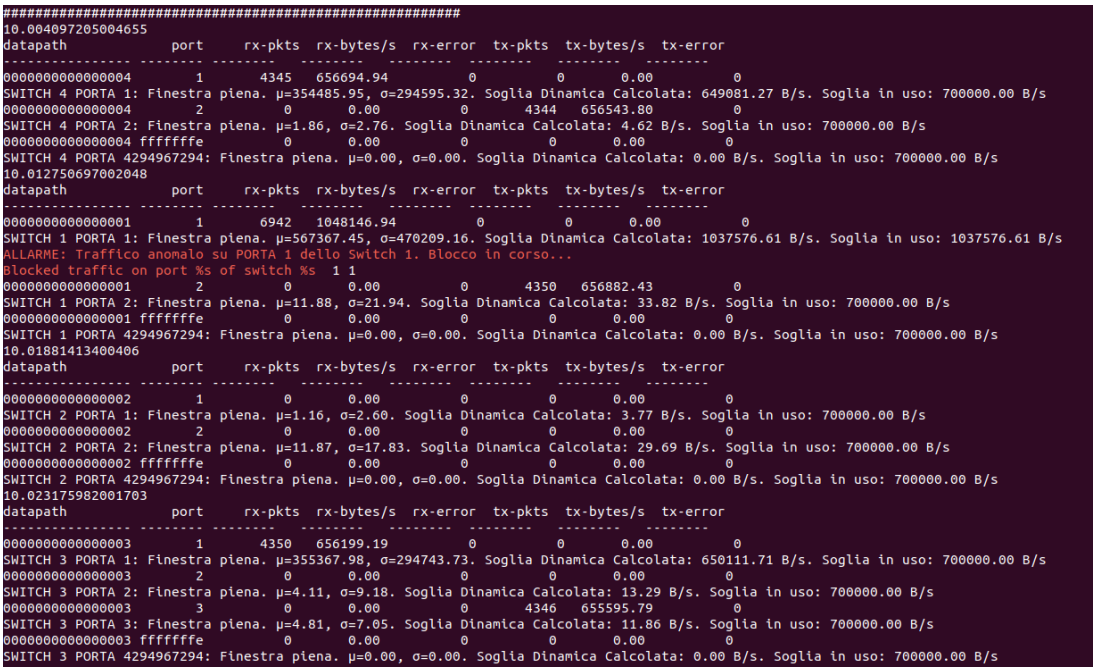

Figure 2.3: Blocking action on port 1 of switch 1, which is the first switch that traffic from h1 to h3 meets along its path.

Once traffic returns to normal, the port is unlocked, effectively returning from the alarm situation.

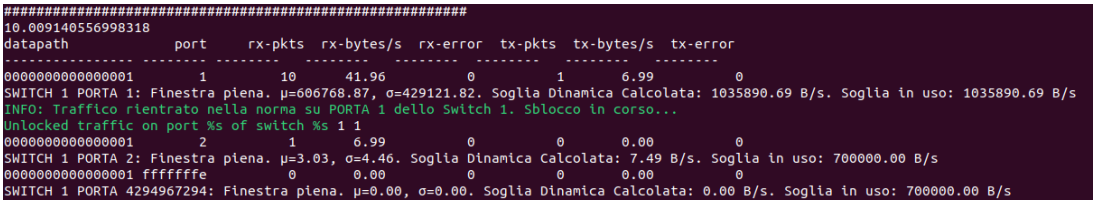

Figure 2.4: Port 1 of switch 1 is now unlocked.

Note that the locking and unlocking processes involve the use of a counter that counts up to 3 before determining a lock, and that unlocking is permitted when the counter decreases below the reception and transmission rate threshold.
One of the main limitations of the dynamic threshold determination approach is that the threshold is calculated on a very short sliding window, consisting of only six samples, corresponding to a *60 second* interval.
To overcome this limitation, a simple improvement could have been implemented by calculating the dynamic threshold based on a wider sliding window. In this way, the threshold would have adapted more slowly to traffic, reducing the risk of overly rapid adaptations, even if it required a longer period to observe the port blocking and unblocking behaviour. A first modification aimed at making the mechanism more robust, introduced after numerous tests, was the implementation of an upper limit on the values that the dynamic threshold can assume. This allows the adaptation to be limited in the presence of traffic that is excessively distant from the reference threshold, defined as the starting point.