IDSaaS: Intrusion Detection System as a Service in Public Clouds

Turki Alharkan
School of Computing
Queen's University
Kingston, ON Canada
alharkan@cs.queensu.ca

Patrick Martin
School of Computing
Queen's University
Kingston, ON Canada
martin@cs.queensu.ca

***Abstract* - In a public cloud computing environment, consumers cannot always just depend on the cloud provider's security infrastructure. They may need to monitor and protect their virtual existence by implementing their own intrusion detection capabilities along with other security technologies within the cloud fabric. Intrusion Detection as a Service (IDSaaS) targets security of the infrastructure level for a public cloud (IaaS) by providing intrusion detection technology that is highly elastic, portable and fully controlled by the cloud consumer. A prototype of IDSaaS is described.**

Keywords-Security; Cloud Computing; Intrusion Detection System

I. INTRODUCTION

As the number of cyber attacks against social networks and large internet enterprises continues to rise, organizations are questioning the safety of moving their computational assets toward the cloud [1]. Traditional network security measurements face new challenges in the cloud such as virtual machine intrusion attacks and malicious user activities. New security measures are therefore needed to increase users' level of trust in clouds. Currently, cloud providers enforce data encryption for the storage containers, virtual firewalls and access control lists [2]. However, cloud consumers need to develop secure and customizable solutions to comply with their application requirements. For example, an attack classified as SQL injection with the ability to control the host operating system targeting the business application may wish to impose a combination of application and system level policies [3]. The current security mechanisms from the cloud providers are not intended to enforce this level of constraints so additional measurements are required.

In this paper, we propose the Intrusion Detection System as a Service (IDSaaS) framework, which is a network and signature based IDS for the cloud model. In particular, IDSaaS is an on-demand, portable, controllable by the cloud consumer and available through the pay-per-use cost model. IDSaaS mainly targeting the IaaS level of the cloud. However, other levels of the cloud can be monitored such as the SaaS level. Therefore, the IDSaaS primary task is to monitor and log suspicious network activities between virtual machines within a pre-defined virtual network in public clouds. A proof-of-concept prototype for the Amazon EC2 cloud [4] is presented.

The major contribution for this work is a scalable and customizable cloud-based service that provides cloud consumers with IDS capabilities regardless of the cloud model. IDSaaS administrators have the abilities to monitor and react to attacks on multiple VMs residing within a consumer's Virtual Private Cloud (VPC) [5], and to identify specific attacking scenarios based on their application needs. Moreover, the system can adapt its performance to the traffic load by activating the on-demand elasticity feature. For example, the number of the available IDS Core components can change based on the amount of traffic targeting the protected business application. Furthermore, IDSaaS components can be scaled to protect virtual machines residing in different cloud regions. These features are designed with the consideration of the cloud environment.

The rest of this paper is organized as follows: Section II describes related work. Section III introduces the concept of IDSaaS and outlines its main features. Section IV reviews the IDSaaS main components and tools. A proof-of-concept prototype implementation of IDSaaS in Amazon's EC2 public cloud is presented in Section V. Section VI presents a sample attack scenario and evaluates the operation of the prototype IDSaaS. Finally, Section VII summarizes the paper and discusses future work.

II. RELATED WORK

The introduction of IDS in the cloud is the focus of several research projects. Each of these projects, however, targets different service models of the cloud or pursues a different goal. IDSaaS is intended to fill the gap in this research area.

The Intrusion Detection based on Cloud Computing (IDCC) architecture [6] was developed to achieve a global monitoring view of the network resources and to help in discovering coordinated attacks on local sites. This architecture consists of two major parts, the local sites and the global site. The purpose of the global site is to collects the alerts generated by the local sites. When a threat is detected by the global site, the particular local site security administrator is informed so a proper action can be taken such as blacklisting the source of the attack. This architecture is more suitable for private clouds that are designed with the needed infrastructure to allow global and local site nodes to be communicated privately. As a result, cloud users at the local sites are more dependent on the cloud provider's global IDS administration. Furthermore, the process of

 ISBN: 978-1-61208-216-5

administrating the global and local sites raises some serous security challenges.

The work by Mazzariello et al. [7] discusses various deployments of existing IDS to an open source cloud environment. The suggested model is to deploy multiple IDSs next to every cloud physical controller, which monitors a smaller portion of network traffic for a set of virtual machines. The general setup for this approach requires deep alteration of the physical implementation of the cloud assets, which results in a strong dependency between the IDS components and the cloud provider's infrastructure. Consequently, the IDS administration process available to the cloud consumers is limited and lacks customization.

The authors of Intrusion Detection In the cloud (IDC) [8] introduce the concept of a partial IDS management for the cloud users. The proposed architecture consists of several sensors and a central management unit. This distributed-IDS architecture is implemented in all of the three cloud computing layers (Application layer, Platform layer and System layer), which includes a combination of host-based IDS (HIDS) and network-based IDS (NIDS) sensors. HIDS is incorporated with every VM initialized by the user. On the other hand, NIDS sensors are placed in each cloud layer to monitor the management module of that layer. In the central IDS management unit, alerts can be correlated and analyzed from different sensors in different layers. Furthermore, cloud users can configure which rules to use from the existing rule-set based on their application needs. One of the main issues with the IDC approach is the strong dependency between the cloud users and the cloud provider substructure. The cloud provider has to implement the main components of the intrusion detection environment like the central management unit, the integrated HIDS for each VM host, signature databases, and the communication channels between VMs and the IDS management unit. Cloud users are totally dependent on the provider's IDS infrastructure but they still partially control the IDS management unit with limited functionality. Moreover, there are serious privacy concerns arising from integrating IDS components on every customer virtual machine that is installed by the cloud provider.

Much of the proposed academic research on IDSs in the cloud has focused on providing intrusion detection mechanisms for specific security problems. The Autonomic Violation Prevention System (AVPS) [9] concentrates on self-protection against security policy violations generated by privileged users. This goal is achieved by defining the system's access policies and continuously monitoring the internal traffic for any violations of these policies. The authors of the AVPS framework suggest their system can be deployed to virtual network environments like the cloud. However, AVPS is not evaluated against many cloud features. For example, scaling the system for multiple core IDS nodes is needed to bear the increase in the traffic due to heavy application requests. Moreover, the need to support the distributed nature of the cloud by protecting multiple applications in different cloud locations is absent. Additionally, the work in [10] introduces a maneuvering tactic to confront the denial of service attack on the cloud by moving the attacked virtual machine to a safe datacenter.

Our main aim is to provide a general defense strategy by protecting different levels of the cloud, and incorporating tailored signatures for various security threats. IDSaaS is intended to work in different cloud models and to provide flexible user control of security.

III. IDSAAS IN THE CLOUD

A. Overview

Cloud consumers should not have to only depend on the cloud provider's security infrastructure. They need to be able to monitor and protect their virtual existence by enforcing additional security methods with other network security technologies like firewalls, access control lists and data encryption within the cloud fabric. Consequently, cloud consumers require the capability to deploy IDSs within their virtual boundaries.

IDSaaS, which is shown in Figure 1, assists cloud consumers with securing their virtual machines by deploying an intrusion detection system in public clouds. It protects them against attacks initiated from any external source over the internet in addition to those originating from inside the cloud. Here, cloud consumers implement the applications they want to protect in the form of Virtual Machine Instances (VMI) within a secure virtual network (V-LAN). Concurrently, IDSaaS components can be placed in the same V-LAN to guard these valuable assets.

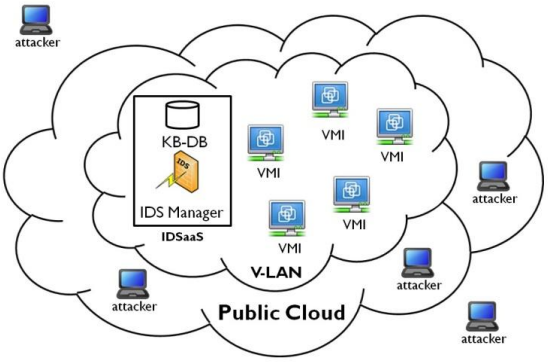

Figure 1. IDSaaS in the Cloud

B. IDSaaS Features

IDSaaS provides the following features to cloud consumers:

- *On-demand Elasticity:* Cloud consumers have the ability to scale IDSaaS core components that are responsible for discovering suspicious traffic based on the traffic volume for the protected business application.
- *Portability:* The IDSaaS model is implemented as a collection of Virtual Machine (VM) instances based on Xen virtualization [11]. Therefore, IDSaaS components can reside in public or private clouds or even in multiple regions within a single cloud.

- *Full-Control:* IDSaaS management, functionality and architecture are independent from the cloud provider. For example, an IDSaaS administrator can deactivate a particular IDS core node or enable a specific threat signature definition.
- *Customizable Signatures:* IDSaaS is equipped with predefined threat scenarios for faster and more accurate detection rates. These scenarios are represented in the form of rules. In addition, IDSaaS users can write customized signatures based on the nature of the defended application. These rules can protect the application (SaaS), the system (PaaS), and the network (IaaS) levels of the cloud model. The grammar and examples of threat signatures are given in Table I.

TABLE I. Signature Examples

	Signature
Grammar	*[action] [protocol] [src_ip] [src_port] [direction] [dist_ip] [dist_port] [option]*
Application Level	alert tcp $EXTERNAL_NET any -> $HOME_NET 80 (uricontent:".php";pcre:"/(\%27)|(\')|(\-\-)|(%23)|(#)/"; msg:"SQL Injection Attack"; sid:1000005;rev:1;)
System Level	activate tcp any any -> $HOME_NET 22 (content:"/bin/sh"; activates:1; msg:"Possible SSH buffer overflow"; sid:1000023;) dynamic tcp any any -> HOME_NET 22 (activated_by:1; count:10;)
Network Level	drop tcp $EXTERNAL_NET any -> $DB_NET 3306 (msg:"Unauthorized Access to DB server"; content:"mysql -p"; nocase; sid:1000019;)

- *Reliability:* IDSaaS has the ability to backup the collected alerts with system configuration files and store them in an off-cloud location. This facilitates an efficient system recovery in the case of failure.

IV. IDSaaS Architecture

IDSaaS, as shown in Figure 2, consists of five main components: the Intrusion Engine, the Output Processor, the Events Database, the Alerts Management, and the Rule-set Manager.

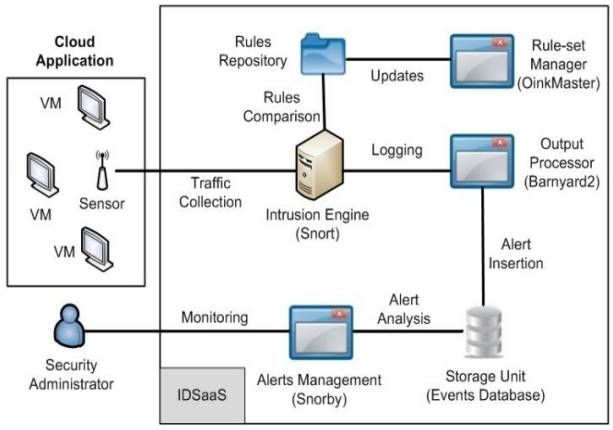

Figure 2. IDSaaS Components

A. Intrusion Engine:

Initially, the sensor taps into the network and collects network packets, which are decoded for the analysis step. The Intrusion Engine is the brain of the system. It preprocesses the incoming packets and examines their payload section looking for any matching pattern of a threat defined in the loaded attacking rules. The processed packet is logged only if it matches a rule. The output binary file is a collection of captured alerts. The signature-based detection model is selected because of its suitability to the cloud environment. Simplicity, flexibility and ease of sharing signatures are some of the advantages of this approach. Also, it will enforce the elasticity feature by eliminating the learning time for the system's behavior required for the anomaly-based approach.

B. Output Processor

The main purpose of the Output Processor is to increase the performance of the intrusion engine by formatting the output log files and inserting them into the Events Database. This allows the intrusion engine to focus on processing network packets and logging alerts while leaving the relatively slow process of database insertion to the Output Processor component.

C. Events Database

The Events Database stores the formatted events generated from the Output Processer component. Also, the database stores other relative information like sensor ID, event timestamp and packet payload details.

D. Alert Management

The Alert Management component is used as a GUI tool to view the generated alerts and correlate them. It allows the security administrator to extract events and relate them to predefined attacking situations. Moreover, it provides the ability to generate reports based on time, source of the attack, or types of threat.

E. Rule-set Manager

IDSaaS is a rule-based IDS system, and its rule base has to be frequently updated to reflect the new threats and attacking scenarios. The Rule-set Manager automatically downloads the most up-to-date set of rules from multiple locations. Rules are generally obtained either for free from the public community service or through a subscription service such as the SourceFire VRT [12].

V. PROOF-OF-CONCEPT IDSaaS

A proof-of-concept prototype of IDSaaS is implemented in Amazon web services using the EC2 cloud. Although it is tested on a public cloud, the IDSaaS framework can be applied to other types of clouds that support V-LAN implementation. All IDSaaS components are constructed and bundled in the form of Amazon Machine Images (AMI). The on-demand elasticity feature of IDSaaS can therefore be enforced by starting the AMI instances on the fly.

 ISBN: 978-1-61208-216-5

A. Tools

The prototype of IDSaaS is built using a collection of open source tools. Snort [13] is an open source network-based intrusion detection system that is used in the Intrusion Engine component. As for the Output Process component, Barnyard2 [14] is used to act as the middle tier between the Intrusion Engine and the Event Database. MySql is used as the relational database to store the generated alerts. Snorby [15] is used as a graphical interface for the system to display various information and statistics about the collected incidents. The Rule-set Manager is built using Oinkmaster [16], which is a simple Perl script that compares locally stored rules with the shared communities' rules repository and downloads updated rules based on user preferences.

B. Network Environment

The IDSaaS utilizes the Virtual Private Cloud (VPC) service from Amazon. This V-LAN setup has the advantage of creating a private network area that can only be controlled by the application owner within the public cloud borders. In the VPC space, both private and public subnets were created. The private subnet maintains the protected business application VMs. Any virtual machine that is placed in the private subnet is isolated from the cloud traffic except the traffic traveling from or to the public subnet of the VPC. The public subnet hosts various IDSaaS VMs. Figure 3 illustrates the general layout of the IDSaaS in the Amazon VPC.

C. IDSaaS VMs

1) IDSaaS Manager

The Manager VM is the security administrator access point where various supervision tasks can be performed. For instance, it hosts the Alert Management component that monitors traffic for any suspicious activity in the VPC. The Event Database also resides in the Manager VM. The Manager VM can be used as an access point to configure other VMs in both public and private subnets.

2) IDS Core

The IDS Core VM is the gatekeeper to the business application VMs in the private subnet. It inspects all incoming traffic using the Intrusion Engine component. Based on the threat rule matching process, a request to the business application VMs can be allowed or trapped by the IDS Core VM. As a result, the Output Processor will send generated alerts to the Event Database.

D. Security Groups (SG)

Security Groups are used to define permissible network services that can run on each VM in the Amazon EC2 cloud. These virtual firewalls can decide the nature of the traffic permitted for each VM in the form of inbound and outbound allowable ports. Any VM that is attached to a particular SG will comply with the services defined for that SG.

VI. IDSaaS Evaluation

We conducted several experiments to evaluate the effectiveness of our proof-of-concept prototype of IDSaaS in EC2. We first present an attacking scenario and then show the results of the experiments.

A. Attacking Scenario

In the scenario, a business application that consists of web and database servers are placed into a private subnet of the Amazon EC2 cloud in order to be accessed by the end users via the IDS Core VM. On the other hand, the IDSaaS VMs are placed on the public subnet. Figure 3 demonstrates the network setup and the deployment of IDSaaS components.

The business application can be accessed using the exposed URL or IP address assigned to the IDS Core VM. Experiments were conducted with different IDSaaS network setups. Each setup experienced two attacking locations; an External Attacker located outside the cloud and an Internal Attacker located inside the Amazon EC2 Cloud. Each attacker used two TCP protocols to attack the victim system. First, they issued a series of HTTP requests to access the registered users' information page of the business application. This area is restricted to the application administrator, so an alert is released for non-authorized access to this area. Second, they used the FTP protocol to upload a suspicious file to the target server through the file transfer service of the business application. Customized rules were enabled in the IDSaaS to capture such a harmful activity for each attacking type.

B. IDSaaS Components Overhead Experiment

The effectiveness of IDSaaS was evaluated by measuring the overhead added by the different IDSaaS components while protecting business applications in a public cloud. By providing an extra level of protection, IDSaaS improves the security element of the virtual machines on the Amazon cloud. Our results so far indicate acceptable increases in the response time for the business application after adding the IDSaaS components (Figure 4). For example, in the case of the FTP requests, IDSaaS imposes 10.60% and 9.27% increases in response time for traffic originating from outside and inside the cloud, respectively. Similarly, for HTTP requests, it imposes increases of 8.58% and 3.57% for traffic originating from outside and inside the cloud, respectively. We believe this size of increase in response time is justifiable given the additional ability to enforce tailor-made attacking rules. Table II shows the average response time for all network setups of the experiment.

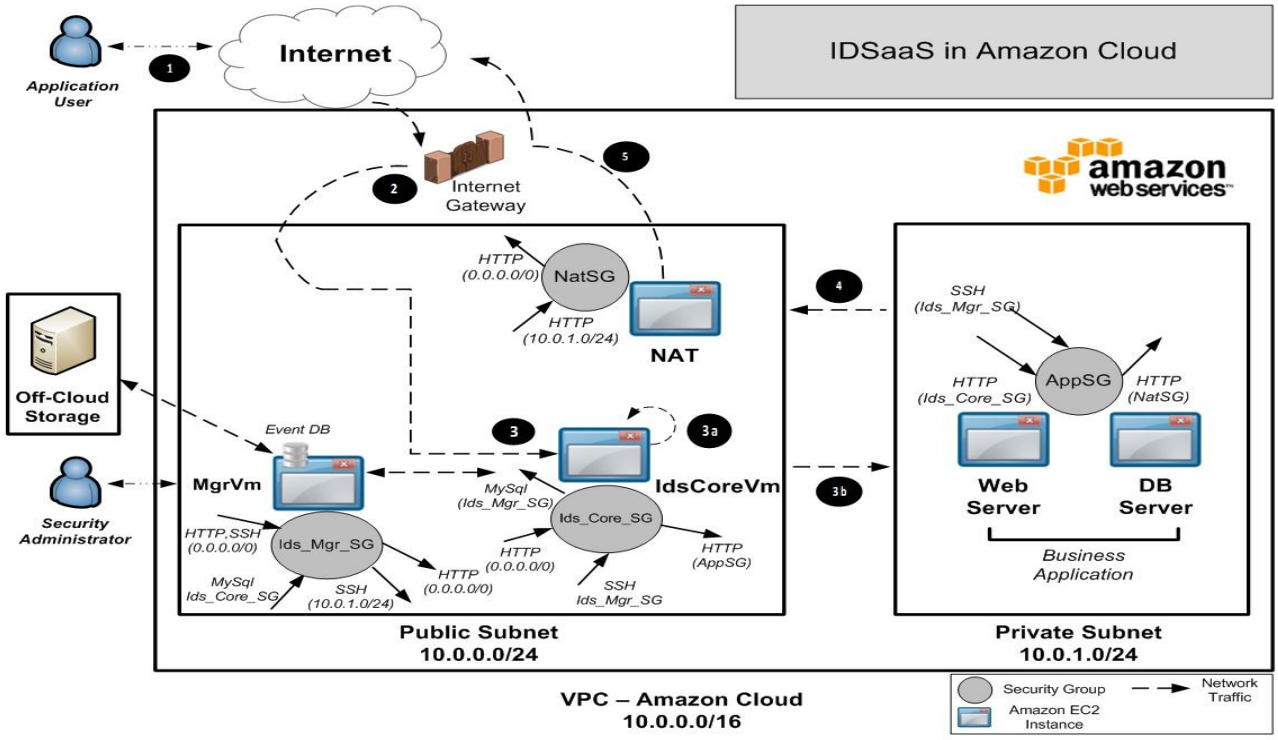

Figure 3. IDSaaS in Amazon Cloud

TABLE II. Components Overhead Experiment results

Network Setup		External Location	Internal Location
Base Network (No VPC, No IDSaaS)	HTTP Request	0.303 sec	0.224 sec
	FTP Request	16.731 sec	3.969 sec
VPC Network (No IDSaaS)	HTTP Request	0.310 sec	0.230 sec
	FTP Request	17.040 sec	4.098 sec
IDSaaS Network (VPC, IDSaaS)	HTTP Rule	0.329 sec	0.232 sec
	FTP Rule	18.505 sec	4.337 sec

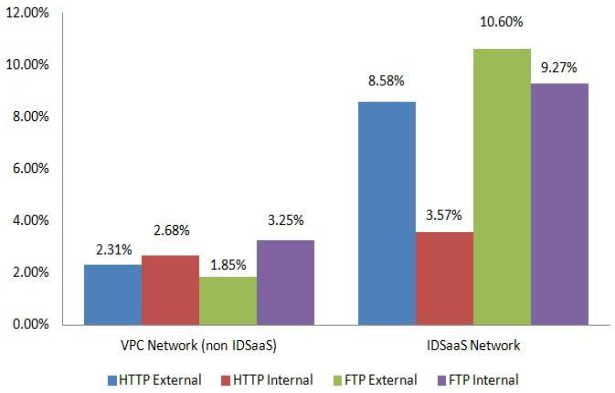

Figure 4. IDSaaS Components Overhead

C. IDSaaS Rules Overhead Experiment

The number of loaded attacking rules can also affect the efficiency of the IDSaaS in capturing many threats. In this experiment, we observed the performance of the intrusion engine (IDS Core VM) against different rule set stages. Stage one includes a complete set of rules (18,833 rules) addressing different attacking situations. This rule-set contains a collection of the intrusion engine's preinstalled rules as well as rules obtained from the public communities like Emerging Threats team [17]. Stage two decreases the rule set to 11 rules, which represents the Attack-Response (A-R) rules. Finally, the last stage incorporates a single rule to detect the Automatic Directory Listing (ADL) attack.

The IDS Core VM is used to compare the rules from the rule repository with the captured traffic in the form of a pcap file [18]. Intrusion engine performance was defined as the run time to process incoming packets, compare them with enabled rules and produce alerts in the form of binary logs. Therefore, the smaller the run time to analyze traffic packets, the better the performance of the intrusion engine.

Figure 5. IDSaaS Rule Overhead Experiment Results

Figure 5 shows that intrusion engine produced a single alert in an average time of 24.77 seconds by enabling the ADL rule (Stage 3). On the other hand, the intrusion engine took an average of 28.46 seconds to discover 68 threats by enabling the A-R subset. As a result, there was a 14.90% increase in the overhead for the extra rules enabled between stage 3 and stage 2. Similarly, stage 1 managed to capture 10,504 threats from the data sample within an average time of 48.72 seconds. This can be translated into an increase of 71.19% of the overhead compared to stage 2. For that reason, enabling all rules will degrade the performance of the intrusion engine and it will increase the percentage of false-positive alerts. Hence, fine-tuning the intrusion engine component to reflect the nature of the protected application is an important step when dealing with large number of rules.

D. Distributed IDSaaS Experiment

We implemented a distributed version of IDSaaS (D-IDSaaS) that has the ability to protect application VMs residing in multiple cloud regions. This is achieved by placing one or more IDS Core VMs in the same VPC as the business application VMs and placing the Manager VM in a centralized location. The security administrator can therefore monitor multiple business applications in different regions of the cloud from the central Manager VM.

We examined the cost of sending alerts from the IDS Core VM to the Manager component in three network configurations. Configuration 1, places the IDS Core VM and the Manager VM in the same VPC of the same cloud region. This typical IDSaaS setup is illustrated previously in Figure 3. Configuration 2, positions the Manager VM in a corporate network outside the cloud to meet with the privacy concerns of storing alerts in the cloud as well as reducing the storage costs of archiving historical alerts. In configuration 3, the IDS Core VM and the Manager VM are placed in different regions of the cloud. The business applications and the IDS Core VM are placed in the EU region of Amazon cloud and the Manager VM is positioned in the US East region of the Amazon cloud. Figure 6 displays the network layout for configuration 2 and 3.

The intrusion engine component was configured to read from a single pre-captured traffic file rather than from live network traffic. This standardized the input traffic to be analyzed by the intrusion engine for all network layouts. The used pcap file contained 30,000 network packets, which generates 145 alerts when enabling all installed rules (18,833 rules). Both the IDS Core VM and Manager VM were initialized using the small EC2 instances (OS Ubuntu, 1.7 GB memory, 1 virtual core CPU and 160 GB storage). However, the Manager VM in the off-cloud network (OS Ubuntu, 1.7 GB memory, 1 virtual CPU, 20 GB storage) was initialized using the VMware software [19] as a guest operating system.

The average dispatching time for 145 alerts from the pcap file using 100 trails was 2.35 seconds in configuration 1. In configuration 2, the same number of alerts was received on an average of 30.94 seconds. However, the highest dispatching time was obtained from configuration 3 with 119.70 seconds. The results are demonstrated in Figure 7. We believe this high value is due to alerts transmission time between the two Amazon regions.

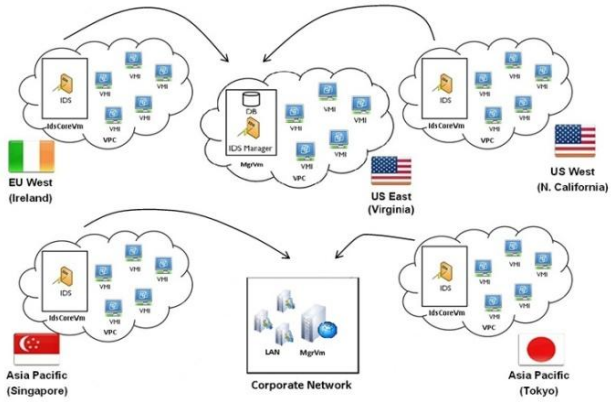

Figure 6. Distributed IDSaaS in Public Cloud

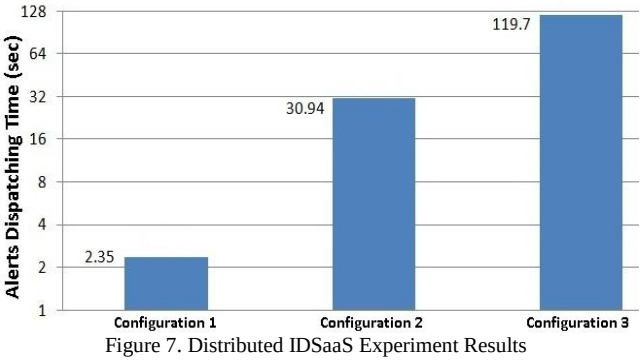

Figure 7. Distributed IDSaaS Experiment Results

VII. SUMMARY

In this paper, we introduced IDSaaS, which is a framework that enables consumers to protect their virtual

machines in public clouds. IDSaaS is compatible with many cloud features, such as portability, elasticity, on demand requests and pay-per-use service. The approach presented in this paper is implemented as a collection of virtual machines in order to comply with the cloud model.

Cloud consumers need to have customizable and controllable security solutions in the clouds. The major contribution for this work is a service to provide them with IDS capabilities regardless of the cloud model. With IDSaaS, users can define a virtual private area within the cloud space for their applications that can be secured with application-specific policies. Therefore, IDSaaS adds new levels of security onto those already supplied by cloud providers.

Increasing system availability is a feature to be implemented for future IDSaaS system. A replica of the IDS Core VM can be created to distribute the traffic load to prevent single point of failure situations. Therefore, a virtual load balancer node can increase the accessibility of the IDSaaS components in the cloud. Also, it can be responsible for balancing the traffic load between multiple IDS Core VMs.

ACKNOWLEDGMENT

This research is supported by the Ministry of Higher Education in the Kingdom of Saudi Arabia, the Saudi Arabian Cultural Bureau in Canada and Queen's University.

REFERENCES

[1] C. Burns, "Public cloud security remains MISSION IMPOSSIBLE," Network World. Oct, 2011. [Online]. Available: http://www.networkworld.com/supp/2011/enterprise5/101011-ecs-cloud-security-250973.html, [Retrieved: June, 2012].

[2] Amazon Web Services: Overview of Security Processes. [Online]. Available: http://aws.amazon.com/security, [Retrieved: June, 2012].

[3] B. Damele and A. Guimaraes, "Advanced SQL injection to operating system full control", Black Hat Europe 2009, April 2009

[4] Amazon Elastic Compute Cloud (Amazon EC2). [Online]. Available: http://aws.amazon.com/ec2/, [Retrieved: June, 2012].

[5] Amazon Virtual Private Cloud (Amazon VPC). [Online]. Available: http://aws.amazon.com/vpc, [Retrieved: June, 2012].

[6] W. Xin, H. Ting-lei, and L. Xiao-yu, "Research on the Intrusion detection mechanism based on cloud computing," Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference pp.125-128, 22-24 Oct. 2010

[7] C. Mazzariello, R. Bifulco, and R. Canonic, "Integrating a Network IDS into an Open Source Cloud Computing Environment," Sixth Internationl Conference on Information Assurance and Security (IAS), 2010

[8] S. Roschke, F. Cheng, and C. Meinel, "Intrusion Detection in the Cloud", In Proceedings of Workshop Security in Cloud Computing (SCC'09), IEEE Press, Chengdu, China, pp. 729-734 (December 2009).

[9] F. Sibai and D. Menasce, "Defeating the Insider Threat via Autonomic Network Capabilities," Communication Systems and Networks (COMSNETS), 2011 Third International Conference pp. 1-10, 4-8 Jan. 2011

[10] A. Bakshi and Y. Dujodwala, "Securing Cloud from DDOS Attacks Using Intrusion Detection System in Virtual Machine," ICCSN '10, pp. 260-264, 2010, IEEE Computer Society, USA, 2010

[11] Feature Guide: Amazon EC2 User Selectable Kernels. [Online]. Available: http://aws.amazon.com/articles/1345, [Retrieved: June, 2012].

[12] The official Snort Rule-set, Sourcefire Vulnerability Research Team (VRT). [Online]. Available: http://www.snort.org/vrt, [Retrieved: June, 2012].

[13] Sourcefire, Snort (version 2.9.5). [Online]. Available: http://www.snort.org, [Retrieved: June, 2012].

[14] The Barnyard2 Project. [Online]. Available: http://www.securixlive.com/barnyard2, [Retrieved: June, 2012].

[15] Snorby (version 2.2.6). [Online]. Available: http://www.snorby.org, [Retrieved: June, 2012].

[16] The OinkMaster Project. [Online]. Available: http://oinkmaster.sourceforge.net, [Retrieved: June, 2012].

[17] Snort Rules, Emerging Threats Project. [Online]. Available: http://rules.emergingthreats.net, [Retrieved: June, 2012].

[18] 1999 Training Data - Week 4, DARPA Intrusion Detection Evaluation. [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999/testing/week4/index.html, [Retrieved: June, 2012].

[19] VMware Inc, Wmware Player (version 4.0.2). [Online]. Available: http://www.vmware.com/products/player, [Retrieved: June, 2012].