Enhancing Intrusion Detection System Using Distributed Environment

Safaa O. Almamory, **Wesam S. Bhaya,** **Anees M. Hadi**

College of Computer Technology, College of Computer Technology , College of Science for woman,

University of Babylon, University of Babylon, University of Babylon

safaa_vb@yahoo.com wesambhaya@yahoo.com anees_alhamzawi@yahoo.com

07813554590 **07801672425** **07801342940**

Abstract:

The study of Intrusion Detection System (IDS) has become an important aspect of network security. As soon as the IDS detects a set of attacks it will generate many alerts referring to as security breaches. Snort IDS is an open source IDS which is suffer from flooding vulnerability if it will be used with fast network. To solve this problem, we suggest using several Snort sensors with one packet classifier. Each one of these sensors contains a part of the overall signatures. A packet classifier is built to distribute the flow over these Snort sensors, and to avoid spread of the same attack over more than one sensor and then avoid detection. The experimental results, on DARPA 1999 data set, have shown that the efficiency of Snort is enhanced by about 30%.

1. Introduction

An intrusion is an active sequence of related events that deliberately try to cause harm, such as rendering system unusable, accessing unauthorized information, or manipulating such information. There are two types of intrusion systems:

Intrusion Detection System (IDS), and Intrusion Prevention System (IPS). IDS is simply trying to detect the unauthorized penetration by the intruder in the network. The basic process for the IDS is to passively collects data, Preprocesses and classifies them. There are two IDS types Host-based Intrusion-Detection System (HIDS), Network-based Intrusion-Detection System (NIDS). IPS sits

inline on the network and monitors it. It takes action based on prescribed rules when an event occurs.

IPS has different functionalities and strengths from the IDS. Also, IPS is considered as a blocking product (proactive) while the IDS, is considered as an alerting product (reactive) [1].

There are some of IDS open source softwares (or called network sensors); like Advanced Intrusion Detection Environment (AIDE) [2], Open Source Host-based Intrusion Detection System (OSSEC) [3], Suricata which is the Host-based intrusion-detection and prevention system. [4], Prelude Hybrid IDS which is host-based and network-based IDS [5], Bro NIDS [6], and Snort is the network-based (IPS/IDS). Snort is combining the benefits of signature, protocol, and it is the most widely deployed IDS/IPS technology worldwide [7].

The main aim of this paper is to improve the Snort IDS sensor work by finding new methodology that will permit to process more packets than Snort in normal case.

We will build classifier able to sniff packets and classify them according to its protocols using suitable packet classification algorithm.

These protocols refer to different TCP/IP layers protocols. After packet classifying according to its protocols, the packets will forwarding to different Snort sensor nodes for processing.

This paper is organized as follows. Section 2 surveys the related works.

Section 3 provides preliminaries about the proposed system like Snort IDS and packet classification. Section 4 proposes our system. The results and experiments are presented in section 5. Finally, Section 6 concludes this paper.

2. Related Work

Several researchers tried to enhance the work of Snort which includes Snort rules, Snort components, content matching, compilation, distributed systems, parallel systems, and other fields. Wheeler P. *et al*. [8] approach proposes to use a distributed system where in which each node has the same rules. The packets are sent to each node using load balancer. Thus, each node will receive 1/N from original load, where N is the number of nodes.

Kopek C. *et al*. [9] approach proposed that the packets are divided into fragments and forwarded to the array of processors. Each fragment could process independently from other fragments. Match-bit field is assigned to each packet, which allows one processor to quickly indicate to other processors that a match-bit field has been found for a given packet.

When a notification is received, the remaining processors can start inspecting another packet.

Sanz-Bobi *M. A. et al*. [10] approach

proposes an intelligent system for automatic detection of intrusions in computer networks (Intrusion Detection System based on Artificial Intelligence IDSAI). Its architecture is based on multi-agent system in which several types of agents cooperate together to perform a fast and reliable detection of intrusions.

Schuff D. L. *et al*. [11] approach bases on Snort 2.6 version. It only executes multiple instance of original Snort in parallel and use load balancer that distributes task queues to dispatch traffic. After distributing tasks by load balancer, each processing thread does a job, such as decode, preprocess, and detect.

Yoshioka A. *et al*. [12] Approach generates a hash value using five protocol fields for each rule. The five protocol fields are source IP, destination IP, protocol type, source port, and destination port. Each hash value is stored in a single hash table. When a packet arrives, the same hash function is applied to the incoming packet and the hash value generated from the packet is used to search in the hash table for the matching rules. This approach searches for the matching rules multiple times against each packet to cover all possibilities.

3. Preliminaries

This section provides an introduction about snort NIDS (Subsection 3.1)
and packet classification which appears in Subsection 3.2.

3.1 Snort as Intrusion Detection System

Snort logically consists of five components. These components work together to detect particular attacks and to generate output in a required format from the detection system [13]. The first component is the packet decoder which takes packets from network interface via the libpcap library, and determines which protocol is in use for a given packet. As a preprocessors plug-ins component, this component can be used with Snort to arrange or to modify data packets before the detection engine does some operation to find out if the packet is used by an intruder.

The third component is the detection engine. The packets are sent to detection engine by the decoder and preprocessor plug-ins components. These packets are compared against the Snort rules; if a packet matches any rule, the output plug-ins will activate, otherwise, the packet will be dropped. The detection plug-ins, as the fourth component, provides additional detection functions on the packets. Finally, there are many outputs plug-ins used by Snort to save generated output (logs/alerts) from the previouscomponents.

Several reasons encouraged us to select Snort NIDS. First, Snort is one of the more mature open source packages, which is free of charge. Second, there is an ability to program and reconfigure this software. Third, bugs for Snort are virtually nonexistent due to

periodic industry maintains and support. Fourth, the preprocessors arrange or modify the network data flow in real time to increase the chances of a signature noticing a malicious packet. Fifth, Snort is very flexible in the ways it can be used with network as a packet sniffer, packet logger, or as IDS. Finally, industry support where the Snort open source community keeps Snort signatures up to date in order to identify the new attacks.

The NIDS can be evaded by flooding it using noise or dummy traffic. The flooding technique requires the NIDS to utilize valuable CPU and memory resources to analyze the dummy traffic. If the performance of the NIDS is not sufficient to handle the traffic generated by flooding, the actual intrusive activity performed by the intruder may be missed, due to the excessive dummy traffic generated. Even if the real attack is detected, the NIDS response may be significantly delayed due to the exhaustion of CPU and memory resources on the NIDS [14]. In this paper, we will increase the efficiency of Snort IDS to avoid the flooding problem.

3.2 Packet Classification

Packet classification is the process of categorizing the packets according to its header fields. This process is applied in the forwarding machine (like router) to identify the context of the packets and to perform important actions.

The criteria for classifying packet is called rule R, and the set of finite rules R_1, R_2, …, R_n contained in forwarding machine is called rule database or a classifier [15]. The fields of rule and packet header are related; for example, the rules that implement IPv4 consist of 5 fields (source IP address, destination IP address, protocol type, source port, and destination port). The incoming packet to router matches specific rule if the distinct fields in the packet match the corresponding filed in that rule [16].

In this paper, we have used a packet classification algorithm which uses hash tables. The reason for choosing hash table based algorithm is that we have a small number of rules; in addition, it has time complexity O(1).

4. Proposed System

As we said above, the main idea of this paper is to use packet classification algorithm to enhance processing speed of Snort IDS. This is done by distribute the packet flow over a set of Snort sensors. Figure 1 depicts the main components of the proposed system. The packet classifier online receives the packet flow and distributes the flow over snort sensors. These sensors monitor the flow for any suspicious activity and trigger an alert for every detected attack. The triggered alerts by these sensors will go to a central database which is monitored by the security analyst.

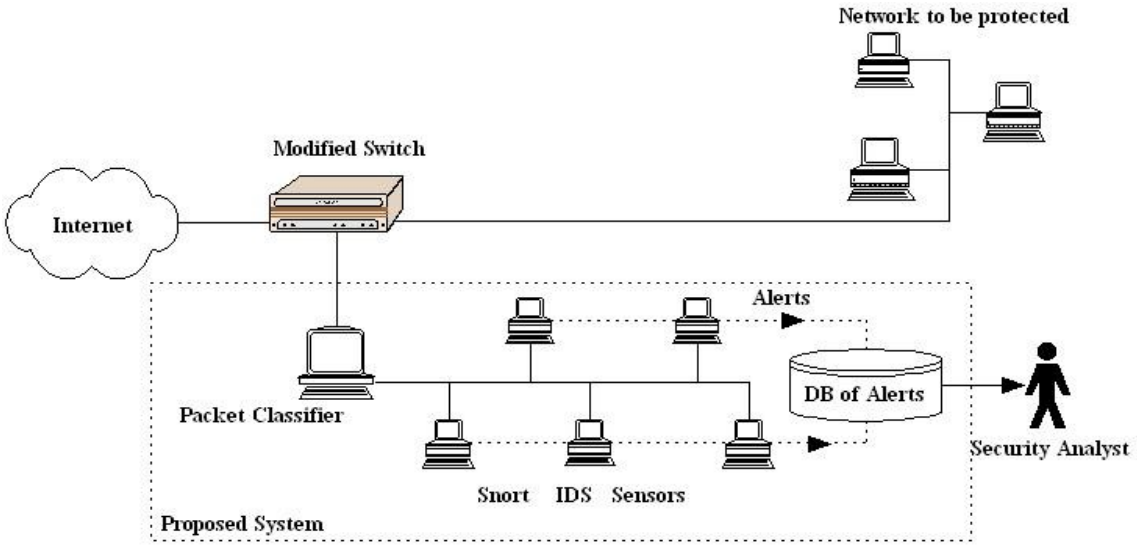

Figure 1: main components of the proposed system. The packet classifier online receives the packets and distributes the flow over snort sensors.

One critical point about the classifier is how to classify packets and guarantee that the packets of the same attack go to the same sensor. For this reason, we clustered the signatures by the type of protocol and distribute these signatures over Snort sensors. Every Snort sensor has to monitor partial set of the whole rules.

Algorithm 1 demonstrates the main steps of the proposed system. It receives a real-time packet flow and produces a set of alerts. It composed of two parts. The first part (Lines 2-6) is related to packet classifier. The classifier decodes every packet to extract protocol type. According to protocol type, the classifier forwards the packet to the suitable Snort sensor. The second part of the algorithm (Lines 7-10) demonstrates the work of every Snort sensor.

Algorithm 1: Classification for detection Input: a real-time sequence of packets Output: a sequence of alerts

Begin

1. duplicate network traffic and forward it to the packet classifier;

// Packet classifier

2. foreach packet PKT received do{

3. P=protocol type extracted from PKT header;

4. using hash table to find IP address of snort sensori monitoring P;

```
5.      modify PKT header with new destination IP address;
6.      forward PKT to snort sensori;}
// Snort sensori; it works in parallel with packet classifier
7. foreach PKT received do{
8.      foreach S □ partial signature setI  do {
9.       if PKT contains S then
10.           report alert;}}
End
```

5. Results

The experiments were implemented in academic CISCO lab at the University of Babylon under Linux operating system. Snort IDS software version 2.8.5.1 had been used in the experiments. The number of rules that we use are 3306 rules. In addition, we used DARPA datasets [17] to evaluate our system in real-time. We used this dataset to measure the load on CPU, RAM, and disk swapping for the Snort in a normal case and for the classifier that we built.

The proposed system is divided into two stages, the first stage is concerned with how to distribute Snort rules over some nodes, and the second stage is concerned with how to test that system; these two stages are stated in the following two subsections, respectively.

5.1 Snort Rules' Distribution

The DARPA 1999 [17] is an off-line dataset to evaluate the intrusions detection system. These dataset includes data for five weeks. Three weeks of training data with background traffic and labeled attacks and two weeks of test data with background traffic and unlabeled attacks. A rich variety of background traffic that looks as if it were initiated by hundreds of users on thousands of hosts is generated in the test bed.

Instead of using one Snort IDS sensor with complete rules, we used five Snort sensor nodes with partial rule-set. In this case, we have to distribute Snort rules in order to process the network traffic in parallel fashion. Table 1 shows the protocols distribution over the five sensor nodes, whereas Figure 2 show as the Snort rules distribution according to its protocols for system to sensor nodes.

Table 1: Rules distribution according to its protocols.

Sensor node¡	Monitored protocol(s)
1	IP,UDP,ICMP
2	HTTP
3	SMTP, ORACLE
4	FTP, NETBIOS, SQL, MYSQL, EPMAP, MICROSOFT DS, SSH
5	TELNET, POP, TIME, IRC(CHAT), IDENT, FINGER, SNMP, IMAP, DNS, NNTP, Others Prot.

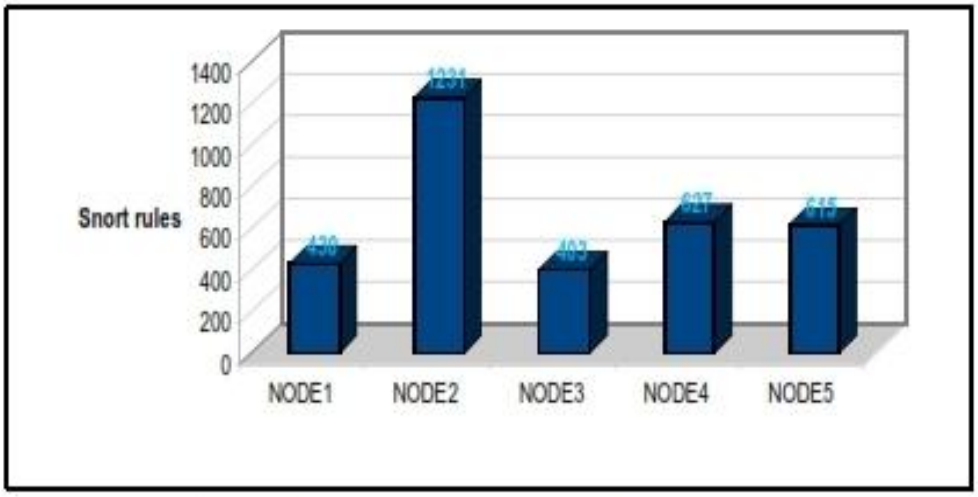

Figure 2: The suggested Snort rules' distribution.

In Figure 2, we show that the Node2 has the largest amount of Snort rules because of its protocol distribution in DARPA 1999 dataset. In spite of the fact that node2 has the most number of rules, these rules should be placed in one sensor for detection purposes.

5.2 Resources Consumption Experiments

We use the TCPREPLAY program for replaying the DARPA 1999 dataset. Because of the large size of inside.tcpdump file, it takes very long time in replaying. Thus, we replayed it with 10MB/Sec because this speed is considered as a suitable speed since it is not lost in packets or slowly in packets read, and the time required to replay inside.tcpdump file is about from 240 to 250 second.

At same time we replayed the dataset, we had computed the load on the PC resources using ATOP benchmark software. We use ATOP program to calculate the consumption on the CPU, memory, and disk swapping on a Linux system. By applying ATOP command line with –C option, we computed the average CPU consumption for the Snort in normal case, and the classifier as shown in Figure 3.

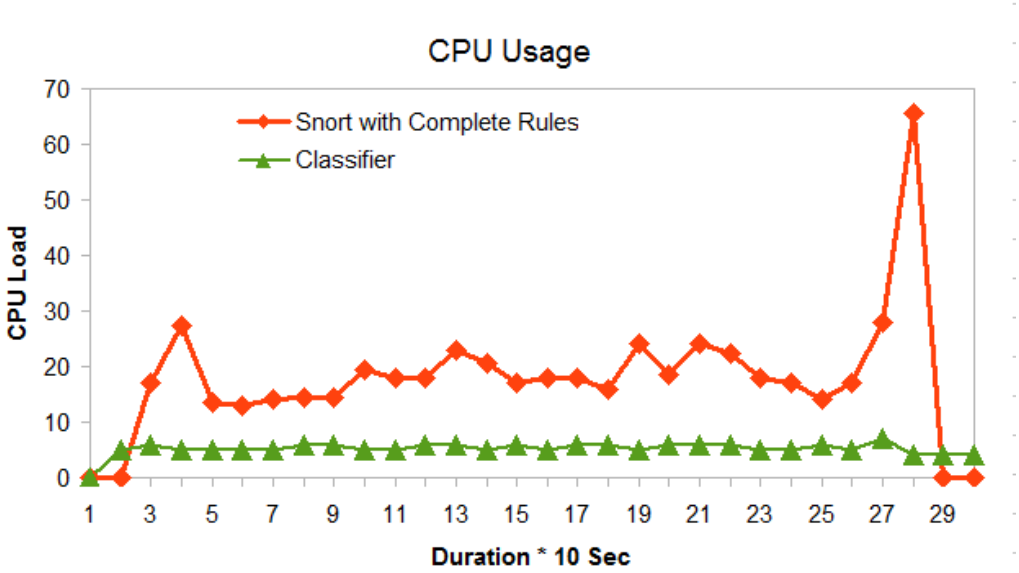

Figure 3: CPU usage for the classifiers compared with the Snort in normal case.

In the Figure 3, we note that the classifier has less CPU consumption than the Snort IDS in normal case (Snort sensor that have the complete rule- set). It should be noted that Snort IDS in normal case suffered from flooding in the 290 second and became out of service.

Figure 4 explains the CPU consumption on sensor nodes which belong to our system.

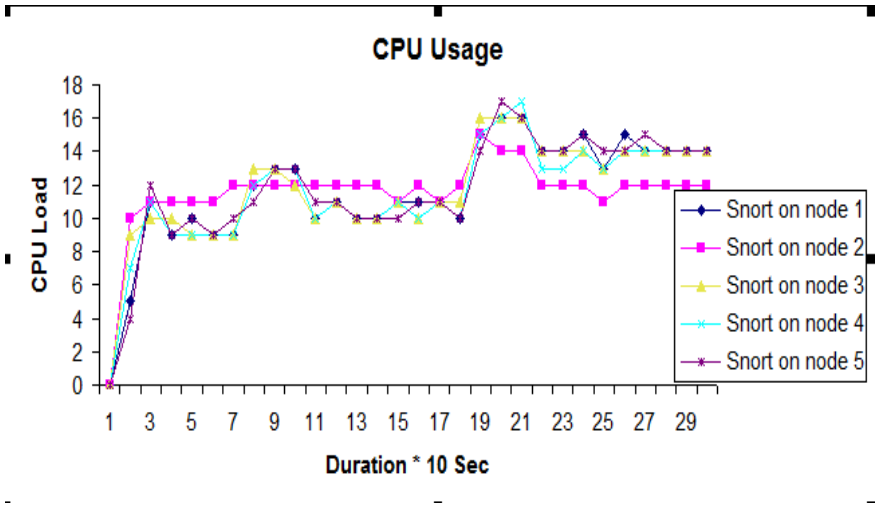

Figure 4: CPU consumption for the sensor nodes associated with our system.

By ATOP command line with –M option, we computed the average memory consumption for the Snort in normal case, and the classifier as shown in Figure 5.

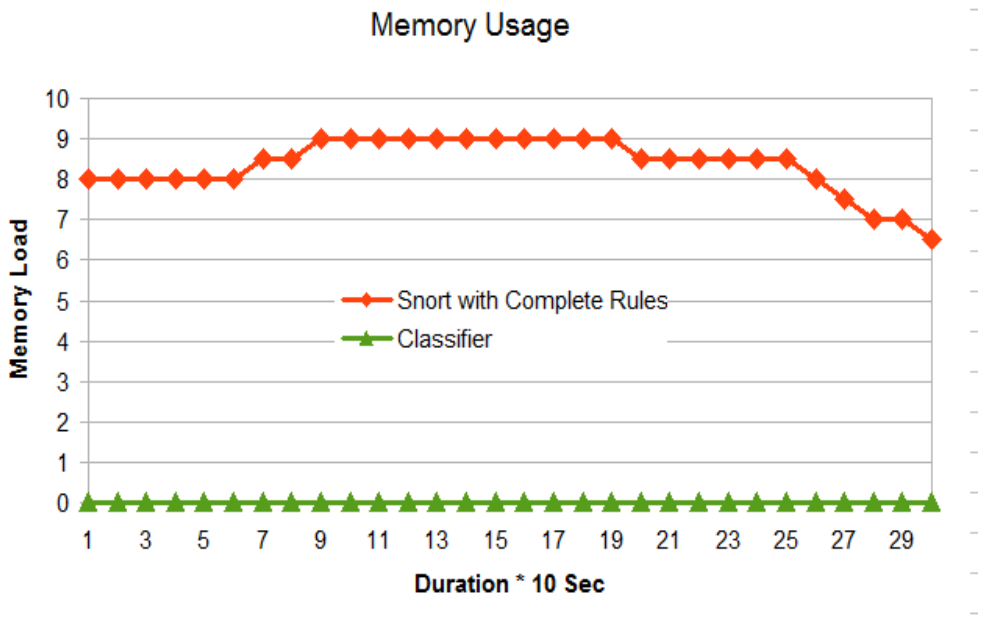

Figure 5: Memory usage for the classifier compared with the Snort in normal case

By applying the ATOP command line with –D option, we computed the average disk swapping consumption for the Snort in normal case, and the classifier as shown in Figure 6.

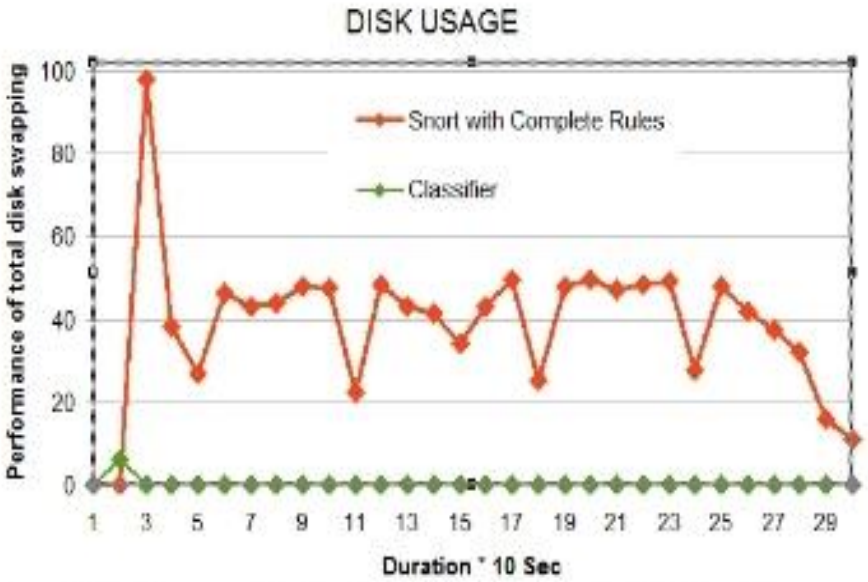

Figure 6: Disk usage for the classifiers compared with Snort in normal case

Table 2 explains the packets throughput by the Snort IDS in normal case compared with the classifier with different dataset replay speed as shown in Table 2.

Table 2: Average number of packets processed by the classifiers compared with Snort in normal case

No.	Software	Dataset file replaying speed	Average number of packet processed
1	Snort in normal case	10Mbps	418
		92Mbps	3731
3	Classifier	10Mbps	4370
		92Mbps	9832

6. Conclusions

Several researchers have worked on enhancing Snort IDS using different techniques. However, that process is a trade off between speed and detection accuracy. The aim of this paper is to improve efficiency of Snort IDS by distributing Snort rule-set over five Snort sensors and a classifier. We have concluded that the efficiency of the Snort IDS has been increased by about 30% after applying our system on DARPA 1999 dataset. A little increasing in efficiency is expected when applying the proposed system with real packet flow.

References

[1] Endorf C., Schultz E., and Mellander J., "Intrusion Detection & Prevention", McGraw-Hill, 2004.

[2] Advanced Intrusion Detection Environment (AIDE), [aide. sourceforge. net], 2011.

[3] Ossec IDS, www.ossec.net, 2011.

[4] Suricata Intrusion Detection and Prevention Engine, redmine. openinfosecfoundation.org/projects/show/suricata, 2011.

[5] Prelude Hybrid Intrusion Detection System, prelude-ids.org, 2011.

[6] The Bro Network Security Monitor, www.bro-ids.org, 2011.

[7] Roesch, M., "Snort-lightweight intrusion detection for networks," In Proceeding of USENIX LISA Conference, pp. 229–238 (1999)

[8] Wheeler P., and Fulp E. W., "A taxonomy of parallel techniques for intrusion detection", In Proceedings of the 45th Annual Southeast ACM Regional Conference, Winston-Salem, North Carolina, USA, P.P 278-282,2007.

[9] Kopek C. V, Fulp E. W, and Wheeler

P. S., "Distributed Data Parallel Techniques for Content-Matching Intrusion Detection Systems", In Proceeding of MILCOM 2007 IEEE Military Communications Conference (2007), PP 1-7, 2007.

[10] Sanz-Bobi M. A., Castro M., and Santos J., "IDSAI: A Distributed System or Intrusion Detection Based on Intelligent Agents," In Proceedings of the Fifth International Conference on Internet Monitoring and Protection, pp.1-6, 2010.

[11] Schuff D. L., Choe Y. R., and Pai V. S., "Conservative vs. optimistic parallelization of stateful network intrusion detection," in proceeding of the 12th ACM SIGPLAN symposium on Principals and Practice of parallel programming, 2007

[12] Yoshioka A., Shaikot S. H., and Kim M. S., "Rule Hashing for Efficient Packet Classification in Network Intrusion Detection," In Proceeding of the International Conference on Computer Commu

[13] Beale J., Foster J. C., Jeffrey Posluns, and Brian Caswell ,"Snort 2.0 Intrusion Detection ", Syngress Publishing , 2003.

[14] Menga J., and Timm C., "CCSP Secure Intrusion Detection and SAFE Implementation Study Guide ", SYBEX, 2004.

[15] Madhi D., and Ramasamy K., "Network routing algorithms, protocols, and architectures", Morgan Kaufmann, USA, pp. 1-957, 2007.

[16] Varghese G., "Network Algorithmics An Interdisciplinary Approach to Designing Fast Networked Devices ", Morgan Kaufmann, USA. pp. 1-491, 2005.

[17] Lippmann R., Haines J. W., Fried D. J., Korba J., and Das K., "Analysis and Results of the 1999 DARPA Off-Line Intrusion Detection Evaluation", Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, p.162-182, October 02-04, 2000.

تحسين أنظمة اكتشاف الهجومات باستخدام البيئة الموزعة

د. صفاء عبيس مهدي د. وسام سمير بهيه أنيس محسن هادي
كلية تكنولوجيا الحاسبات كلية تكنولوجيا الحاسبات كلية العلوم للبنات
جامعة بابل جامعة بابل جامعة بابل

أصبحت دراسة أنظمة اكتشاف الاختراقات من الثوابت المهمة في أمنية الشبكات . حالما يكشف نظام الاختراقات أي هجوم ، فأنه يقوم بتوليد مجموعة من التحذيرات التي تشير الى وجود خرق أمني في الشبكة . برنامج Snort المفتوح المصدر هو احد أنظمة اكتشاف الاختراقات المجانية ولكنه يعاني من مشكلة البطؤ نسبيا اذا ما تم استخدامه مع الشبكات السريعة . لحل هذه المشكلة ، نقترح استخدام عدة نسخ من برنامج Snort مع مصنف حزم . كل نسخة من هذا البرنامج تحتوي على جزء من المجموعة الكلية للهجومات المراد مراقبتها . أما دور مصنف الحزم فهو توزيع الحزم على النسخ المستخدمة من برنامج Snort وكذلك لضمان عدم ارسال الحزم الخاصة بهجوم واحد الى اكثر من نسخة من برنامج Snort وبالتالي تجنب الاكتشاف . النتائج العملية مع بيانات DARPA 1999 بينت ان كفاءة البرنامج Snort قد تحسنت بنسبة 30% .