International Journal of Computer Applications (0975 – 8887)
Volume 41– No.1, March 2012

# Managing Intrusion Detection as a Service in Cloud Networks

Hatem Hamad
The Islamic University of Gaza
Gaza, Palestine

Mahmoud Al-Hoby
The Islamic University of Gaza
Gaza, Palestine

## ABSTRACT

Cloud computing is frequently being utilized to eliminate the need to local information resources. In this paper, we address the problem of intrusion detection in cloud environments and the possibility of allowing intrusion detection to be provided to clients as a service. The paper describes the Cloud Intrusion Detection Service (CIDS), which is intended to function as an intrusion detection web service to be provided for cloud clients in a service-based manner. CIDS utilizes the "Snort" open source intrusion detection system. The operating logic and user access webpages were developed using J2EE. We implemented a proof-of-concept prototype to evaluate the performance. CIDS was proved to be very friendly to resource allocation. Additionally, CIDS gave better attack detection rates and attack detection times than other solutions. These improvements can be beneficial to both cloud providers and cloud subscribers alike.

## General Terms

Cloud Computing, Intrusion Detection Management

## Keywords

Cloud Computing; CRE; Intrusion Detection; SaaS

## 1. INTRODUCTION

Cloud computing is a large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet [1]. The cloud computing paradigm is usually linked to SaaS or Software as a Service model [2]. This service model works by providing applications for end users on service-based manner. A recent addition to cloud services was security-related services, in what is termed as Security-as-a-Service [3]. Different systems have been made available to end user to provide the security products for users in a service-based manner. This included many product services and types like Remote Vulnerability Scanning [4], Web root's Email and web Security SaaS [5], and Panda's Managed Office Protection [6].

In this paper, we design and implement a security-related cloud service. More specifically, we design the Cloud Intrusion Detection Service (CIDS). CIDS is intended to be used as a service-based intrusion detection system for which cloud clients can subscribe with. The remaining of this paper is as follows. In section 2, we define the main problem of this paper. In section 3, we describe some of the currently published papers that are specific to cloud intrusion detection solutions. Later in section 4, we introduce the CIDS. This is where we describe our approach to the solution. After that and in section 5, we test and evaluate the different performance measures for this system. We conclude in section 6.

## 2. PROBLEM DEFINITION

Intrusion detection systems are commonly used by network administrators to monitor the traffic being exchanged between different network segments. And by replacing the traditional local server-based network environments with cloud-based network infrastructure, system administrators will need to purchase additional services from the cloud provider so that they can deploy their own network intrusion detection systems. This paper discusses the effective design of an intrusion detection system that can be integrated with the available services in cloud networks. The main idea is to provide intrusion detection as a service for the cloud users. This in turn will enable the clients to choose the protection settings they wish to utilize using a simple and easy-to-use web interfaces.

Currently, multiple research activities were introduced to address the issue of intrusion detection within cloud computing environments. These activities can be classified as those to detect intrusions against the cloud itself. And those that to detect attacks that target individual machines inside the cloud. Our study is on the latter type of the two. More specifically, it will cover the service-based or subscription-based intrusion detection. Which is a field that did not received as much attention as the classical intrusion detection activities.

The required intrusion detection framework has some desired criteria, where these are needed to comply with the traditional SaaS service models. These include the ability of users to subscribe or unsubscribe from the service, change subscription requirements (i.e. protection requirements), pay for size and complexity of subscription database, and to be an easy to use service.

## 3. CURRENT STATUS

Multiple research activities were introduced to address the issue of intrusion detection within cloud computing environments. Dastjerdi et. al. [7] implemented applied agent-based IDS as a security solution for the cloud. The model they proposed was an enhancement of the DIDMA [8]. The system is mainly designed to protect the networks' resources and cannot be customized as a service. Bakshi et. al. [9] proposed another cloud intrusion detection solution. The main concern was to protect the cloud from DDoS attacks. The model uses an installed intrusion detection system on the virtual switch and when a DDoS attack is detected. Despite being reported as effective, the model helps to protect the cloud itself, not the cloud clients who in turn don't have any kind of authority over the intrusion detection system being used. Another recent and significant contribution to this field is the work of Mazzariello et. al. [10] where they proposed a model for detecting DoS attacks against Session Initiation Protocol (SIP). The model is limited to detecting SIP flooding attacks and falls largely with the category of intrusion detection systems designed to protect the cloud itself.

Lo et. al. [11] proposed a framework that is mainly designed to create cloud networks that are immune against the Distributed Denial of Service (DDoS) attacks [12]. The utilized IDS implementation was the Open Source Snort IDS and the framework itself is designed as a Distributed Intrusion Detection System (DIDS) [13] [14]. The proposed framework supports the idea of cooperative defense by the IDS sensors in the cloud network. Within the framework, an IDS sensor is deployed in each network region. Any sensor will send out the alert to the other sensors while they are suffering from a severe attack defined in its block table. Each sensor exchanges its alerts and has a judgment criterion to evaluate the trustworthiness of these alerts. After evaluation, the new blocking rule is added into the block table if the alerts are regarded as a new kind of attack.

Roschke et. al. [15] have proposed an intrusion detection framework based on the VM-based IDS [16]. In their work, they have developed a general framework for intrusion detection. It consisted of separate IDS sensors for each virtual host. The IDS sensors can be of different vendors. To enable the collection and correlations of alerts from the different IDS implementations, an Event Gatherer was made to work as a medium to standardize the output from the different sensors as well as realize the logical communication. The cloud user can have access to both the applications and the IDS sensors. The users can access the sensors, configure, modify rule sets, and modify detection thresholds. Additionally, users can review the alerts generated when attacks that target their virtual hosts or services are spotted. The framework also includes the IDS Management module which is responsible for orchestrating the message passing and alert transfer among the different IDS sensors and the main storage unit whether it was a file system, a network database, or a shared folder. This approach of separating the IDS from the protected hosts is of great advantage. But it is criticized for requiring the large consumption of computing resources since every virtual application, platform, or host needs a separate VM-Based IDS.

## 4. CLOUD INTRUSION DETECTION SERVICE

The proposed framework builds upon the fact that intrusion detection systems utilize very fast and very efficient search algorithms [17]. So by increasing the complexity of the signature database definitions, we will be able to customize the behavior of the intrusion detection system in such a way that it acts as a cloud-capable intrusion detection system.

The proposed system is therefore nicknamed Cloud Rule Engine (CRE) and is capable of receiving the subscriptions requests from the cloud users and translates these requests into a standardized signature database that can then be deployed and utilized as the Cloud Intrusion Detection Service (CIDS). This process will convert standard intrusion detection system into a fully capable system of handling the cloud variations. Figure1 summarizes this process.

Figure2 displays the Use-Case diagram of the CIDS Web Service. As the model shows, the main actors in the CIDS Web Service are the clients and administrators. The CIDS clients need first to login before they can call the different functions available. For example, a client might view the categories currently supported which in turn calls a special function that view the number of signatures in the selected category. The client then may like to subscribe in the selected category based upon the description available and the number of attack signatures definitions within it.

To do so, the client can *subscribe to category*. Later the client may wish to *view the attacks* detected on his own protected resources. The client may not like to activate the selected package, so he can *unsubscribe from* the category or even *remove his subscription* totally.

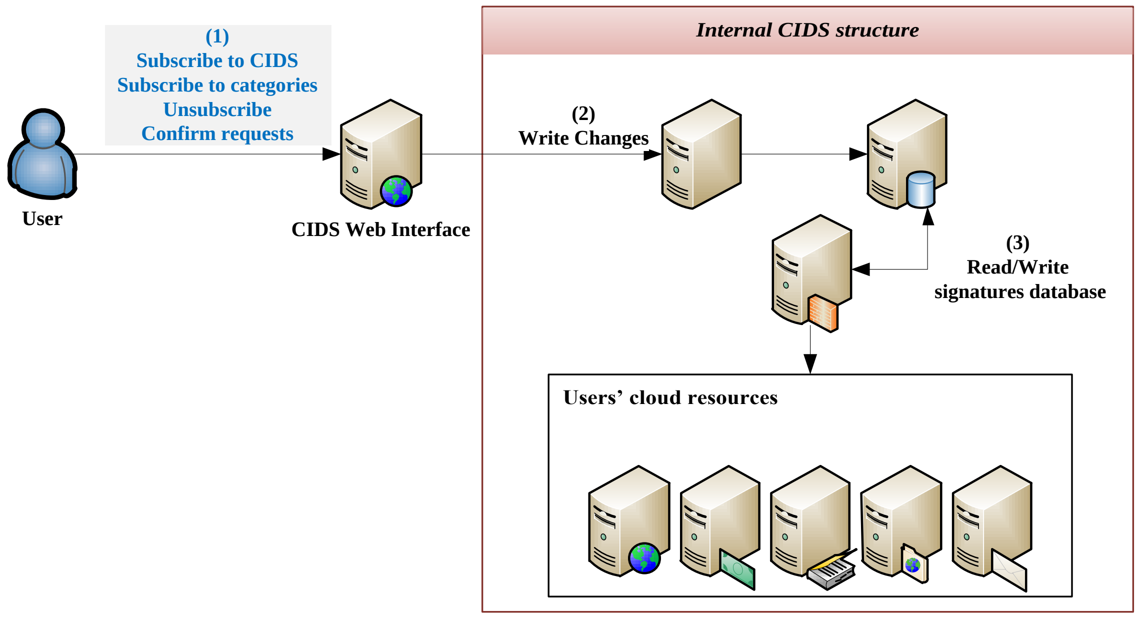

**Figure 1 – IDS Service within the Cloud**

On the other hand, the administrators will also need to *login* before using or managing the CIDS. He can *view the current subscribers* or *view the categories*. The administrator may have defined a new protection package, he can *add the category* to the system or even remove *the category* if it gives inaccurate results or is not popular among clients. The administrator may also *view alerts* by a certain user (i.e. client) or even *view the alert summary* for all clients. The administrator can also *remove the users' subscription* himself for whatever reasons

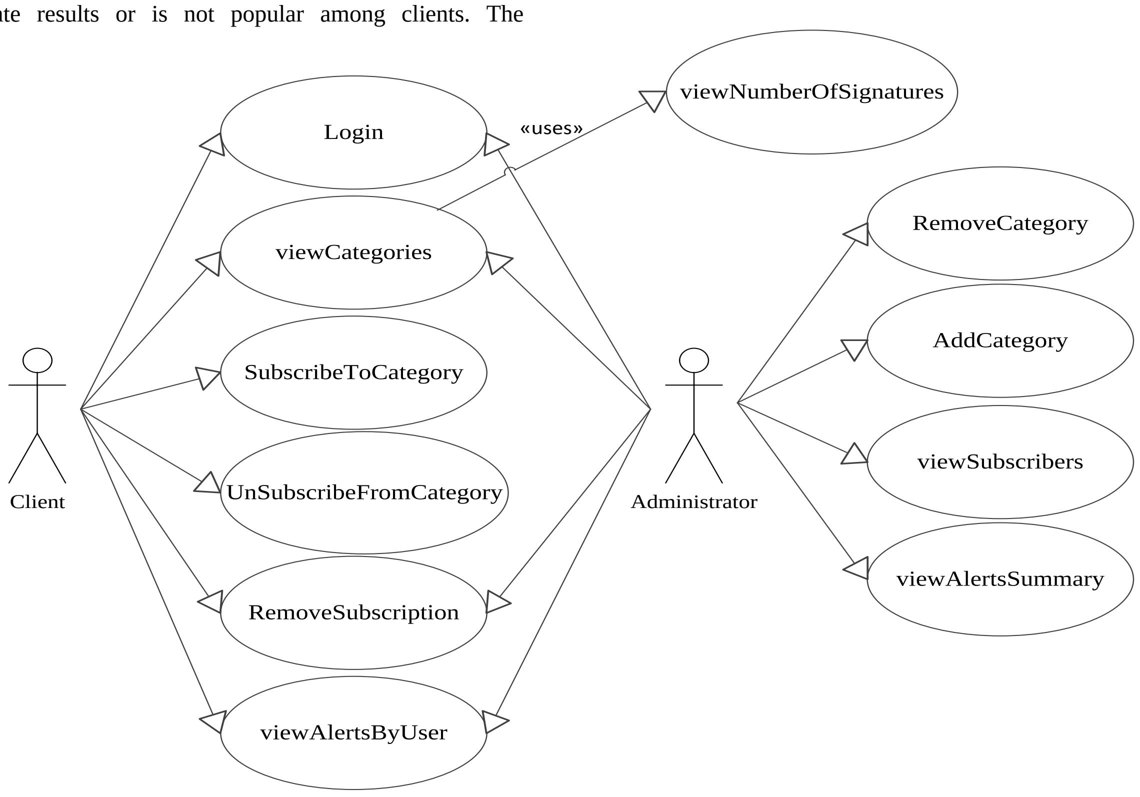

**Figure 2 – CIDS Use-Cases**

## 4.1 Cloud's Rules Engine

The most critical and important part of the service-oriented intrusion detection system for cloud networks. As mentioned earlier, CRE works on different layers with varying complexities. These Layers are the User Layer, The System Layer, and the Database Layer. The User Layer includes the interface that will enable the cloud subscribers to define the subscription and protection requirements. The user in this case can include both the cloud's clients and administrators alike. The common thing is that they can easily access the configurations, the subscription details, and the security monitoring and alerting system. This layer sends the different requests to the other layers in order to convert them to actual IDS runtime-configurations. The second layer is the System Layer. This layer will be the driver for the IDS service and can understand both the alerting mechanism and the signature syntax. It can do the actual translations to IDS signature database and also provides the required Application Programming Interface (API) for accessing the alerts database. The third layer is relatively relaxed layer. The Database Layer's task is to track the subscribers' settings and to enable fast access to their settings for any later updates either to the network segment or to the subscription details. Figure 3 depicts these layers and their interactions within the system. The CIDS API provides different functions and operations as part of the proposed CIDS framework. These operations are displayed in table 1.

**Table 1 – CIDS OPERATIONS**

| ***Type*** | ***Operation*** | ***Target User*** |
|---|---|---|
| Category | AddCategory | Administrators |
| | RemoveCategory | Administrators |
| | ResetToDefaultSettings | Administrators |
| | viewNumberOfSignatures | Administrators, Subscribers |
| Subscription | ViewSubscripers | Administrators |
| | AddSubscription | Administrators, Subscribers |
| | RemoveSubscription | Administrators, Subscribers |
| | SubscribeToCategory | Subscribers |
| | UnSubscribeFromCategory | Subscribers |
| Alerting | viewAlertSummary | Administrators |
| | viewAlertsByUser | Administrators, Subscribers |

As the figure illustrates, if the user's request is related to his subscription details, the request is forwarded to the database layer. And if the requests are related to the IDS operations then the requests are forwarded to the system layer. Additionally, the communication between the system layer and the database layer is needed so that the user's preferences that exist in the database layer can be translated into runtime-configurations that are effective at the system layer.

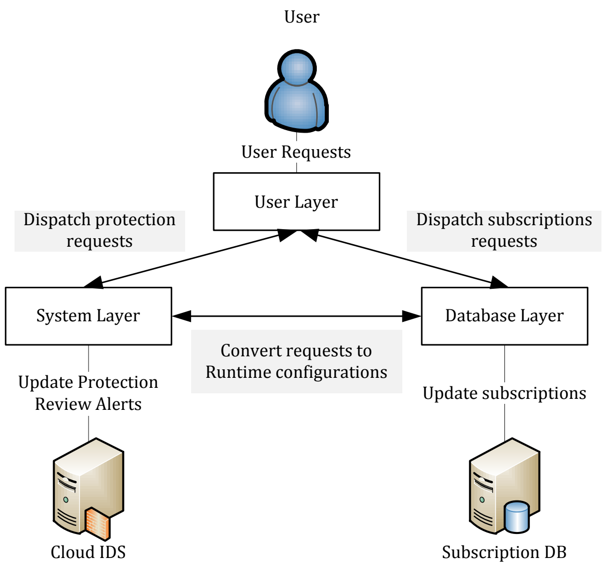
User
User Requests
User Layer
Dispatch protection requests
Dispatch subscriptions requests
System Layer
Database Layer
Convert requests to Runtime configurations
Update Protection Review Alerts
Update subscriptions
Cloud IDS
Subscription DB

**Figure 3 – CIDS Layers and Interactions**

## 5. Implementation and Results

As a proof of concept implementation, The CIDS was implemented using the Java 2 Enterprise Edition (J2EE) [18]. The entities in the UML model were translated to Java-based classes. The web interface was created using Java Server Pages (JSP) [19] technology on Ubuntu [20] Linux distribution. For subscription, we used a simple signatures database with approximately 500 signature definitions.

The CIDS system model was successfully implemented and tested. The final implementation contained two simple web folders for cloud administrators and cloud subscribers. The implemented interfaces for both types of users were made as a proof-of-concept only while the real implementation may include more complex interfaces. However, this was not the main concern of this paper, which focuses more on providing general framework for cloud-capable service-based intrusion detection system.

Table 2 reviews some of the currently published researches on integrating intrusion detection systems with cloud computing networks. As stated in the table, the CRE have enough advantages –based on the initial design requirements– that the other cloud IDS solutions doesn't have.

**Table 2 – Cloud IDS Comparison**

| System | CIDS | Roschke | Lo |
|---|---|---|---|
| Service-Based | Yes | Yes | No |
| Customized Subscription | Yes | Yes | (N.A) |
| Client-Oriented | Yes | Yes | No |
| General Protection | Yes | Yes | No |
| Fully Parallelizable | Yes | No | Yes |
| Separated IDS | Yes | No | Yes |

Due to the unique requirements that are set before designing and implementing the CRE component of the Cloud Intrusion Detection System (CIDS), the results obtained in this section are the results of two testing scenarios. The first Scenario works by comparing the performance of shared-Profile scheme in CIDS with the basic single-client scenario to determine the overhead of using CIDS over traditional implementation. The second scenario compares the CIDS with the case of using separate profiles for each client. For the two scenarios, the parameters that are obtained for comparison are total memory consumption, attack detection rates, and time required to process each attack packet.

### 5.1 First Scenario

This scenario aims at measuring CIDS overhead. It consists of using CIDS to protect 50 networks and using simple snort to protect a single network. The rules vary from using 200 signatures to 1000.

Figure 4 displays the results for the attack detection rates for the two cases. CIDS was able to detect a very high rate of attacks that was targeting either of 50 networks. The rate is slightly less than the one obtained by protecting a single network. This is an expected behavior since we use the same number of signatures and because the search algorithms that are usually utilized are very efficient in terms of the complexity of the patterns in question.

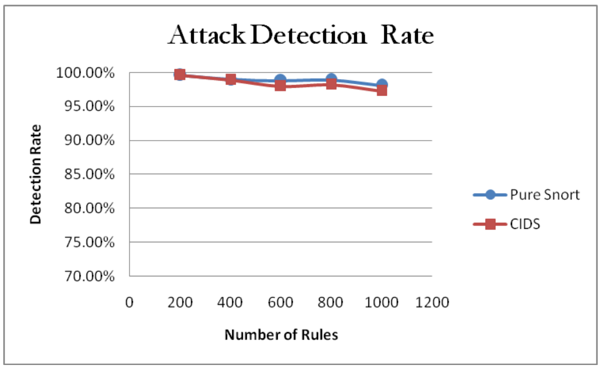

**Figure 4 – Attack detection rates**

Figure 5 displays another relatively good result for CIDS. The overhead required to detect a single attack that targets either of 50 networks is only 3 milliseconds more that the time required for detecting a single attack against a single network

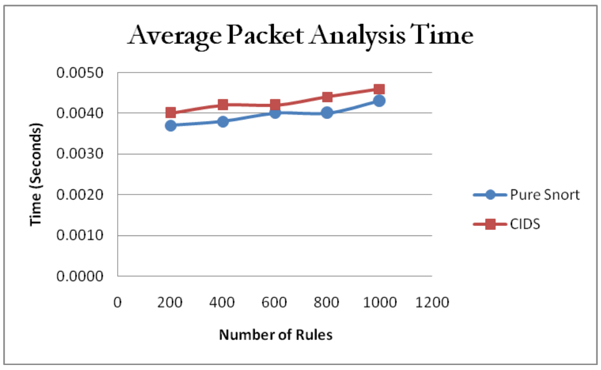

**Figure 5 – Average Packet Analysis Time**

The result in this case is due to the fact that pattern matching is very efficient and that for the two cases we use the same number of signatures.

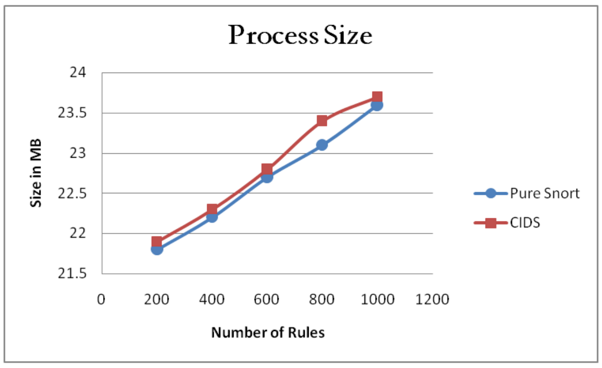

**Figure 6 – Memory consumption rates**

Next, we test for memory consumption rates for the two cases in scenario 1. The CIDS consumes very little extra memory compared to the single use case. This is because all subscribers share the same resources except for the content of each signature. The CIDS framework poses very small overhead over the traditional implementations for single-network protection. This is advantageous since very small overhead is realized when can include the protection to more networks.

## 5.2 Second Scenario

This scenario aims at measuring CIDS advantage in saving the utilization of computing resources. The scenario consists of comparing memory consumption for the case of CIDS with the case of using separate profiles for each subscriber. This is similar to allocating separate IDS for each subscriber. In this scenario we vary the number of subscribers from 100 to 500 subscribers.

Figure7 displays the comparison in memory consumption rates for the two cases. The figure illustrates clear advantage of using CIDS. Cloud providers who designate separate IDS processes or Virtual IDS Hosts experience tremendous overhead.

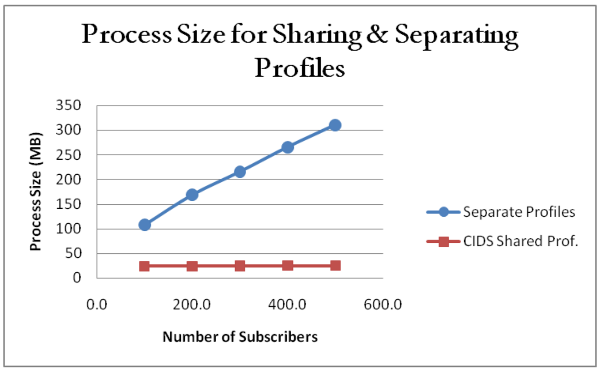

**Figure 7 - Memory consumption rates**

CIDS gets advantage since the same process can be used to analyze the entire traffic. This is where the advantage appears. Using separate IDS requires larger resource allocation for both the ID process and the data while the CIDS utilize a single ID process.

Next we compare the attack detection rates. Figure 8 displays the results for this parameter. The CIDS framework can detect attacks better than the separate profile architecture. This is also due to the fact that the IDS process consumes fewer resources and requires fewer signatures to match against.

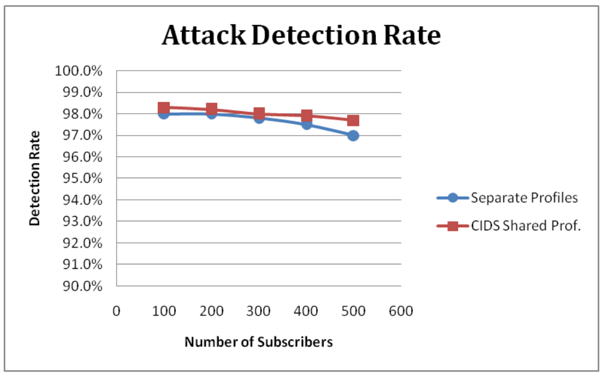

**Figure8 – Attack detection rates**

The utilization of larger resources causes more buffer utilization for matching against larger number of signatures. This in turn causes some of the packets to be dropped before they can even be processed, which causes the CIDS to outperform in detection rates.

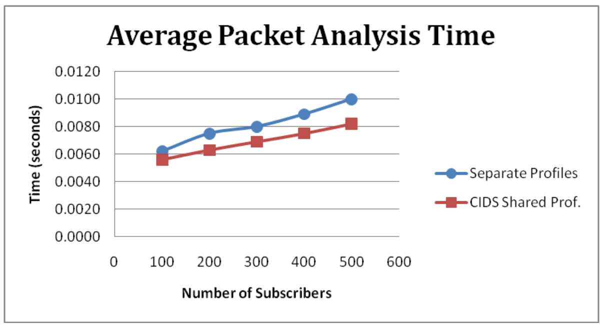

**Figure 9 – Average packet analysis time**

Finally, we test for the average packet analysis time. These results are displayed in Figure 9. As earlier explained, this metric measures the speed of the IDS Architecture in detecting attacks. As proved in the results, the shared scheme of the CIDS framework performs better than the separate scheme of other implementations. This is due to the fact that the IDS process has smaller number of signatures to match the traffic against, which leads to faster full analysis of the network packets and hence faster detection speed.

## 6. Conclusion

In this paper, we have designed and implemented the Cloud Intrusion Detection Service (CIDS) which has been proved to be a very effective solution to the problem of providing intrusion detection as a service in cloud environments. The system which consisted of three separate layers (User Layer, System Layer, and Database Layer), aims to be a scalable intrusion detection framework that can be deployed by cloud providers to enable clients to subscribe with the intrusion detection in a service-based manner. The system is a reengineering of the existing intrusion detection system (snort) but can be implemented with other systems if required. The model outperforms currently used solutions for service-based IDS but at the same time provides minimal overhead to the case of traditional IDS deployment for single network protection.

## 7. References

[1] I. Foster, Yong Zhao, I. Raicu, and S. Lu, "Cloud Comuting and Grid Computing 360-Degree Compared," in Grid Computing Environments Workshop, Austin, Texas, 2008, pp. 1 - 10.

[2] F. Liu, W. Guo, Z. Q. Zhao, and W. Chou, "SaaS Integration for Software Cloud," in IEEE 3rd International Conference on Cloud Computing, Miami, FL, 2010, p. 402.

[3] McAfee Security. Security as a Service. [Online]. http://www.mcafee.com/us/small/security_insights/security_as_a_service.html

[4] HackerTarget.com. (2008, April) Security from the Cloud: Remote Vulnerability Scanning. Whitepaper.

[5] K. Balakrishnan, S. Roy, and M. Holt. (2009, April) Email and Web Security SaaS. Whitepaper.

[6] Panda Security. (2009) Switching from Anti-Virus to Security as a Service (SaaS). Whitepaper.

[7] A. V. Dastjerdi, K. Abu Bakar, and S. Tabatabaei, "Distributed Intrusion Detection in Clouds Using Mobile Agents," in Third International Conference on Advanced Engineering Computing and Applications in Sciences, 2009, pp. 175-180.

[8] P.Kannadiga and M.Zulkernine, "Distributed Intrusion Detection System Using Mobile Agents," in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005.

[9] A. Bakshi and Yogesh B, "Securing cloud from DDOS Attacks using Intrusion Detection System in Virtual Machine," in Second International Conference on Communication Software and Networks, Singapore, 2010, pp. 260-264.

[10] C. Mazzariello, R. Bifulco, and R. Canonico, "Integrating a Network IDS into an Open Source Cloud Computing Environment," in Sixth International Conference on Information Assurance and Security, Atlanta, 2010, pp. 265-270.

[11] Ch. Lo, Ch. Huang, and J. Ku, "A Cooperative Intrusion Detection System Framework for Cloud Computing Networks," in 39th International Conference on Parallel Processing Workshops, 2010, pp. 280-284.

[12] Wikipedia. Denial-of-service attack. [Online]. http://en.wikipedia.org/wiki/Denial-of-service_attack

[13] D.J. Ragsdale, C.A. Carver, Jr. J.W. Humphries, and U.W. Pooch, "Adaptation Techniques for Intrusion Detection and Intrusion Response Systems," Computer Networks, vol. 4, pp. 2344-2349.

[14] E. H. Spafford and D. Zamboni, "Intrusion Detection Using Autonomous Agent," Computer Networks, vol. 34, no. 4, pp. 547-570, 2000.

[15] S. Roschke, F. Cheng, and Ch. Meinel, "Intrusion Detection in the Cloud," in Eighth IEEE International Conference on Dependable, Autonomic, and Secure Computing, 2009, pp. 729-734.

[16] M. Laureano, C. Maziero, and E. Jamhour, "Protecting Hostbased Intrusion Detectors through Virtual Machines," International Journal of Computer and Telecommunications Networking, vol. 51, no. 5, pp. 1275-1283, April 2007.

[17] Y. Weinsberg, S. Tzur-David, D. Dolev, and T. Anker, "High performance string matching algorithm for a network intrusion prevention system (NIPS)," in High Performance Switching and Routing, Poznan, 2006, p. 7 pp.

[18] Oracle America. Oracle Corp. [Online]. http://www.oracle.com/technetwork/java/javaee/overview/index.html

[19] Sun Microsystems. Sun Developer Network (SDN)). [Online]. http://java.sun.com/products/jsp/

[20] Canonical Ltd. Ubuntu Linux. [Online]. http://www.ubuntu.com/