Adaptive Load Balancing Architecture for SNORT

M. Shoaib Alam, Qasim Javed, Dr M. Akbar, M. Raza Ur Rehman, M. Bilal Anwer
Military College of Signals, National University of Sciences and Technology, Rawalpindi, 46000. Pakistan.
Email: {mail_shoaib, qasim_nust, makbar_mcs, netwizio, mbanwer82} @yahoo.com

***Abstract* - Nowadays the importance of intrusion detection is amplified due to incredible increase in number of attacks on the networks. The ubiquity of the Internet and the easy perpetration of the attacks will lead to more hostile traffic on the Internet. With the advent of high-speed Internet connections, the organizations today find it difficult to detect intrusions. So multi sensor Intrusion Detection Systems are inevitable. The optimum distribution of traffic to the sensors is a challenging task. In this paper we present a mechanism to split traffic to different intrusion detection sensors to make the task manageable. This splitting of traffic to each sensor is managed by policies enforced on the splitter by the management console. The system is adaptive in the sense that it can adjust the splitting policies for keeping load disparity among sensors reduced. This mechanism of policy- reloading also take into the account the similarity between all possible pairs of policies and tries to minimize the packet duplication rate during the operation of the system. Our mechanism is based on the observation that minimizing the percentage of traffic being duplicated can enhance system performance. We have also discussed the effects of reloading of splitting policies on packet duplication rate and load on sensors.**

***Index Terms* – intrusion detection, traffic splitting, snort, sensor cluster, misuse detection.**

I. INTRODUCTION

Network security is a growing concern as new vulnerabilities are found in the systems resulting in increased number of intrusions and attacks. Different intrusion detection systems have been developed to secure the networks from these intrusive activities. With the advent of high-speed networks the scalability of intrusion detection system has become a vital issue. This problem can be addressed by dividing the load so that the task becomes manageable. In this paper, we present the mechanism of traffic splitting while balancing the load among sensors. Different traffic splitting techniques have been presented earlier. In our proposed solution the traffic splitting is governed by policies defined at the splitter for each sensor. The user defined traffic-forwarding policies may not be mutually exclusive resulting in duplicate packets being forwarded to more than one sensor. Our solution employs the mechanism to reload policies in order to avoid packet duplication. This reloading of policies ensures that similar policies are shifted onto a single sensor thus minimizing packet duplication. Our system also tries to minimize the disparity of load among while splitting the traffic. We keep track of system statistics such as memory occupied, CPU load, CPU usage etc at each sensor. The load balancing process is triggered when the disparity of the above mentioned statistics exceeds the defined threshold.

The major constraint on the design of an efficient load balancing technique is the requirement of forwarding packets belonging to the same flow to a single sensor. This problem of flow preservation is solved by the mechanism of reloading policies. The policy reloading, results in the forwarding of the whole flow to another sensor. Our mechanism also ensures that the disparity of load does not increase due to the decision of reloading policies in order to minimize packet duplication. This is done by keeping in view the policy load factor while reloading policies. The policy load factor is the percentage of the sensor load due to that policy. The experiments done by us shows that traffic load is effectively being distributed among sensors as the traffic load for a particular policy changes.

II. BACKGROUND

A. Network Intrusion Detection

An intrusion is defined as an attempt to break into or misuse your system [1]. With the increase in number of attacks on computer networks, the organizations are paying more attention towards the technologies like intrusion detection systems. Intrusion detection systems can be categorized in to two classes – misuse detection and anomaly detection [1]. The word "misuse" is broad, and can reflect something severe as stealing confidential data to something minor such as misusing your email system for Spam. In misuse detection the network traffic is compared with signatures of known attacks. Anomaly detection, on the other hand tries to identify deviations from the normal traffic patterns [1]. Today's intrusion detection systems employing misuse detection technique cannot keep up with the increasing network traffic. A conventional misuse detection system operates by matching the packets against a rule-set. The rule-set is a two-dimensional data structure, the first dimension comprising of the rule header and the second the packet payload. After a packet matches the rule header its payload is compared with the string patterns stored in the rule-set. This pattern matching is carried out by using string-matching algorithms such as Boyer Moore, which match the packet payload with the stored patterns [2]. The string matching a computationally intensive task forms a big part of the total processing done by the IDS [3].

B. *Problem of Scalability*

The inability of misuse detection systems to handle heavy traffic loads is making them infeasible for deployment in high-speed networks. This problem can be addressed by using a distributed architecture employing multiple sensors. As each sensor handles a portion of the traffic so the task becomes manageable. Very few approaches have come up for splitting the traffic to multiple sensors for intrusion detection. One of the recent works is of I. Charitakis et al [5] in which they have proposed a traffic splitting mechanism based on early filtering and locality buffering mechanisms. In their approach they are reordering the packets to improve the memory access locality on sensors.

C. *Packet Duplication Problem*

The problem of packet duplication arises, as the user-defined policies may not be mutually exclusive. As the rule-set on each sensor is same, there is no need to enforce same policies on more than one sensor. Moreover, the packet duplication due to overlapping policies adds to the computational load of the system. The avoidance of duplication of packets while forwarding the traffic to intrusion detection sensors is not been investigated. This problem is solved by using the mechanism of reloading overlapping policies enforced for different sensors to one sensor. This is done by determining the similarity measure for all pairings of policies enforced for different sensors.

D. *Distribution of single attack's traffic to multiple sensors*

While splitting the traffic to each of the sensors preservation of traffic stream comprising of attack should be done, as it would ensure that packets belonging to a single attack are forwarded to the same sensor. On the other hand, if such packets are split among different sensors the attack detection rate will be lowered. Different Hashing based schemes are implemented for web clusters to solve this problem [4], [6]. This technique is also used by I. Charitakis et al in their work on splitter for intrusion detection system. In our work, we have investigated the mechanism of policy reloading for this purpose. The mechanism of dynamically reloading policies ensure that the whole traffic of the attack is shifted from one sensor to another rather than distributing it to more than one sensor.

III. DESIGN

Some of the major requirements while designing the traffic splitting mechanisms are listed below:

1- Packets belonging to the same attack should be forwarded to the same sensor otherwise the attack cannot be detected.

2- While splitting traffic according to the user-defined policies, the packet duplication should be kept at minimum.

3- The load balancing should be done so that the disparity of load among sensors remains at minimum.

We have designed our solution while keeping in view the above issues. Figure 1 shows the designed architecture.

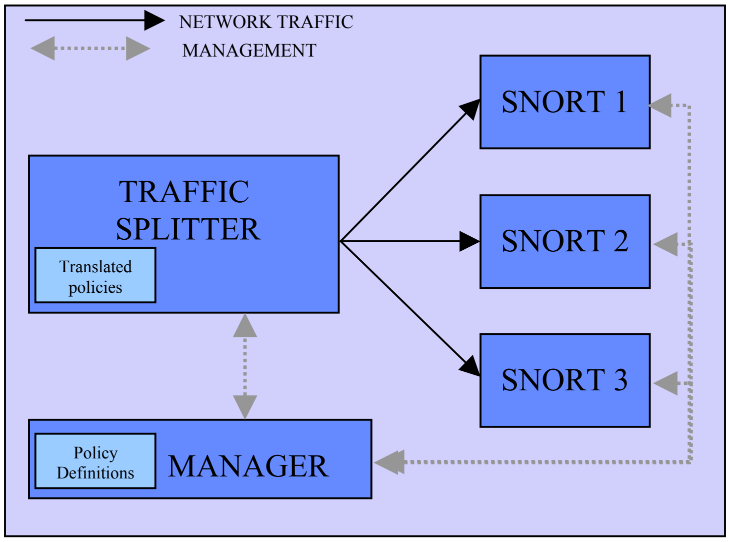

Figure 1 – Load Balancing Architecture

A. *Traffic Splitting*

The above architecture is highly scalable. The splitter distributes the traffic to each of the sensors. More than one policy can be enforced for a single sensor on splitter. The manager keeps track of the system parameters such as memory occupied, load etc on each of the sensors and does policy reloading on the splitter to keep the load disparity among sensors low while keeping the similar policies on single sensor to avoid packet duplication.

In our system the splitter loaded with policies for each sensor does the traffic splitting. The user can define any number of policies according to the requirement. These policy definitions enable the splitter to send the respective traffic flows to the corresponding sensor. There can be multiple policies loaded for a single sensor. All the sensors have a common rule-set. The policies define rules against which the packet header is matched and the packet is then forwarded to the sensor for which that policy is defined. In this way policies are used as a pre-processing mechanism to reduce the amount of traffic to be analysed by the sensor. The policy definition mechanism is very flexible as the user can define any Boolean combination of traffic parameters such as source IP, destination IP, source port, destination port, and protocol type. A policy can comprise of multiple above-mentioned parameters. One example of the policy could be

[{src-ip(202.125.153.45),(212.69.134.45)},
{src-port(11337)},{dest-port(1863)]

The policy definition mechanism is very flexible. The user can define a very specific policy or a very general one. Generally a more general policy will load the system more.

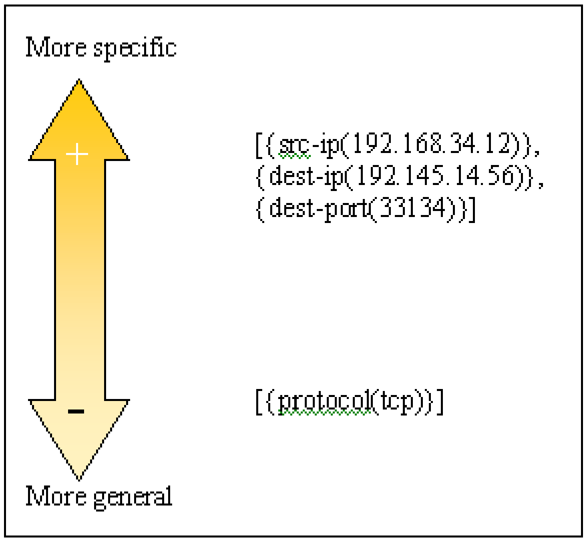

Figure 2 – Difference between specific and general policies

As all the user-defined policies may not be mutually exclusive so two or more policies might result in the forwarding of same packets to more than one sensors thus degrading system performance. This is termed as the problem of packet duplication. In order to deal with this problem, a similarity measure is determined for each possible pairing of policies by monitoring the number of similar packets forwarded by each policy in the pair. It is determined by calculating the number of packets forwarded by each policy. Each policy has its own domain, which comprises of the packets forwarded due to that policy. The similarity measure is indicative of the amount of overlap in two policies. Figure 3 shows two policies A and B having overlapping policy domains.

The policies are reloaded for multiple sensors to a single sensor when the similarity measure for the pair exceeds a certain threshold defined by the user.

B. *Load Balancing*

Balancing the load among different sensors is one of the major requirements of good traffic splitter. Traffic should be distributed among sensors so that their performance is maximized, by keeping the load disparity at minimum. Assuming the cluster of N identical sensors, the ideal situation is to distribute 1/N of the total load to a single sensor. In our approach when a disparity in sensors' load crosses a defined threshold and a need arises for shifting some traffic load from more loaded sensor to less loaded one, a reloading of policies is done. This mechanism of policy reloading to shift traffic load works well to evenly balance the load among sensors. When packets of a single attack are distributed among different sensors, the attack would go undetected.

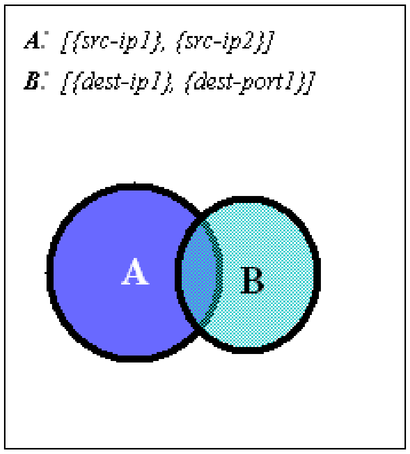

Figure 3 – Overlapping of Policies' Domain

This problem is handled as our approach of reloading policies for shifting the traffic load ensures that packets of a single attack are not distributed among different sensors. Hence the attack detection rate is not lowered due to shifting of flow.

We can divide the problem of load balancing into three steps; load evaluation, policy selection, and policy reloading. Our system keeps track of system-load parameters such as memory occupied, CPU usage etc. for all sensors. These parameters are sent to system manager from each sensor for evaluating the need for balancing the load. We also approximate the load due to a policy on a sensor. This is called the Policy Load Factor (PLF) of the policy. It is approximated by determining the factor of traffic forwarded by that policy to the sensor. So when the disparity is observed among sensors, the suitable policy to be reloaded is selected by comparing PLF of each policy.

C. *Trade-off between Packet Duplication and Load Disparity*

Ideally load disparity and packet duplication both should be kept at minimum. While reloading policies for shifting the load from a one sensor to another it might be a case that the reloaded policies increase packet duplication rate. This implies that we should also consider packet duplication rate while reloading policies on the basis of the load disparity among sensors. In our approach we make sure that while reloading the policies for balancing out the traffic load, the policies comprising a pair having the greater similarity measure are not reloaded to different sensors. The system performance is enhanced as the policy reloading decision is based on both factors.

In order to have consistent system performance, previously calculated measures of similarity measure are considered while taking the decision to reload policies. Moreover, multiple values of load disparity are concerned over a time interval in order to avoid abrupt reloading of policies due to instantaneous heavy system load.

IV. TESTING ENVIRONMENT

The systems used for testing are 900 MHz Pentium III machines with 256 MB RAM. The operating system installed on each machine was Linux Red Hat 9. Using TCPDump, we made a traffic trace of one day. The traffic was known to be free of any attacks. We embedded different attacks in the traffic for the testing purposes. This traffic was replayed using the TCPReplay tool. Each of the sensors had snort 2.0.2 [7] installed for intrusion detection purposes.

V. EXPERIMENTAL DETAILS

First of all, we carried out the tests to compare the performance of system when it is loaded with splitting policies statically and when the mechanism of dynamically reloading of policies is enabled. The Table I depicts the results when the tests were carried without using the mechanism of dynamic reloading of policies. We define and enforce the policies for sensors statically. For judging the performance of the system, we recorded values for system load and memory occupied at each sensor, and percentage of total attacks detected by that sensor (ADP: Attack Detection Percentage). After this we ran the system with dynamic policy reloading mechanism and recorded values for system load, memory occupied and percentage of attack detected at each sensor updated every 90 seconds. Table 2 shows the result of it.

TABLE I – RELOADING MECHANISM NOT ACTIVATED

	Sensor-1	Sensor-2	Sensor-3
System Load	24%	88%	37%
Mem Occupied	58%	85%	65%
ADP	27%	51%	22%

TABLE II – RELOADING MECHANISM ENABLED

	Sensor-1	Sensor-2	Sensor-3
System Load	31%	65%	49%
Mem Occupied	58%	83%	65%
ADP	28%	43%	29%

TABLE III

	Sensor-1	Sensor-2	Sensor-3
System Load	39%	54%	51%
Mem Occupied	61%	81%	70%
ADP	45%	31%	24%

TABLE IV

	Sensor-1	Sensor-2	Sensor-3
System Load	41%	48%	61%
Mem Occupied	67%	62%	74%
ADP	42%	39%	19%

TABLE V

	Sensor 1	Sensor 2	Sensor 3
System Load	53%	49%	55%
Mem Occupied	74%	64%	68%
ADP	43%	38%	19%

All the result parameters shown here have been calculated by averaging the last three values of the particular parameter. All the system decisions are also based on these averaged values in order to avoid sudden changes and to maintain performance while keeping the cost of reloading policies low.

Table I shows the recorded results when the policy reloading mechanism is disabled. The readings show that sensor-2 is heavily loaded as compared to the other two sensors, which means that there is load disparity. However, no reloading of policies would occur, because, the policy reloading mechanism is yet to be activated.

Table II shows the results when the policy reloading mechanism has been activated. The load on sensor-2 is balanced off to two other sensors. This occurred because the disparity of load exceeded the defined threshold and resulted in reloading of policies from sensor-2 to other sensors. Some of the policies on sensor-2 were shifted onto sensor-3 and others were shifted on to sensor-1 as depicted by the results. The result shows that the percentage of total detected attacks detected by sensor-3 has increased significantly. This is due to the fact that reloaded policy from sensor-2 is bringing in more attack traffic to it.

The results from Table III show that the system further levels off the load from sensor-2 to other sensors. We note that the percentage of attacks detected by sensor-3 has come down as the policies loaded for sensor-3 is no more forwarding much attack traffic. Moreover, it is noted that the reloading of policies causes the percentage of attacks detected by sensor-2 to decrease whereas the percentage of attacks detected by sensor-1 has increased. The significant increase in percentage of attacks detected by sensor-1 can be explained by the fact that the reloaded policies from the sensor-2 are now forwarding more attack traffic.

The readings in Table IV are showing that the load on sensor-3 has increased significantly as the policies loaded for it, are now forwarding more traffic. No reloading of policies has occurred. Here load on sensor-2 has decreased as it is now getting less traffic. The percentage of attacks detection by sensor-3 is very low because the policies reloaded for this sensor is forwarding attack free traffic. But on the other hand the percentage of attacks detected by sensor-2 has increased as the traffic forwarded to it now contains more attacks.

In Table V the load on sensor-3 is balanced off to sensor-1 by reloading a policy from sensor-3 to sensor-1. The percentage of attacks detected by sensor-1 remains unchanged because the new reloaded policy from sensor-3 is forwarding attack free traffic. As the sensor has the same policies loaded for it, there is no change in the load on the system or percentage of attacks detected by it.

Now we present another scenario in order to show the effect of the policy load factor (PLF) on policy reloading. There are three sensors A, B and C. Two policies are loaded on sensor-A, three on sensor-B and two on sensor-C.

Considering the cumulative load on each sensor due to the policies loaded on them, policy-7 is reloaded on to sensor-B and policy-3 is shifted to sensor-C to balance off the load. The resulting loads on each sensor are 6.3%, 6.6% and 5.9%. Table 7 shows this situation.

TABLE VI

Policy	Sensor	PLF
1	A	6.3%
2	B	2.1%
3	A	3.2%
4	B	1.0%
5	B	1.6%
6	C	2.7%
7	C	1.9%

TABLE VII

Policy	Sensor	PLF
1	A	6.3%
2	B	2.1%
3	C	3.2%
4	B	1.0%
5	B	1.6%
6	C	2.7%
7	B	1.9%

TABLE VIII

Policy	Sensor	PLF
1	A	2.9%
2	B	2.3%
3	C	4.1%
4	B	1.3%
5	B	3.4%
6	C	2.4%
7	B	1.5%

TABLE IV

Policy	Sensor	PLF
1	A	2.9%
2	A	2.3%
3	C	4.1%
4	B	1.3%
5	B	3.4%
6	C	2.4%
7	B	1.5%

Now the policy reloading mechanism is deactivated for some time. Table XIII shows the readings of the policy load factor when the policy reloading mechanism is activated again after some time. Here policy-2 is reloaded on to sensor-A, which results in loads of 5.2%, 6.2% and 6.5% on the corresponding sensors as shown in Table IX.

VI. CONCLUDING REMARKS

In this paper, we have investigated the possibility of splitting the traffic among the snort sensors using the policy-based splitting mechanism. To dynamically adapt the system with incoming traffic and doing the load balancing among sensors, the mechanism of policy reloading to shift the traffic load from one sensor to the other one was presented. The results show that the disparity of load among sensors is well catered by using our technique. By taking into account the similarity measure among the policies' pairs, while reloading the policies keeps the packet duplication rate low. More testing is being done currently to observe other characteristics of the system as well and bring maturity in results. In the near future, we will test the system to record the effect of number of policies loaded for a sensor and the system performance.

REFERENCES

[1] The NIDS FAQ Available: http://www.robertgraham.com

[2] S. Antonatos, K. G. Anagnostakis, E. P. Markatos, M. Polychronakis. Performance Analysis of Content Matching Intrusion Detection Systems

[3] Nathan Tuck, Timothy Sherwood, Brad Calder and George Varghese- Deterministic Memory-Efficient String Matching Algorithms for Intrusion Detection

[4] Z. Cao, Z. Wang, and E. W. Zegura. Performance of hashing-based schemes for Internet load balancing. In *Proceedings of IEEE Infocom*, pages 323.s341, 2000.

[5] I. Charitakis, K. Anagnostakis, E. Markatos An Active Traffic Splitter Architecture for Intrusion Detection

[6] Network Processor Load Balancing for High-Speed Links Gero Dittmann and Andreas Herkersdorf IBMResearch, Zurich Research Laboratory

[7] M. Roesch. Snort: Lightweight intrusion detection for networks.
In *Proceedings of the 1999 USENIX LISA Systems Administration Conference*, November 1999. (software available from *http://www.snort.org/*).