A scalable distributed IDS Architecture for High speed Networks

Hassen Sallay[1], Khalid A. AlShalfan[1], Ouissem Ben Fred j[1]

[1]College of Computer Science and Information, Imam Muhammad Bin Saud University, Riyadh, - Saudi Arabia

Summary
As networks become faster there is a need for security analysis techniques that can keep up with the increased network throughput. Traditional centralized approaches to traffic analysis cannot scale with the increase of bandwidth advances mainly due to their memory and computational requirements. In the last few years a number of distributed architectures have already been proposed for dedicated network monitoring tasks but they are not scalable in the context of high speed networks. In this paper we present an optimized scalable distributed architecture which is about 10 times quicker than the centralized architecture. The solution is based on switch-based splitting approach that supports intrusion detection on high-speed links by balancing the traffic load among different sensors running Snort.

Key words:
Intrusion Detection, High Speed Networks, Distributed Architecture, Scalability

1. Introduction

Networks began a potentially hostile environment, where intruders are passively or actively trying to breach network security. Passive intruders may browse through sensitive data files, monitor private conversations, or intercept e-mail messages. Active intruders, on the other hand, are malicious and seek to destroy information, deny others access to network resources, and introduce false data or unauthenticated messages onto the network. This type of intruder may even seek to destroy programs and applications by introducing viruses or worms into the network.

Current intrusion detection systems (IDS) are barely capable of real-time traffic analysis on saturated Fast-Ethernet links (100 Mbps). As network technology presses forward, Gigabit-Ethernet (1000 Mbps) has been imposed as a standard for large network installations. In order to protect such installations, a new approach for intrusion detection is necessary to manage the ever-increasing data volume. Network speeds have increased faster than the speed of processors. Thus, centralized solutions have reached their limit, in particular for multi-step attacks or packet's content analysis which requires maintaining much information about the attack. This may seriously overloads a single node. In addition, the centralized solution becomes overloaded as the number of attack's signatures that the IDS must check grows. This is due to the continuous increase of the number of possible attacks. For example, the size of the rule set used by the Snort IDS was quadrupled from 1000 rules in 2001 to over 4000 at the beginning of 2008 [1]. Recent researches have introduced different parallel IDS scheme based on a set of sensors. Each sensor analyses the whole traffic or just a part of it. These parallel IDS has raised new issues including the network architecture, the software architecture, the traffic duplication, and the traffic splitting. In other hand, the correlation of IDS information with vulnerability data is necessary to increase the effectiveness of the security monitoring to satisfy the real-time constraints. This allows for greater automation to take action in real time against intruders. This research paper presents an efficient scalable IDS architecture dedicated for high speed networks. Section 2 shows the related work. Section 3 presents our distributed IDS architecture. In Section 4 the performance study of our system is detailed and finally the paper is concluded by some future work.

2. Related work

The speed of the new backbone network has reached up to 10G so that one single Network Intrusion Detection system (NIDS) can't be capable to monitor the whole network effectively and react in real time to the security attacks. In this section, we present some related work of the research investigations on parallel IDS for high-speed networks based on the distributed architecture [2, 3].

To promote the NIDS performance and efficiency, present studies on IDS for high-speed network monitoring have begun to choose the distributed architecture as an alterative. There are two key technologies in the parallel IDS which are traffic splitting and load balancing. The Traffic splitting designs are mainly based on flows or on security policies and IDS characteristics [4, 5]. It is based on the

Manuscript received August 5, 2009
Manuscript revised August 20, 2009

following some main principles (a) distribution of the packets of the same attack to the same sensor (b) keeping up with the network speed efficiently (c) adaptation to the variety of the network traffic. The load balancing aims to assign an appropriate load among the sensors [6]. It is implemented either by (a) the traffic splitter which make easy the management of the overall system [7]. This choice requires an expensive specific high-speed instrument to work at a full wire speed, and more may be likely the bottleneck of the system. Or by (b) each sensor based on the load balancing approach by applying of some proposed effective algorithms for load evaluation and adjustment which will predict overloading on nodes precisely and reduce the load smoothly to get a smallest packet loss rate [8]. Due to the complexity of these algorithms, they are enhanced by several improvements to increase the system performance such that (a) early filtering, where a portion of the packets are processed on the splitter instead of the sensors (b) multiple levels of hashing to disperse the highly intensive traffic on one sensor (c) or an analyzing node, which receives messages issued by sensors, detects the multi-object attack behaviors and adjusts the distribution of the network flow dynamically.

Mainly, four parallel techniques exist in the literature for IDS: packet-level parallelism, session-level parallelism, rule-level parallelism, and component-level parallelism (see figures 1.a, 1.b, 1.c, 1.d).

The packet-level parallelism consists on creating several dependent sensors [9]. Each sensor applies the complete set of rules. A round-robin-like algorithm is then used to split the traffic among the sensors. A perfect load-balancing across all sensors is then guaranteed. However, a session analyzer is necessary (see figure 1.b) in order to perform stateful analysis. This is required for example for multi-packet or multi-session attacks. The session analyzer has to maintain session integrity, flow and stream tables, and a reassembly list. The analyzer fills these structures from the preprocessors. The analyzer can be considered as a shared space updatable by the preprocessors and consulted by the sensors. This shared space is expected to be a serious bottleneck since it should be protected against access each time a component tries to use it. Moreover, the space is used at least as many times as the number of incoming packets and sessions.

The session-level parallelism is motivated by the fact that the packet from one session will never affect another session state [10, 11]. Therefore, different sessions can be processed by independent sensors. A session splitter (see Figure 1.c) are then used to load balancing the sensors by sending session's packets to different sensors using a round-robin-like algorithm. The shared space required by the packet-level parallelism can be avoided by simply maintaining separate session tables for each sensor. However, the reassembly stage could not be eliminated. In addition, the network analyzer is needed in order to detect multi-session or multi-host attacks. The session-level parallelism does not guarantee a perfect load balancing among sensors. In fact, since over 80 % of the snort rules are TCP-based and over 30 % are HTTP-based, some sensors will be overloaded against some discharged sensors.

The rule-level parallelism creates a set of rules for each sensor [12]. In this case a traffic duplicator is then used to send every packet to every sensor. Each sensor performs quick check of every packet to determine if the sensor contains a rule associated with the given packet. If so, the sensor would process the packet, otherwise it would drop it. This approach guarantees a load balancing among the sensors if the administrator divides the rules perfectly. However, each sensor, in this case, wastes much time while preprocessing every packet and dropping unnecessary ones.

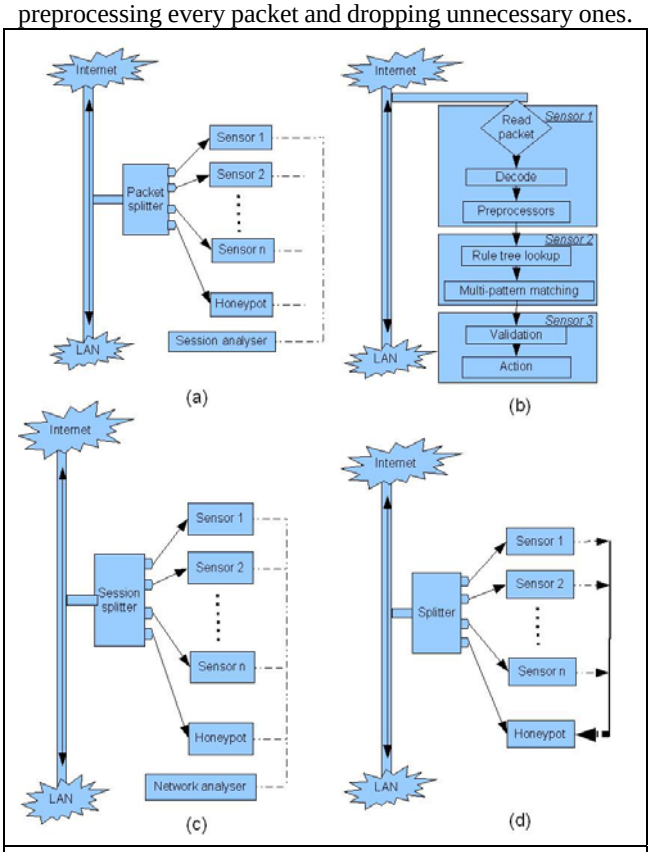

Fig. 1: parallel architectures

In component-level parallelism, individual components of the IDS architecture are isolated and given their own processing elements (see Figure 1.d). Two natural candidates are preprocessing and multi-pattern matching. There are a number of ways in which component-level parallelism could be implemented. Consider an example with three sensors. The first sensor read the packets, classifies them and reassembles them, the second sensor

processes the packets and the third sensor validate the attacks and performs the passive/active actions. This performance of the IDS based on this architecture is guaranteed from the high-level pipelining where a sensor feeds the next one.

3. A Distributed IDS Architecture

3.1 Design Challenges

To design and implement new models and architectures we should address the following main challenges imposed by the high speed networks and the real time constraints:

Scalability: IDS should handle heavy traffic load. As a network may grow fast, the IDS design should allow the addition of new sensors. In addition, the design of the IDS should be flexible and extendable in order to support the exponential evolution of the intrusion detection rules' number.

Fault tolerance: since the IDS may fail down for different reasons (hardware failure, Denial of Services attacks, etc), the global IDS architecture should monitor the different sensors and forwards the whole or a part of the traffic from a down sensor to another sensor. The forwarding may also occur if a sensor is overloaded and thus it should be discharged.

Stateful analysis: when using traffic splitting, the traffic stream comprising a single attack should be sent to only one sensor and not to be scattered to different sensors.

In-depth analysis: the parallel IDS design should encompass a honeypot which is called by the IDS in order to provide an in-depth analysis about the attackers' objectives and techniques.

Accuracy and efficiency of alert correlation: how to couple meaningfully different alerts generated by different heterogeneous distributed IDS some of them based on misuse model and some others based on anomaly model and how to correlate these IDS alerts with the network vulnerability assessment alerts efficiently and in real-time

3.2 Architecture Design

We propose the following architecture (see Fig 2).

For incoming traffic:

1. The high speed network switch/router splits the incoming traffic to the IDS sensors by forwarding the packets based on its switching table to the dedicated sensor for this service. For example the FTP sensor will only select packets sent to the FTP server(s) deployed in the network.
2. Each dedicated sensor runs only the IDS rules set dedicated to the depicted packets. For example, the FTP sensor will run the IDS rule set dedicated to the FTP service on the FTP depicted packets.
3. In case of attack detection, the sensor executes the appropriate security policy strategy and it sends an alarm to the control center (see Fig2. TSM component), Otherwise the sensor will resend the packet to the underline local area network.

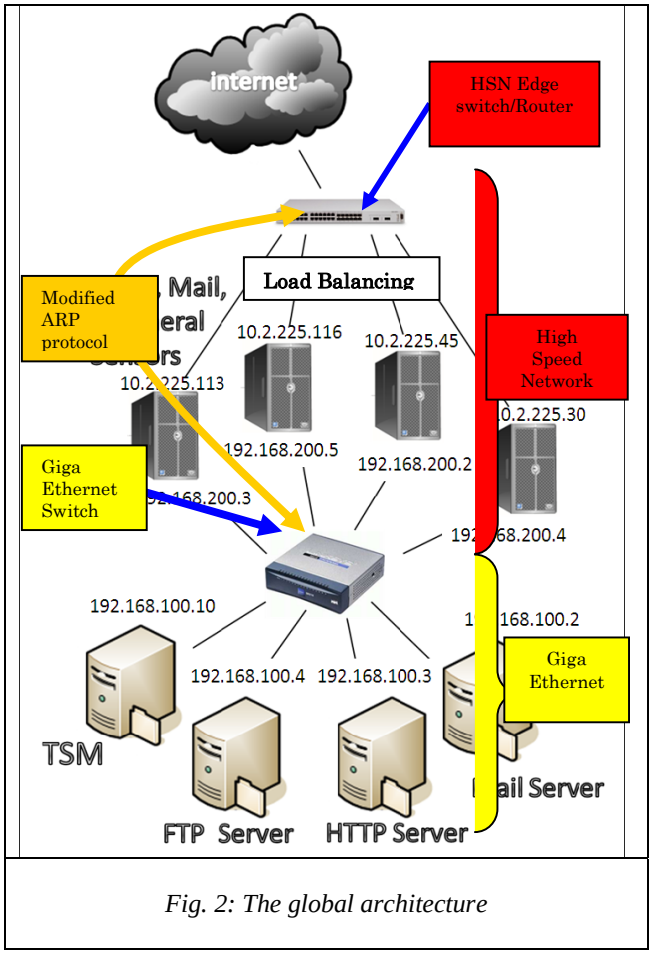

Fig. 2: The global architecture

For outgoing traffic

Each server/host in the local area network has a Host Intrusion Detection System (HIDS). If that HIDS detects an attack, it will send an alarm to the control center.

For the internal or external attacks

The control center (TSM) aggregates and analyzes the different alarm to detect an eventual internal or external multi host attacks. It will execute some event correlation algorithms for that purposes. Depending on the attack nature, the control center executes an automatic security policy strategy or sends a notification to the network security administrator to do the right actions. In all the case, the control center produces some different security reports about the network security status which can be used for the security forensics tasks.

3.3 Critics and benefits of the architecture:

- This solution does not require a special hardware. Therefore, it is easy to deploy and integrate in the existent real running infrastructure
- A little investment in network configuration is needed since the splitting was delegated to the switch itself which should be monitored either dynamically via a variant version of the ARP protocol to change dynamically the switching table or via the switch port mirroring techniques.
- As opposite of the existent parallel IDS This simple and efficient splitting design meet the demand of high speed, without using some techniques such as encapsulation and des-encapsulation of the packets, at the contraire, the IDS will not touch the traffic packets except the content itself for security analysis.
- It assigns the traffic across nodes as evenly as possible and it adapts itself to the variety of the network traffic.
- This systematic splitting of the traffic reduces the set of security rule to be only those needed by the service-specific IDS sensor.
- The multi host attacks will be detected by the control center by integrating the node-issued alert messages (based on event correlation algorithms) to detect multi-object attacks on the whole network.
- The real time defense could be done either by automatic security policies or by the administrator security actions decision.

4. Performance study

4.1 Four scenarios

In order to validate the performance of our architecture (which will be called the optimized architecture), three other architecture are built. First, the centralized architecture which includes only one sensor configured with all the Snort's rules and fed with the whole traffic. This is the typical architecture which is the reference. Second the rule-based architecture which encompasses seven sensors. Each sensor uses only a part of the rule-set. The division of the rule-set is based on the destination port of the traffic. Third, the data-based architecture which encompasses seven sensors each one uses the whole rule-set but fed with a part of the whole traffic. The data-splitting is done according to destination port via port mirroring techniques. Finally, the optimized architecture encompasses seven sensors each one uses only a part of the traffic and only a part of the rule-set.

4.2 Testbed

The testbed is composed by seven sensor. Each sensor has a Intel Core 2 Duo CPU E6750 processor running at 2.66 GHz with a 4096 KB cache size. The operating system is Ubuntu 9.04 over Linux kernel 2.6.28-13. The installed IDS is Snort 2.7.0 using the Sourcefire VRT Certified Rules (8134 rules). Snort is configured is a way that generates a binary log file without checking the checksum of the packets. All the alerts are saved in a mySQL-5.0.75 database using the Snort-mySQL schema version 107.

In order to guaranty an accurate comparison between the four scenarios, we have opted to use the same dumped traffic for all the tests which is the RootFu!-11 traffic. RootFu! is the descendant of the Capture the Flag competition held at Defcon conference each year [13]. It is organized and run by the Ghetto Hackers. RootFu! elects eight teams at the DefCon conference against each other in a test of network defense and hacking skills. Each team has to defend its own server and applications while trying to break into the servers of the seven other teams.

Each team had to run five Web services on a variant of Unix known as BSD. The services consisted of the music streaming application IceCast, a Web news portal based on Slashcode, two ads, and a multiuser text-based role-playing game known as FurryMuck. Each team accumulated points for having the applications available. The longer a service was up, the more points its supervising team won. However, each team lost points if a service it was running became compromised. Teams can choose to port, upgrade or replace services.

The traffic generated during RootFu! days was dumped using the tcpdump program. RootFu!-11 is a part of the Defcon-11 conference. The dumped file encompasses 10527588 packets in a 3.7 GB file. It contains about 31500 Snort's alert.

4.3 Benchmark

RootFu! Dump files encompasses 10527588 packets. For the optimized and the data-based scenarios, the packets are divided into seven files, one file for each sensor, each one encompasses about 1500000 packets. The packets are divided in a way that all the packets of a given destination port are grouped together in the same sensor. So that, the task of the packet splitter become easy since it checks only the destination port of each packet an forward it to a unique sensor. As a result, each sensor is specialized to a specific range of ports as shown in table 1.

The configuration depends on the destination ports of the dump file which guarantees a load balanced architecture. However, the real traffic in a real network is different from the RootFu! traffic. But, when applying the same port-splitting method dynamically on the fly or statically with a real dumped file, another port distribution will be resulted which is suitable for the new network architecture.

Sensor number	Range of ports
1	[0,80] and no a tcp or a udp packets
2	[81, 1539]
3	[1540, 3713]
4	[3714, 8312]
5	[8313, 36412]
6	[36413, 60158]
7	[60159, 65536]

Table 1: service ports distribution among sensors

Given this port-splitting configuration, the Snort' rules are also divided into seven files. Each file contains the rules that must be applied to the given range of ports. e.g. the rule file number one contains the rules that correspond to the range of port [0,80] as a destination port or that is neither a TCP packet nor a UDP one. The resulted rule distribution is shown in Fig 3.

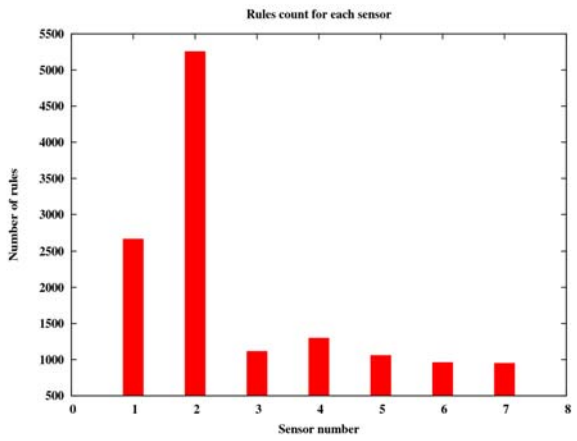

Fig.3 : Rules count for each server

Finally, for each scenario, Snort is fed with the specified dump file and configured to process only a given number of packets. This number varies from 100000 to the whole number of packet in the file. Each run is repeated 15 times. For each run, the real processed number of packet, the processing time and the number of generated alerts are taken. Then, the average of the processing time is saved. Finally, the Snort's log file and the Snort's database are emptied after each run. Finally, for each scenario, the minimum and the maximum processing time among the sensors is kept.

4.4 performance analysis

The figure 4 is a comparison between the centralized and the rule-based architectures. The X-axis is the number of packet processed by Snort and the Y-axis is the corresponding processing time. The doted red curve corresponds to the centralized architecture. It shows a linear curve which is a normal behavior. The others curves correspond to maximum and the minimum of the rule-based scenario.

Note that a rule-based sensor contain sonly a part of the rule set and is fed with the whole dump file. In order to select the packets that correspond to the sensor' ports, a Snort port filter is added. Thus, in order to process 100000 packet for example, Snort must read and filter much more than 100000 since the packets that correspond to the sensor's filter are spread on along the file, in contrast to the centralized sensor which reads the first 100000 packet of the traffic. Hence, the maximum curve and even the minimum curve are always above the centralized curve. The maximum curve points correspond to the sensors six and seven' points because their ports are useless.

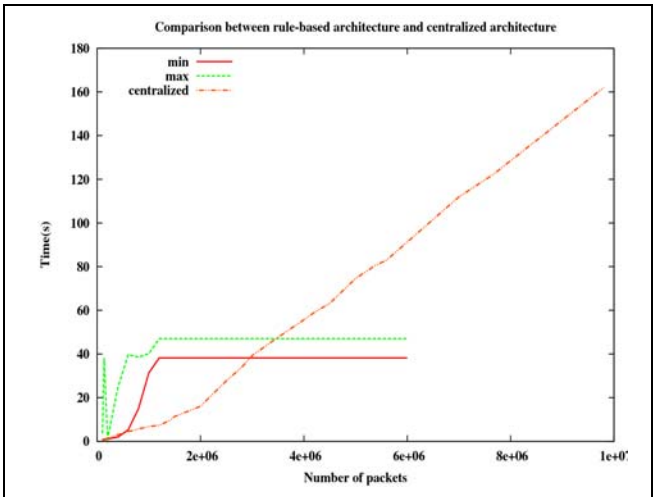

Fig. 4: comparison between the rule-based architecture and the centralized architecture

For Example, the sensor seven must read the whole traffic

in order to process only 134495 packets which requires about 38 seconds. Then, the processing time of the sensor seven remains about 38 seconds while the number of packets that Snort is asked to process is more than 134495 because there are only 134495 packets that verify to the sensor's filter. The maximum number of packet that satisfies a filter is 1092915 which corresponds to sensor one's filter. After that point, the Snort's processing time remains constant.

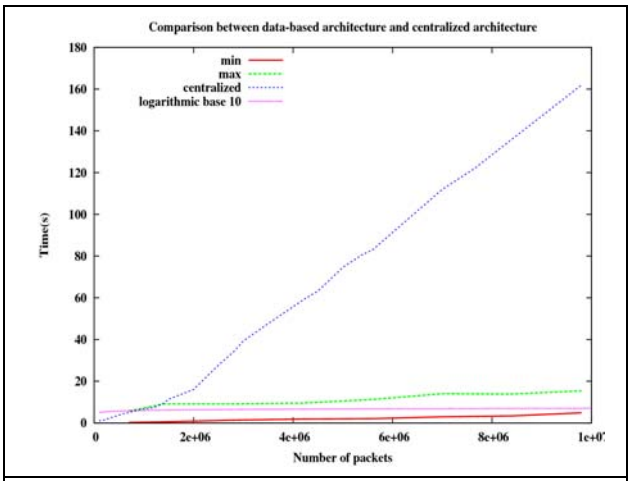

Fig. 5: comparison between the data-based architecture and the centralized architecture

Figure 5 shows a comparison between data-based and centralized architecture using the same axis as the previous figure. There are four curves, the centralized curve, the minimum curve and the maximum curve among the seven sensors of the data-based architecture and the logarithmic base10 function. In this architecture, the traffic is split according to the port ranges of each sensor. The curve of the maximum shows an excellent result especially after 1600000 packets. Between 700000 and 1600000, the centralized demonstrates a better performance, this is due to the types of packet processed by the sensors. In fact the centralized sensor generates about 461 alerts against only 128 alerts by the sensor number one with 700000 packets. 143 alerts among 461 are caused by the destination port 80 which requires 2400 pattern matching checks and a quarter of the 266 alerts are caused by packets with destination port 3306 which are checked using pattern matching against 1681 rules. The 128 alerts generated by the data-based sensor number one are ICMP packets which checked against 710 rules. The Snort's processing of ICMP packets is very expensive.
After 1600000 packets, the data-based architecture is always better than the centralized one. As compared with the logarithmic curve, the data-based architecture demonstrates a logarithmic behavior which is a very good result.

The figure 6 is a comparison between the optimized architecture and the centralized architecture using the same axis as the previous figure. Each sensor of the optimized architecture is fed with a part of the whole traffic and contains only the associated rules. The optimized architecture is the best architecture that guarantees a logarithmic curve. The maximum processing time is always less than the one spent by the centralized architecture especially after 1600000 packets. Before 1600000 packets, the ICMP packets produce almost all the alerts and dominate the packet processing phase.

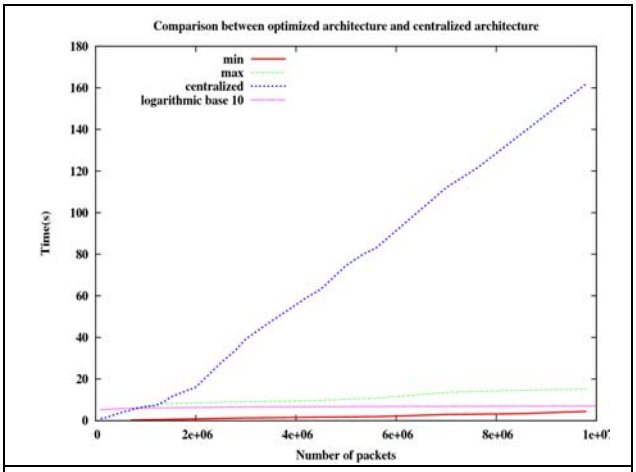

Fig. 6: comparison between the optimized-based architecture and the centralized architecture

The optimized architecture is a large scale architecture. With 9800000 packets is requires only 15.12 seconds against 162 seconds with the centralized architecture. In other words, the optimized solution is about 10 times quicker than the centralized architecture.

The figure 7 is a comparison between the more greedy sensor of the optimized architecture with the more greedy sensor of the data-based architecture. It also includes a comparison between the fastest sensors of the two architectures. We notice that the optimized architecture is always better than the data-based one. However, the performance of the two architectures is very close. This proves that the overhead of the rule check and the maintenance of the rule-tree are not very time-expensive.

The comparison between the minimum and the maximum curves shows that the maximum processing time is approximately five times the minimum processing time. For Example, with 6000000 packets, the minimum

processing time is about 2 seconds (sensor 3) and the maximum processing time is about 10 seconds (sensor 1). This result proves that the sensors are not load balanced. This is due to the huge number of alerts generated by the sensor 1 (about 1000 alerts) against a few number of alert generated by the sensor 3 (87 alerts).

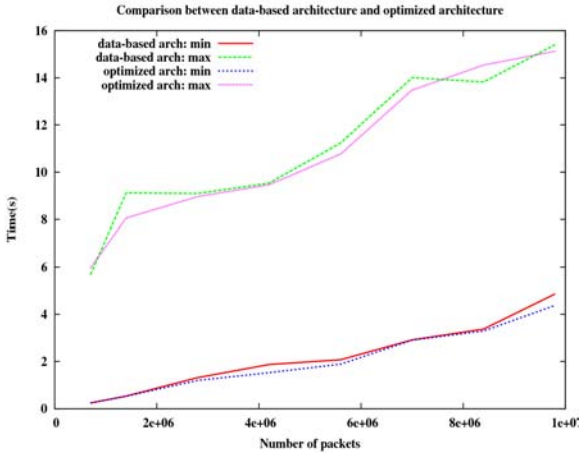

Fig. 7: comparison between the optimized-based architecture and the data-based architecture

This optimized architecture is based on a traffic splitting among sensors with a corresponding rule splitting which lead to a no-load balanced rule distribution. This figure proves that a load balanced architecture must take into account the vulnerable ports/servers as an important parameter in addition to the amount of traffic and the number of rules.

5. Conclusion

In conclusion, the performance study proves that our distributed IDS architecture can effectively decreases the bottleneck caused by the intrusion detection processing and analysis for high-speed networks. It has a better scalability and flexibility with its switch-based splitting structure. Based on this architecture, the IDS will effectively monitor the backbone network for security and helps in the evaluation and forecast of network security situations. As future work we will investigate in the design of algorithms and techniques for the security event correlation to improve the real-time security defending against the security attacks. We will try also to develop some network scanning and vulnerability assessment techniques to reduce the false alarms and therefore improve the efficiency of intrusion detection task.

6. References

[1] Snort:. http://www.snort.org/, 2008.

[2] SUN Qin-dong, ZHANG De-yun, GAO Peng and ZHANG Xiao. Study of Parallel IDS Load Balancing Algorithm. Mini-microsystems, 2004, 25(12):2215-2217.

[3] LU Zhi-Jun, ZHENG Jing, HUANG Hao. A Distributed Real-Time Intrusion Detection System for High-Speed Network. Journal of Computer Research and Development, 2004, 41(4):667-673.

[4] Tarek Abbes, Alakesh Haloi, Michaël Rusinowitch. High Performance Intrusion Detection using Traffic Classification. Proceedings of the IEEE International Conference on Advances in Intelligent Systems (AISTA2004), Luxembourg , Nov 2004.

[5] T. Abbes, A. Bouhoula, and M. Rusinowitch. A traffic classification algorithm for intrusion detection. In AINA Workshops (1), pages 188–193, 2007.

[6] L Schaelicke, K Wheeler, C Freeland. SPANIDS: A Scalable Network Intrusion Detection Loadbalancer. Proceedings of the 2nd Conference on Computing Frontiers, Ischia, Italy, 2005.

[7] I.Charitakis, K.Anagnostakis, E.Markatos. An Active Traffic Splitter Architecture for Intrusion Detection. Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2003), Orlando, October 2003. 238- 241

[8] JIANG Wenbao, HAO Shuang, DAI Yiqi, LIU Tinghua. Load Balancing Algorithm for High-speed Network Intrusion Detection Systems. Journal of Tsinghua Univ (Sci&Tech), 2006, 46(1):106-110.

[9] I. Cisco Technology. Parallel intrusion detection sensors with load balancing for high speed networks. 1999.

[10] C. Kruegel, F. Valeur, G. Vigna, R. Kemmerer. Stateful Intrusion Detection for High-Speed Networks. Proceedings of the IEEE Symposium on Security and Privacy. Los Alamitos, alifornias: IEEE Press, May 2002. 285- 293.

[11] D. L. Schuff, Y. R. Choe, and V. S. Pai. Conservative vs. optimistic parallelization of stateful network intrusion detection. In PPoPP ’07: Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming, pages 138–139, New York, NY, USA, 2007. ACM.

[12] A. M.S., J. Q., A. M., R. M.R.U., and A. M.B. Adaptative load balancingarchitecture for SNORT. In IEEE International Networking and Communications Conference (INCC’04), 2004.

[13] http://www.defcon.org

Hassen Sallay received the B.E degree from national college of computer science of Tunisia in 1999. He received the M.E. degree and Dr. Eng. degree from Henri Poincarre Univ. in 2000 and 2004, respectively. After working as a teaching assistant (from 2000) in Ecole de Mines in France, He has been an assistant professor (from 2004) in the Dept. of Computer Science and Information, Imam Mohammad ibn Saud Univ. His research interest includes Network and information Security Management, Network Management and Multicast services. He is also interested to the Arabization research fields. He is the leader of a research project funded by KACST in Network Security field as well as a steering committee member of Arabization IT research program.

Khalid AlShalfan Khalid A. Al-Shalfan, BEng, M.Sc, PhD in Computer Vision, (Computer Science) from the University of Bradford. Recently He is assistant professor at the College of Computer and Information Sciences, Al-Imam Muhammad Ibn Saud Islamic University. His Research activities are in Information Security and Computer vision. He is the leader of the IT research program of the Saudi IT research plan in Imam Mohammad Ibn Saud University.

Ouissem Ben Fredj received the B.E degree from university of sciences of Tunis, Tunisia in 2002. He received the M.E. degree from Henri Poincarre Univ., France in 2003 and Dr. Eng. degree from University of Evry Val d'Essonnes, France in 2007, respectively. He worked as a teaching assistant in the National Institute of Telecommunications in France (from 2003 to 2005) and in the Université du Sud - Toulon - Var (from 2006 to 2007). He has been an assistant professor (from 2008) in the Dept. of Computer Science and Information, Imam Mohammad ibn Saud Univ. His research interest includes parallel and distributed systems, high-speed networks, Network and information Security, and voice over IP. He is also interested to the Quran research fields. He is an assistant researcher of a research project funded by KACST in Network Security field.