The Islamic University of Gaza

Graduate Studies Deanship

Faculty of Engineering

Computer Engineering Department

Intrusion Detection Management as a Service in Cloud Computing Environments

Mahmoud Omar Al-Hoby

Supervisor

Prof. Hatem M. Hamad

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Computer Engineering

1432H (2011)

Intrusion Detection Management as a Service in Cloud Computing Environments, Mahmoud Omar Al-Hoby

Abstract

Current implementations and research trends for intrusion detection in grid and cloud environments are limited to addressing the requirements for the perfect intrusion detection to be part of the security infrastructure. This doesn't take into consideration the requirements of the cloud's clients. In this thesis, we address the intrusion detection in cloud environments from a different perspective, mainly on the possibilities to allow intrusion detection to be provided to clients as a service. The thesis includes the limitations in current intrusion detection systems that don't allow such user-friendly architecture of intrusion detection. The thesis describes the Cloud Intrusion Detection Service (CIDS), which is a novel 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. The testing environment was composed of two scenarios. The first scenario aimed at measuring the relative overhead of using CIDS while the second one aims at measuring the CIDS effectiveness and performance improvements over other implementations for approaching the same problem. The CIDS was eventually found to put very small overhead due to the extra complexity in the definitions of the attack models but at the same time gave excellent results when it was compared to the other solutions. This improvement would be experienced by both the cloud providers and subscribers alike.

Keywords: Cloud Computing; CRE, Intrusion Detection, SaaS

إدارة كشف التسلل كخدمة في بيئات الحوسبة السحابية

محمود عمر الهوبي

ملخص الرسالة

تقتصر التطبيقات والأبحاث الحالية في مجال كشف التسلل في الشبكات السحابية على معالجة متطلبات كشف التسلل ليكون جزءاً من بنية أمان الشبكات، وبشكل يعتمد على متطلبات وتخصيصات مدراء هذه الشبكات. وهو الأمر الذي يتجاهل متطلبات مستخدمي هذه الشبكات. في هذا البحث يتم التطرق إلى تمكين استخدام قدرات كشف التسلل من منظور آخر، وبشكل أكثر تحديداً إلى تقديمها كخدمة يمكن لمستخدمي شبكات الحوسبة السحابية اختيارها وتضمينها ضمن الخدمات التي يرغبون بالتسجيل فيها. خلال البحث يتم التطرق إلى نقاط الضعف وأوجه القصور الموجودة في الأنظمة الحالية والتي تمنع التوظيف المباشر لاستخدامها كخدمة. كما يتم في هذة الرسالة وصف وتقديم خدمة كشف التسلل للشبكات السحابية (CIDS) والتي تعتبر خدمة ويب جديدة مقدمة خصيصاً لمستخدمي الشبكات السحابية.

تستخدم برنامج "Snort" وهو نظام كشف التسلل المفتوح المصدر. لتجربة وتقييم أداء الخدمة المقترحة تم استخدام لغة J2EE لتطبيق منطق العمل في النظام وأيضاً صفحات الويب التي يتعامل معها مستخدمة الخدمة. تم تقسيم خيارات التقييم إلى جزءين بحيث يهدف الأول إلى قياس العبء الاضافي الحاصل نتيجة استخدام خدمة (CIDS) نظراً للتعقيد الاضافي في تعريق الأنماط الهجومية، وأنا الثاني فيهدف إلى قياس التحسن الحاصل عند استخدام الخدمة في مقابل استخدام الحلول الأخرى المقترحة في هذا المجال. في نهاية الدراسة تم التوصل إلى أن خدمة (CIDS) تفرض عبئاً صغيراً جداً على الأنظمة العادية الغير مصممة للشبكات السحابية ولكنها في نفس الوقت تُحسن من أداء الخدمة في بيئات الحوسبة السحابية بشكل ملحوظ، كما أن هذا التحسن يشمل كل من مزودي الخدمة ومستخدميها على حدٍ سواء.

Acknowledgement

I would like to express my deepest gratitude for my parents who have always been there to support me. I also thank my wife who has been strongly supportive to me to the end of this thesis.

I am also greatly thankful to my supervisor, Prof. Hatem Hamad, whose encouragement, guidance and support from the initial to the final phase enabled me to develop a deep and thorough understanding of the subject. Finally, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Mahmoud Al-Hoby

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

1.1 Thesis Statement

This thesis 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 configure the intrusion detection (ID) components in a matter similar to configuring it within their own local area networks (LANs). The ideas presented in this thesis are the author's original works. The implementations and results are also accurate and were obtained solely by the author.

1.2 Background

The thesis builds upon the fact that cloud computing is becoming a more and more accepted solution for hosting the information resources of organizations across the globe [**1**], with no physical deployments needed at the clients side. Instead every needed service can be made available as a subscription-based service [**2**]. Intrusion detection as a service is by no means an exception. Typically, organizations that tend to host their own information resources can deploy some sort of an intrusion detection system as part of their network. The IDS infrastructure is hosted, managed, configured, and monitored by the technical staff in the organization itself. Since cloud computing can provide solutions in the form of Infrastructure as a Service (IaaS) [**3**], then a normal requirement would be to also include intrusion detection as part of the infrastructure within the cloud.

As will be discussed with the following chapters, the current models for intrusion detection are not mature enough to allow the flexibility for users to be in full control of the utilized intrusion detection system as well as to not being resource friendly for the cloud provider too.

1.3 Research problem

The current research activities for intrusion detection systems within cloud environments [**4**][**6**] have mostly been concentrating on the IDS's ability to handle the enormous volumes of traffic passing through the core backbone of these shared networks and the structuring of intrusion detection to be part of the security infrastructure of the cloud [**7**]. But few of them discussed the ability to provide intrusion detection as a service for clients in the cloud in limited manner. To be more specific, current implementations of intrusion detection didn't take into consideration the opinions of the cloud's client. This thesis provides the cloud intrusion detection service (CIDS) which is designed to be a service-based intrusion detection system for cloud environments.

1.4 Scope of Implementation

The ideas in this thesis can be applied for any network. The main purpose is for implementation within cloud computing environments to provide easily configurable intrusion detection service for the customers.

One possible scenario is for a cloud provider to implement the system in this thesis and provide it for cloud customers in service-based manner. Customers can then choose to subscribe or unsubscribe from the system. Customers can also choose whatever protection requirements are needed and pay for only the volume of protection they request. Another possible implementation is by LAN manager, where they can deploy it within their localized networks. This would enable them to create protection profiles for each asset they have on site. For both cases, the system described in this thesis is very resource friendly and can improve the overall performance considerably regardless of the volume of network traffic.

In the following chapter, we present a background about the topic which includes a preliminary discussion about Cloud Computing and Intrusion Detection Systems in chapter 2. Then in chapter 3 we take an overview about some of the recently published research papers on the matter of integrating intrusion detection with cloud computing. Within this overview, we discuss the limitations of them that hinder their utilization as intended in this thesis. After that, we describe the Cloud Intrusion Detection Service (CIDS) which is the solution presented by this thesis to address the issue of providing intrusion detection in a service-based manner. Later and in chapter 5, we implement the CIDS and compare the results to the existing solutions in order to verify the performance improvement and goal achievement. The final results and future work are concluded in chapter 6.

Chapter 2 – Preliminary Discussions

In this chapter, we review the basic concepts behind this thesis. Mainly speaking, we will discuss the concepts of Intrusion Detection Systems and Cloud Computing. For each concept, a quick and comprehensive review will be made. This includes the main concepts, the advantages, and the types available that distinguish their usage in practice, and finally, a global view of the current deployments around the world, which will be aided by figures to illustrate this.

2.1 Intrusion Detection Systems

An Intrusion Detection System (Commonly referred to IDS) is a system that replaces the typical task of system administrators of constantly reviewing the log files in attempt to spot any abnormal records. And by abnormal we mean any records that indicate a malicious activity by the user. These malicious activities include a wide variety of actions that usually tend to attack and/or damage the system being the target. This method was enough for monitoring the activities of small group of people within a private organization. But with the expanded usage of computing system and by the development of complex interconnected networks, this is no longer an applicable method. Automated methods were needed to make the task faster and easier [**7**]. This was the first step towards building the intrusion detection systems.

Much has been written about intrusion detection systems recently. In fact, work in IDS field has been in progress for more than 25 years now [**8**]. Generally, IDs can be defined as the tools, methods, and resources that help to identify, assess, and report unauthorized or unapproved network activities. The intrusion detection part of the name is a bit of a misnomer, as an IDS system doesn't actually detect intrusions. It rather detects activities in traffic that may or may not be an intrusion. Intrusion detection is usually part of an overall security architecture that is installed around a system or device [**9**].

Intrusion Detection is getting increased importance as the sophistication of Internet-based attacks is also increasing. Figure 2-1 [**10**] illustrates the increasing level of sophistication of attacks from mid-1980s to early 2000s. The following subsections present the basic intrusion detection models that are currently being used and deployed by various organizations globally.

2.1.1 Protection Level

By protection level, we mean the level at which the IDS can provide protection for. For this categorization, two protection levels exist. The first is the Host-Based Protection (HIDS), and the second is the Network-Based Protection (NIDS).

- *Host-Based IDS*

This is where the intrusion detection system is intended to protect a single host. This is usually achieved using a special software running on the host and utilize fire-wall like strategies to intercept traffic and analyze it to report any malicious traffic.

- *Network-Based IDS*

This type of IDS is usually used to protect a complete network segment. In order for them to be able to do this task, they are typically placed on the network perimeter in a place that allows it to read all the exchanged traffic with the protected network segment.

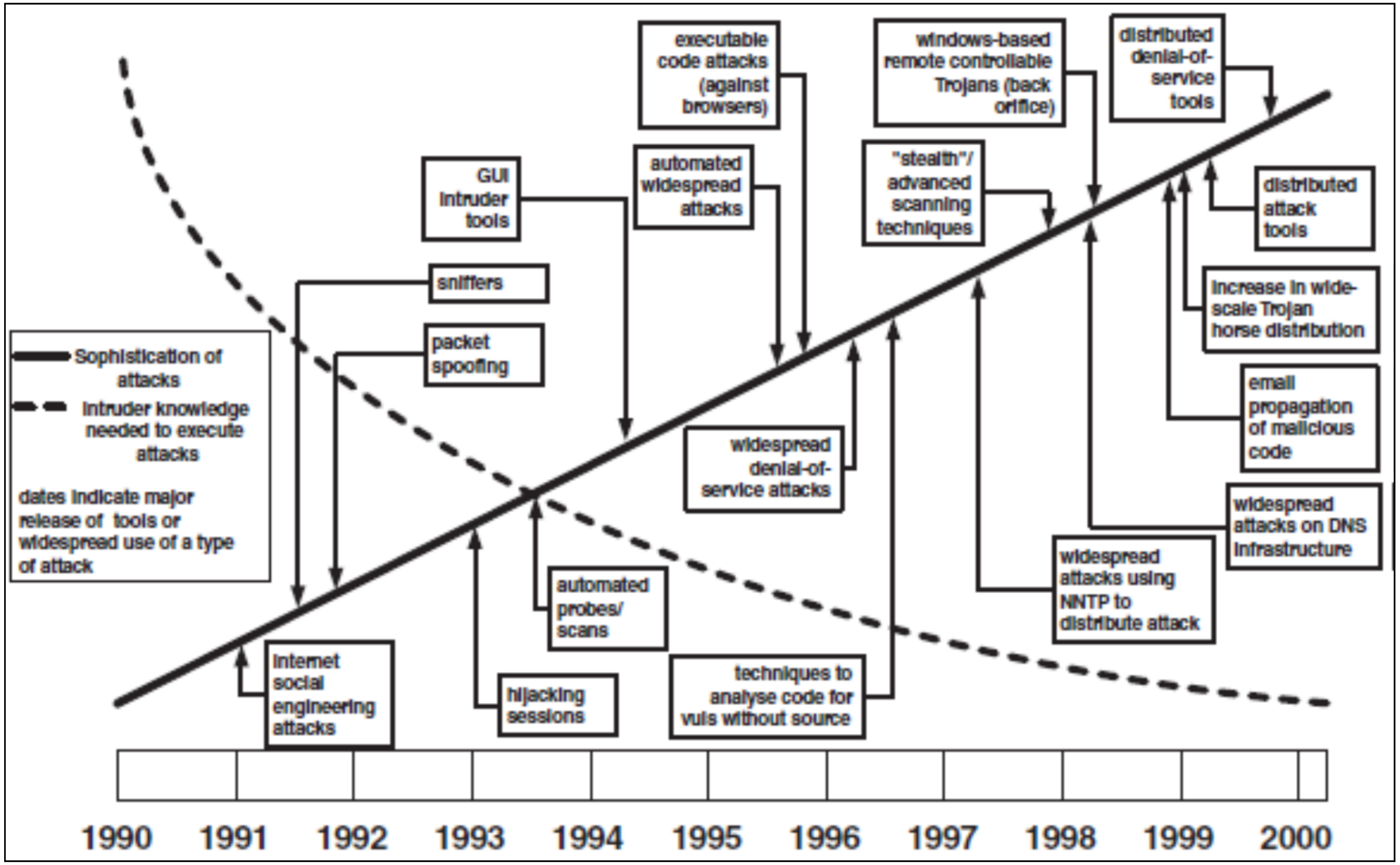

executable code attacks (against browsers)
windows-based remote controllable Trojans (back orifice)
distributed denial-of-service tools
GUI intruder tools
automated widespread attacks
"stealth"/ advanced scanning techniques
distributed attack tools
sniffers
packet spoofing
increase in wide-scale Trojan horse distribution
Sophistication of attacks
Intruder knowledge needed to execute attacks
dates indicate major release of tools or widespread use of a type of attack
email propagation of malicious code
widespread denial-of-service attacks
widespread attacks on DNS infrastructure
automated probes/ scans
widespread attacks using NNTP to distribute attack
Internet social engineering attacks
hijacking sessions
techniques to analyse code for vuls without source
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000

Figure 2- 1: The evolution of attack sophistication and devolution of attacker's skills

2.1.2 Detection Techniques

This defines the techniques and strategies that are utilized for the purpose of identifying intrusions. Generally, there exist two types of these techniques. These include the Anomaly-Based intrusion detection and the Misuse-Based intrusion detection [**12**].

- *Anomaly Detection*

These types of intrusion detection systems attempt to *model normal behavior*. Any events which violate this model are considered to be suspicious. For example, a normally passive public web server attempting to open connections to a large number of addresses may be indicative of worm infection.

The main advantage of anomaly detection systems is that they can detect previously unknown attacks. By defining what's normal, they can identify any violations, whether it is part of the threat model or not. In actual systems however, the advantage of detecting unknown attacks is paid for in terms of high false-positive rates. Anomaly detection is also difficult to train in highly dynamic environments. [**7**]

- *Misuse Detection*

The misuse-based intrusion detection systems attempt to *model the abnormal behavior*, any occurrence of which clearly indicates system abuse. For example, an HTTP request referring to the cmd.exe file may indicate an attack.

The main disadvantage of misuse detection systems is that they can detect only known attack for which they have a predefined signature. These techniques require the modeling and development of new signatures for each newly discovered attack. These signatures must then be added to the published signature database [**7**].

2.2 Architecture of Intrusion Detection Systems

The work of typical IDS is to analyze some input data. The Input data might range from audit trails and operating system or application logs to raw network traffic. An IDS in the basic structure consists of a Sensor, Analyzer, and Event Notifier [**13**]. Figure 2-2 illustrates the canonical model of such structure.

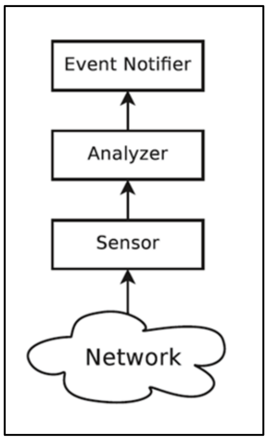


Figure 2- 2: Basic IDS Structure

- **Sensor**: whatever the used input data is, a component is needed which can read such data and convert it to a format which is compatible with the one required from the analyzer. The conversion into such a format sometimes involves the extraction of some parameters of interest aimed at synthesizing the properties of the data which are of greater interest for the problem at hand. In the case of the proposed intrusion detection system, network packets are usually decoded, all the header fields are evaluated, and a set of traffic features are computed, related to some statistical properties of the traffic.

- **Analyzer**: once the data is modeled into a common format, it needs to be analyzed. In principle, the analyzer component could be independent of the type of data. It needs to be aware of a set of criteria aimed at detecting some particular properties in the analyzed data and, when at least one out of such criteria is matched, notify an entity about the occurrence of such an event. If each criteria is associated to the most likely cause which might have generated the event it's related to, the analyzer not only is able to notify in case of the occurrence of some particular events, but is also able to ascribe such events to a generating cause, thus enabling the classification of each reported event.

- **Event Notifier**: any time the analyzer reports the occurrence of some events, it is necessary to enable the whole system to communicate with the external world, in order to allow the notification of such occurrences. The event notifier is in charge of interpreting the results of the analysis and correctly formatting the messages required for communicating with the system users.

2.2.1 "Snort"

"Snort" is a free and open source network intrusion prevention system (NIPS) and network intrusion detection system (NIDS) capable of performing packet logging and real-time traffic analysis on IP networks. "Snort" was written by Martin Roesch [**14**] and is now developed by Sourcefire® [7], of which Roesch is the founder and CTO. Integrated enterprise versions with purpose built hardware and commercial support services are sold by Sourcefire. [**15**]

"Snort" consists of three main components: *Packet Decoder*, *Detection Engine*, and *Logging/Alerting* Subsystem [**16**]:

- **Packet Decoder**, which converts raw network traffic into organized and easily accessible structures that can be later used to reference specific portions of the packets, like source and destination IP addresses and port numbers.

- **Detection Engine**, which takes the decoded packets as input, performs pattern matching for the available signatures and if any export the matches to the alert subsystem. "Snort" maintains its detection rules or signatures in a two dimensional linked list of what are termed Chain Headers and Chain Options.

- **Logging/Alerting Subsystem**, which differ in their intended function. The logging options can be set to log packets in their decoded, human readable format to an IP-based directory structure, or in TCP Dump binary format to a single log file. The Alerting on the other hand enable the documentation of alerts found by positive matches with the predefined attack patterns. Alerts may be sent to SysLog, logged to an alert text file in two different formats, or sent as Win Popup messages using the Samba SMB Client program.

2.3 Cloud Computing

The cloud is not simply the latest fashionable term for the Internet. Though the Internet is a necessary foundation for the cloud, the cloud is something more than the Internet. The cloud is where you go to use technology when you need it, for as long as you need it, and not a minute more. There is no need not install anything on the desktop and only pay for the technology when it is actually used [**3**].

The term 'cloud' first appeared in the early 1990s, referring mainly to large ATM networks. Cloud computing began in earnest at the beginning of this century, just a few years ago with the advent of Amazon's web-based services. Recently, Yahoo and Google announced plans to provide cloud computing services to some of USA's largest universities: Carnegie Mellon, University of Washington, Stanford, and MIT. IBM quickly announced plans to offer cloud computing technologies, followed almost at once by Microsoft. More recent entries into the fray include well known companies: Sun, Intel, Oracle, SAS, and Adobe. All of these companies invested mightily in cloud computing infrastructure to provide vendor-based cloud services to the masses [**2**].

Table 2-1, found in [**3**], demonstrates a comparison between the traditional computing services and the web-based cloud services.

Table 2- 1: The Old IT Infrastructure versus the Cloud

Traditional	**Cloud**
File Server	Google Docs
MS Outlook, Apple Mail	Gmail, Yahoo, MSN
SAP CRM/Oracle CRM/Siebel	SalesForce.com
Quicken/Oracle Financials	Intacct/NetSuite
Microsoft Offce/Lotus Notes	Google Apps
Stellent	Valtira
Off-Site Backups	Amazon S3
Server, Racks, and Firewalls	Amazon EC2, GoGrid, Mosso

Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. And by computing resources we mean any things that can be utilized as part of computing systems, this includes the networks, servers, storage, applications, and services. An important concept that is always being mentioned as an apparent advantage of cloud computing is the availability of resources [**17**].

From a technical viewpoint, the elements of the cloud include processing, network, and storage elements. The cloud architecture consists of three abstract layers: infrastructure, platform, and application. Infrastructure is the lowest layer and is a means of providing processing, storage, networks, and other fundamental computing resources as standardized services over the network. Servers, storage systems, switches, routers, and other systems handle specific types of workloads from batch processing to server-storage augmentation during peak loads [**17**].

Cloud providers' clients can deploy and run operating systems and software for their underlying infrastructures. The middle layer provides higher abstractions and services to develop, test, deploy, host, and maintain applications in the same integrated development environment. This layer provides a runtime environment and middleware to deploy applications using programming languages and tools the cloud provider supports. The application layer is the highest layer and features a complete application offered as a service. Figure 2-3 shows a cloud infrastructure's general layered architecture, with the additional user interface layer, which enables seamless interaction with all the underlying everything-as-a-service layers [**17**]

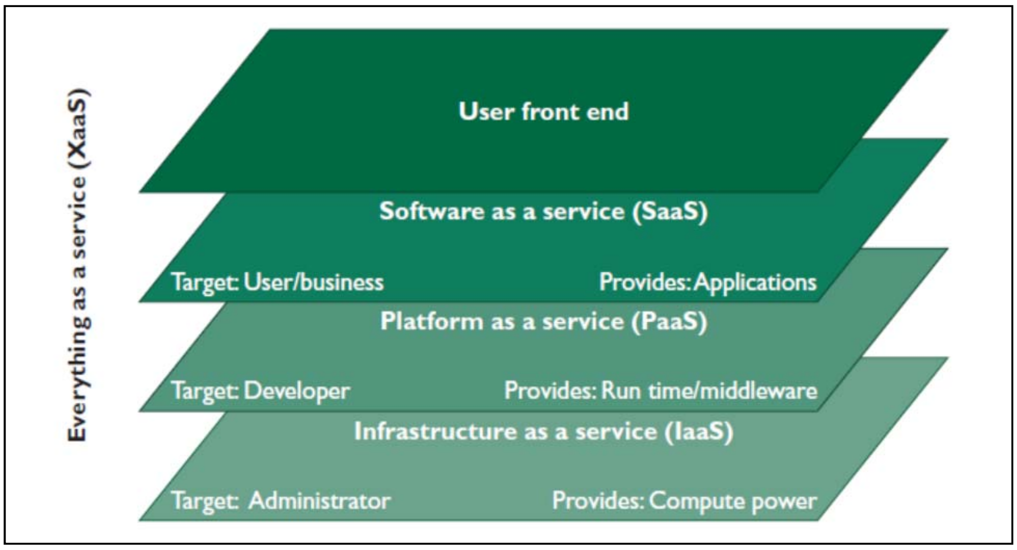

Everything as a service (XaaS)
User front end
Software as a service (SaaS)
Target: User/business
Provides: Applications
Platform as a service (PaaS)
Target: Developer
Provides: Run time/middleware
Infrastructure as a service (IaaS)
Target: Administrator
Provides: Compute power

Figure 2- 3: General Layered Architecture of Cloud Infrastructure

2.3.1 Service Models in Cloud Computing

- Software as a Service (SaaS)

SaaS is a model for which the applications are hosted as services to customers who access it via the Internet. When the software is hosted off-site, the customer doesn't have to maintain it or support it. On the other hand, it is out of the customer's hands when the hosting service decides to change it [**18**].

- Platform as a Service (PaaS)

PaaS on the other hand, delivers the cloud services differently. As the name suggests, PaaS supplies all the resources required to build applications and services completely from the Internet, without having to download or install any kind of software. The services provided in PaaS model include application design, development, testing, deployment, hosting, team collaboration, web service integration, database integration, and versioning [**18**].

However, PaaS lacks the interoperability and portability among different providers. In other words, if an application is created with one cloud provider and then the customer decides to move to another provider, then she may not be able to do so or it may require high prices for the application to run in the new provider's cloud. Google App Engine [**19**] is an example of PaaS clouds where users can create their own applications with either python or Java and deploy it on Google's cloud.

- Infrastructure as a Service (IaaS)

Sometimes referred to as HaaS or Hardware as a Service [**18**], it is considered the next form of services available in cloud computing. Where SaaS and PaaS are providing applications to customers, IaaS doesn't. In the simplest form, IaaS provides the organizations with hardware resources that can be used for anything. The advantage is that instead of buying servers, software, racks, and having to pay for the datacenter space for them, the service provider rents those resources. And by renting resources we mean any resources than a person can think of, including Server Space, Network Equipment, Memory, CPU Cycles, Storage Space, etc… Additionally, the infrastructure resources can be scaled up or down based on the application resource needs.

2.4 Security Services in the Cloud

In a report by Gartner Group [**20**], security services provided in the cloud have the potential to provide cost savings and faster deployment compared with equivalent-capacity, premises-based equipment, but providers are yet to deliver on customer expectations. Currently many traditional security systems are provided as services in the cloud. These systems have been made available to end user to provide the security products for users in a service-based manner. Such model is referred to as Security-as-a-Service model [**21**]. This included many product services and types like Remote Vulnerability Scanning [**22**], Webroot® [**23**] Email and web Security SaaS [**24**], and Panda® [**25**] Managed Office Protection [**26**].

In this thesis, we introduce the usage of intrusion detection systems as services in cloud computing environments. The basic concept is that NIDS are used frequently as a main component in perimeter network security [**27**]. While deploying and configuring NIDS is considered an infrastructure type security measure, IaaS service models still have limited support to offering intrusion detection as services. By limited we mean, that even when cloud subscribers wish to deploy an IDS system in their cloud's network segments, they will need to do this task entirely themselves. An example of this is the usage is Amazon Elastic Compute Cloud (EC2) [**28**] where users can purchase and use Amazon Machine Image (AMI) that comes with "SNORT" IDS on it [**29**]. As we shall see in the next chapter, other proposals have been introduced to enable intrusion detection for the protection of the cloud itself not the cloud's subscribers. And for many, the distinction between the cloud protection and the cloud clients' protection is unclear.

The distinction is based on the classic network and systems security architecture, where the network infrastructure is designed, deployed, supported, and secured by a special staff called network administrators. Their task is to provide a reliable infrastructure which sets scalability, availability, performance protection as the main concerns [**30**]. On the other hand, it is the task of the system developers to design and build their systems in a secure way. The same case exists for cloud computing environments, where cloud providers have some security tasks assigned to them. Kandukuri et. al. [**31**] have proposed a set of requirements to be included in the Service-Level-Agreement (SLA) for cloud computing contracts. These include security at the physical layer, security at the network layer, disaster recovery, and the trustworthy of the encryption schemes.

Chapter 3 – Related Work

In this chapter, we review some of the significant and recent research papers in the field of intrusion detection in cloud computing environments. We present these activities and discuss their advantages and the disadvantages. More precisely, we will discuss the reasons of why these solutions do not give the required intrusion detection as desired in a service oriented model.

3.1 State of the Art

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 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 receive as much attention as the classical intrusion detection activities.

Among the different published works this field is Vieira et. al. [**32**], where they proposed the Grid and Cloud Computing Intrusion Detection System (GCCIDS) which is designed as an audit system for attacks that the networks and hosts cannot detect. In their work, each node identifies local events that could represent security violations and alerts the other nodes. Each individual IDS cooperatively participates in intrusion detection. The system is designed for the purpose of detecting intrusions against the cloud and is not intended for utilization by clients. The protection cannot also be customized by the cloud's clients. Therefore, it doesn't support the requirements of subscription-based intrusion detection service. Dastjerdi et. al. [**33**] implemented applied agent-based IDS as a security solution for the cloud. The model they proposed was an enhancement of the DIDMA [**34**]. The model basically works by sending investigative task-specific Mobile Agent to every virtual host that have generated similar alerts. The mobile agents can then help to verify attacks and later assist in banning the compromised virtual machines and separate them from the network. The system is mainly designed to protect the networks' resources and cannot be customized as a service. Therefore can't support the requirements of adapting it to become a subscription-based IDS service.

Bakshi et. al. [**35**] 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. The attacking network gets blocked and the victim server is transferred to another virtual server. Future connections from the attackers will be blocked and legitimate users are redirected to the new virtual server. As clearly stated, 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 Lo et. al. [**36**]. The Web Service they proposed is mainly designed to create cloud networks that are immune against the Distributed Denial of Service (DDoS) attacks [**37**]. The utilized IDS implementation was the Open Source "Snort" IDS and the Web Service itself is designed as a Distributed Intrusion Detection System (DIDS) [**38**] [**39**]. Mazzariello et. al. [**40**] proposed a model for detecting DoS attacks against Session Initiation Protocol (SIP). The model is limited to detecting SIP flooding attacks and falls largely within the category of intrusion detection systems designed to protect the cloud itself.

Yee et. al [**41**] have proposed an intrusion detection system designed specifically to detect certain attacks against web services. The Web Service they proposed cannot be controlled by users and aims at protecting the web services themselves. Therefore, we can consider this as an intrusion detection system designed to protect the cloud itself which is usually the location where web services are hosted. Bosin et. al. [**42**] have proposed a model for a new generation of intrusion detection systems. The new Web Service, which is designed for security managers, is mainly designed to enable the access to intrusion detection services whenever needed. The Web Service doesn't specify the intrusion detection service itself, but rather focuses on the composition of interoperable intrusion detection (ID) services and aims to promote the reuse of ID tools and systems already available at network nodes and/or supplied by different vendors. The model however assumes the existence of already configured intrusion detection services. It also doesn't allow the clients within the cloud of network to determine the protection requirements but rather use the ID service from one vendor and replaces it if it doesn't suit their demands.

Perhaps the most relevant research was the work of Roschke et. al. [**43**] who have proposed an intrusion detection Web Service based on the VM-based IDS [**44**]. In their work, they have developed a general Web Service 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 Web Service 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. Figure 3-1 illustrates the deployment of the IDS in the cloud in the different possible layers. This approach of separating the IDS from the protected hosts is of great advantage. But this approach is criticized for two things. The first is due to the large consumption of computing resources since every virtual application, platform, or host needs a separate VM-Based IDS and the second is due to the usage of allowing the user to fully control and manage the IDS hosts. The reasons why the second criticism is vital is explained in section 3.3.

3.2 Virtual-Machines Based IDS

Users may legitimately request the full control of the VM-IDS. But this may be actually risky to implement. Virtual interfaces are linked with the physical interfaces. This means that the traffic that is seen in a promiscuous mode for the virtual interfaces is similar. Figure 3-2 illustrates this concept. The figure displays the output of the *tcpdump* command. This command uses the *pcap* (packer capture) libraries to capture all the packets received at the network interface specified in the command line parameters of the command. The figure depicts the packets that are read by the *tcpdump* at a virtual "Ubuntu" machine when another virtual machine is viewing a webpage from an external server. The shown *tcpdumb* output reveals that the virtual machine is capable of receiving the network packets that are being exchanged between the web server and the other virtual machine that share the same NIC. In other words, the same traffic that is sent or received by one virtual machine can be monitored by the other virtual machines that share the same physical interface. The possible risk in this is that a malicious cloud user can write customized rules and IDS

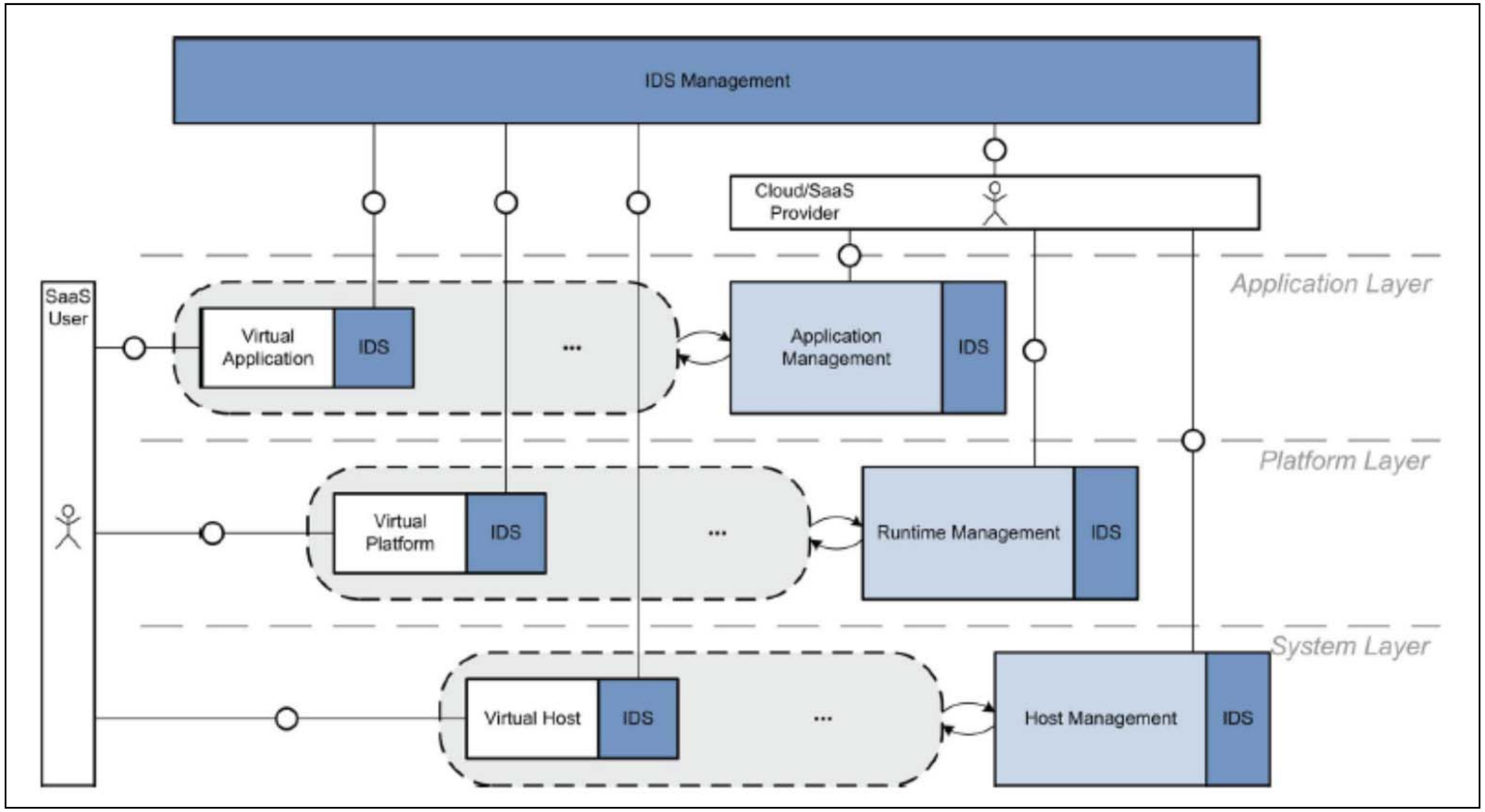
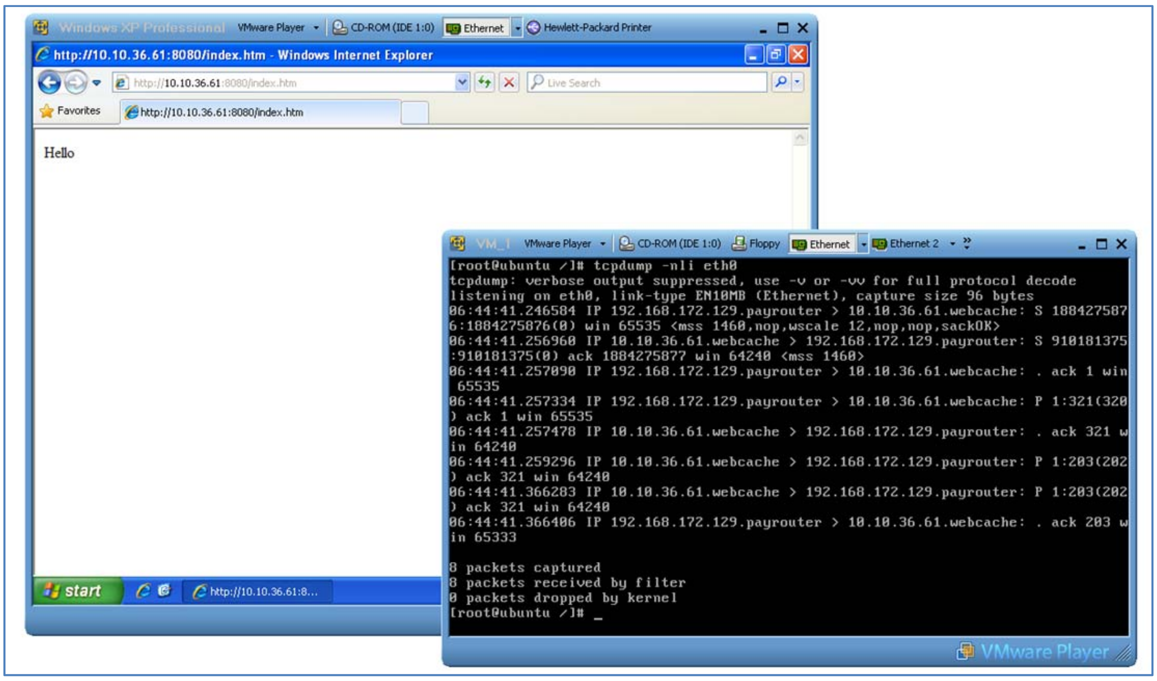


Figure 3- 1: IDS in the Cloud

signatures to monitor the traffic originating from or destined to the other virtual machines. This should not be allowed to happen since it violates a basic principal of information security, i.e. confidentiality.



```
[root@ubuntu /]# tcpdump -nli eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
06:44:41.246584 IP 192.168.172.129.payrouter > 10.10.36.61.webcache: S 188427587
6:1884275876(0) win 65535 <mss 1460,nop,wscale 12,nop,nop,sackOK>
06:44:41.256960 IP 10.10.36.61.webcache > 192.168.172.129.payrouter: S 910181375
:910181375(0) ack 1884275877 win 64240 <mss 1460>
06:44:41.257090 IP 192.168.172.129.payrouter > 10.10.36.61.webcache: . ack 1 win
 65535
06:44:41.257334 IP 192.168.172.129.payrouter > 10.10.36.61.webcache: P 1:321(320
) ack 1 win 65535
06:44:41.257478 IP 10.10.36.61.webcache > 192.168.172.129.payrouter: . ack 321 w
in 64240
06:44:41.259296 IP 10.10.36.61.webcache > 192.168.172.129.payrouter: P 1:203(202
) ack 321 win 64240
06:44:41.366283 IP 10.10.36.61.webcache > 192.168.172.129.payrouter: P 1:203(202
) ack 321 win 64240
06:44:41.366406 IP 192.168.172.129.payrouter > 10.10.36.61.webcache: . ack 203 w
in 65333

8 packets captured
8 packets received by filter
0 packets dropped by kernel
[root@ubuntu /]# _
```

Figure 3- 2: network packets being read by one virtual machine

Chapter 4 – Cloud Intrusion Detection Service

This chapter presents the suggested solution to enable service-oriented access to intrusion detection system process. The chapter begins with an overview of the proposed system and the sequence diagrams of the workflow.

4.1 Overview

The proposed system builds upon the fact that intrusion detection systems utilize very fast and very efficient search algorithms. 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 "Snort" 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. Figure 4-1 summarizes this process. As the figure illustrates, users can choose to subscribe with the intrusion detection service, choose their protection requirements and define any other options that may be available. Once these changes are final, the CRE will translate them to the signature database where they can be deployed and used by the intrusion detection process.

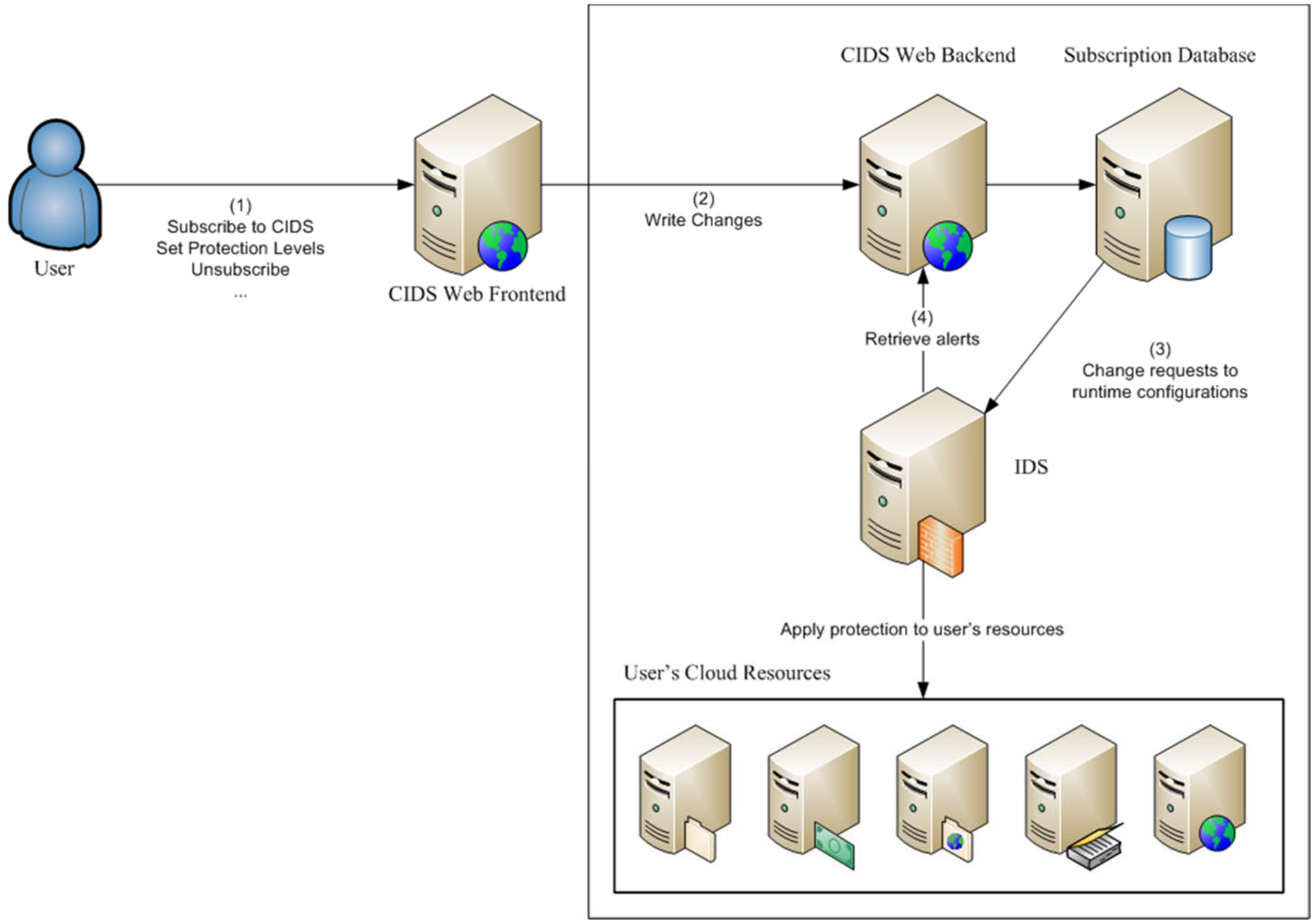


Figure 4- 1: IDS Service within the Cloud

4.2 Cloud Rules Engine (CRE)

This is the most 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 as well. This layer sends the different requests to the other layers in order to convert them to actual IDS configurations. The second layer is the System Layer. This layer will be the driver for the IDS service and will understand both the alerting mechanism and the signature syntax. The third layer is The Database Layer, and its 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. It can also do the actual translations to IDS signature database and also provides the required Application Programming Interface (API) for accessing the alerts database. Figure 4-2 depicts these layers and their interactions within the system.

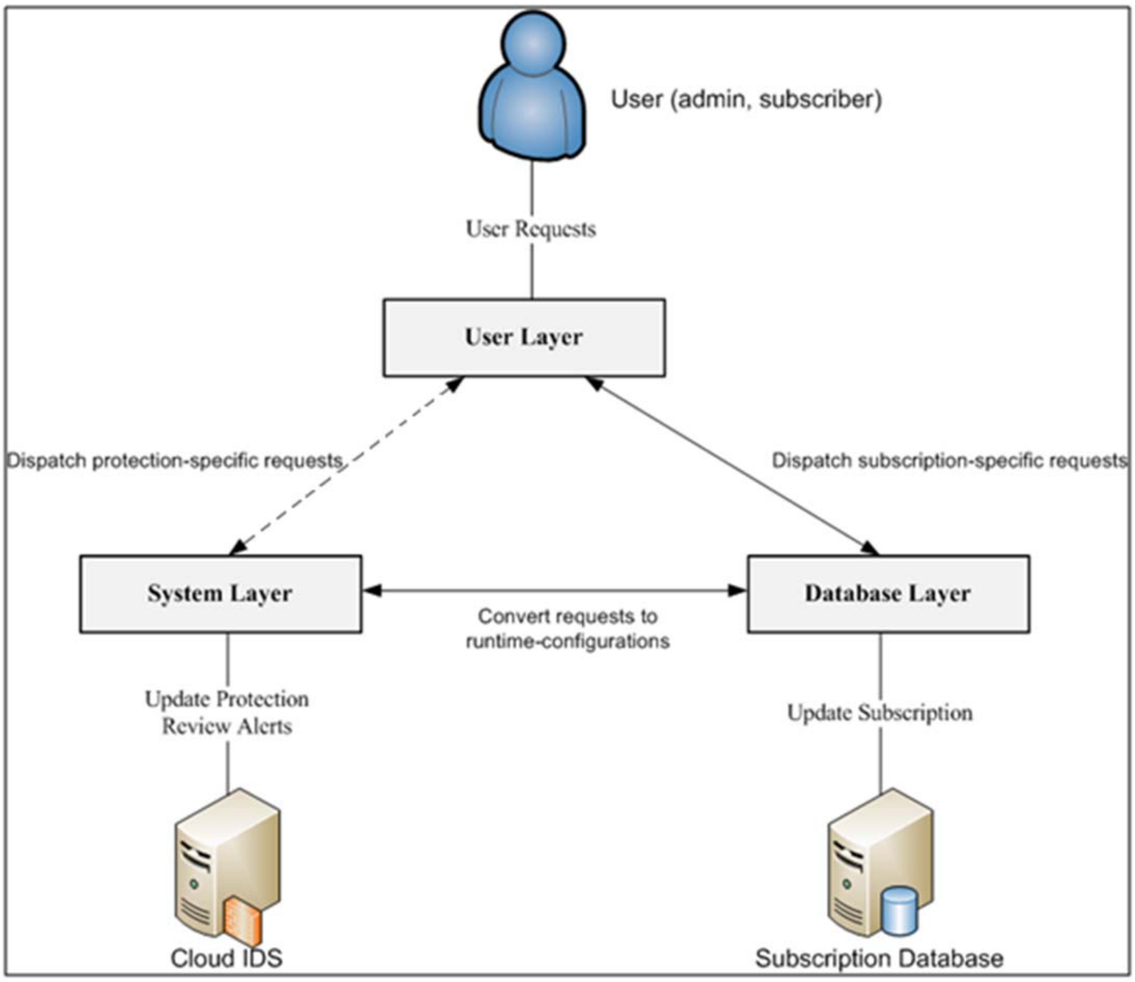


Figure 4- 2: Interactions of CIDS Layers

4.3 CRE Operations

We now describe the operations that are supported with the CRE. These operations are initialized at the User Layer and then dispatched to the two other layers according to the request type. The other layers perform the needed translations to make these requests viable. Before we delve into the details of the supported operations, we review some of the terminology associated with the CIDS design.

- <u>Subscriber</u> – refers to the cloud user who chooses to ‘subscribe’ with the intrusion detection service.

- Administrator – refers to the user who works in administering and managing the CIDS.
- Category – refers to the set of signatures that together work to detect a certain type of attacks and intrusions. A category can include multiple signatures to detect different attack patterns. For example a category called 'SQL Injection' can refer to a set of signatures that detect the SQL injection attacks.
- Alert – refers to the payload that is detected as a malicious one. Alerts usually contain the signature type, the attack source, and attack destination. Also, an alert may optionally contain the name of the subscriber whose resources were the ones targeted by the detected attack.

We now proceed to describe the CRE operations.

4.3.1 Category Operations

These operations are required to add/remove signature categories. The cloud provider may wish to customize standard signature databases in order to optimize the performance of the IDS process. For example, he may manually reorder a public signature database called 'web-attacks' to obtain multiple categories that increase the flexibility of protection options for the cloud users. This can be like converting the single web-attacks database to web-attacks-apache, web-attacks-iis, web-attacks-glassfish, and so on. After these customizations, the cloud administrator will need to deploy these signatures and enable the subscribers to access them. This can be accomplished by the category operation "add" which will receive the new categories as inputs, converts them to cloud capable signatures database, verify them, and finally publish them for subscribers to consider. We mean by cloud-capable signatures, that the signature variables will be set to dynamic mode and the variables defining the targeted networks or hosts are initialized and stored in the IDS System variables. The category operation “remove” works by removing the category from the deployment point and editing the user subscriptions to eliminate the existence of the deleted category.

These sets of operations are first issued as user requests at the User Layer. After that the Database Layer modifies them to support the intrusion detection system variable definitions, and finally deploy them at a publish point recognized by the intrusion detection process. The Database Layer creates entries in the backend database to enable fast and system-independent access to the available categories and subscription details. Each category therefore contains the category name that will appear to the cloud users, a meaningful description that describes the objective behind the category, and also the number of signatures included within this category. The number of signatures will help in two things. First it will help the cloud user to determine the priority of this category. It may be unnecessary to subscribe in a category called general-attacks when it only contains a single signature definition. Second, the cloud provider will utilize the number of rules to determine the charge for subscribing with the given category. It is known that the more signature definitions there are, the more load on the IDS process will be. Therefore, it would be reasonable to charge the subscriber for the number of active signatures he is currently activating. It is worth mentioning that CRE is able to utilize the same signature with the same signature serial number by modifying the variable list only. This means that any active category will only be loaded once despite the number of currently registered subscribers.

4.3.2 Subscription Operations

These operations are used to enable administrators and subscribers alike to review the details of current subscriptions and modify them as requested. A typical operation frequently used by the administrators in scenarios is to view the list of users, perhaps removing or modifying their details. The CIDS provides this function to comply with other standard administration procedures. The cloud users or the subscribers can also use this set of operations to activate their subscriptions with certain categories. They can hence, view the available categories and review their descriptions as well as the available number of signatures in each one of them. Additionally, they can modify their existence subscriptions or completely disable it.

The subscription operations are first initiated at the User Layer, and then translated to the subscription database with the help of the Database Layer. The system layer is used to write translated subscriptions into the cloud IDS service to activate the required protection preferences.

4.3.3 Alert System Operations

These constitute an intuitive set of operations that help the cloud administrator to review the detected attacks targeting the cloud itself and can also help the subscribers to review the alerts generated by attacks detected while targeting their own resources within the cloud. These operations are mostly accomplished by translating the user requests to the System Layer to retrieve the list of detected alerts.

A list of operations that the CRE supports is listed in Table 4-1. These operations are given meaningful names to reflect their intended function.

Table 4- 1: supported operations within the CRE

Type	Name	Target User
Category	AddCategory	Administrators
	RemoveCategory	Administrators
	ResetToDefaultSettings	Administrators
	ViewNumberOfSignatures	Administrators, Subscribers
Subscription	ViewSubscribers	Administrators
	AddSubscribtion	Administrators, Subscribers
	RemoveSubscribtion	Administrators, Subscribers
	SubscribeToCategory	Subscribers
	UnsubscribeFromCategory	Subscribers
Alerting	ViewAlertSumary	Administrator
	ViewAlertsByUser	Administrator, Subscribers

Figure 4-3 below 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.

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.

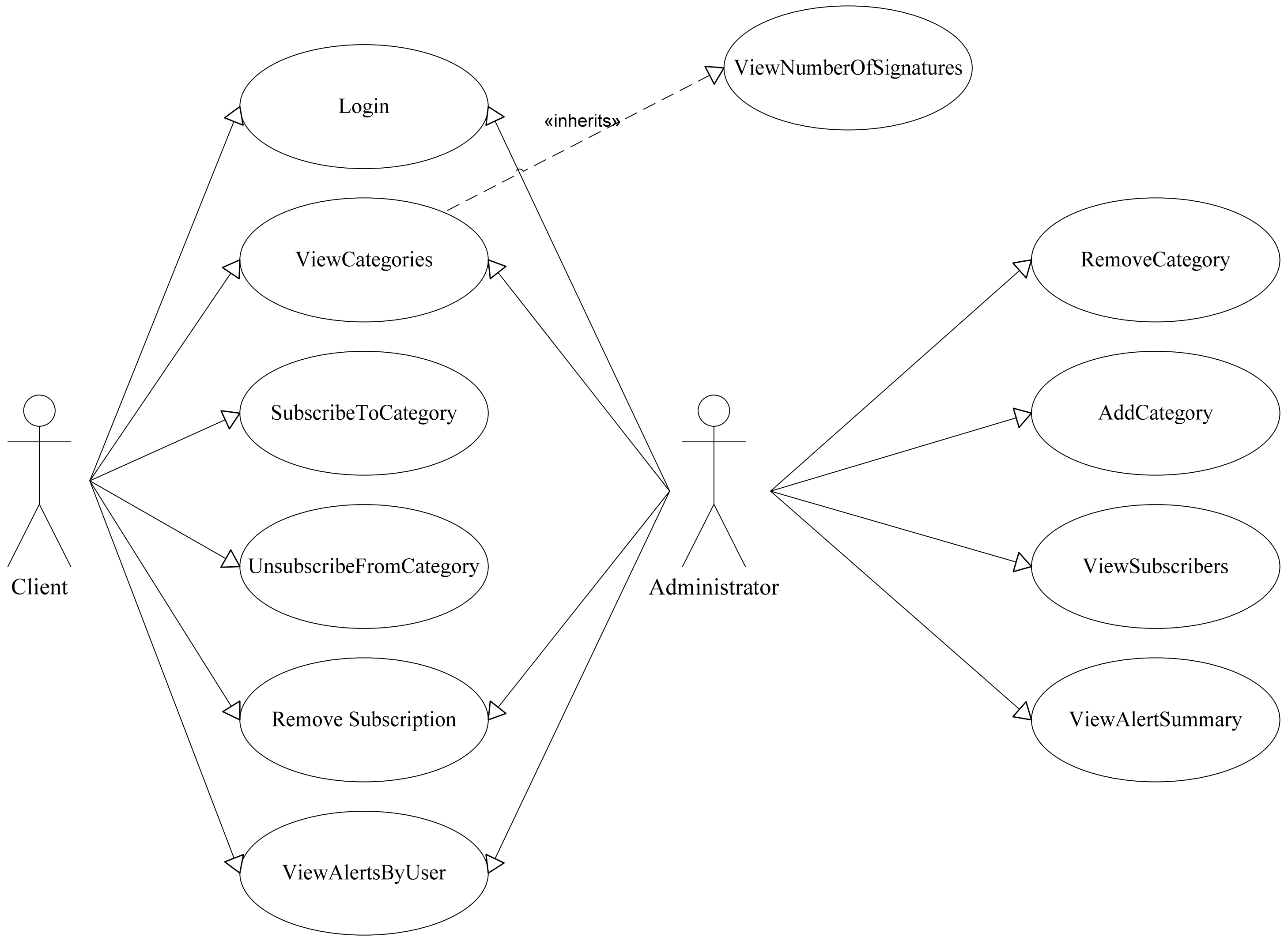


Figure 4- 3: CIDS Use-Cases for clients and administrators

Chapter 5 – CIDS Implementation and Discussion

In this chapter, we review the implementation details of the CIDS architecture. The chapter first describes the proposed system model and then explains the implementation environment and then the final results are summarized. The chapter also compares the CIDS to the implementation of separate process for each user.

5.1 System Model

As described in the previous chapter, the main part of the system is the Cloud Rule Engine or CRE in short. Figure 5-1 views the UML class diagram model for the CIDS system where the different entities and relationships are illustrated. The model suggests many functions that are not required in the original operation specifications mentioned in chapter 3. However, the additional functions are all required for implementation-specific purposes and are not necessarily required to perform the intended functions. For example, the AdminServices and the ConfigManager entities are required for the system but their effect is invisible to the users of the system. As shall be seen shortly, the AdminServices includes functions that are needed to initialize the system while the ConfigManager is needed to parse the CIDS configuration file. Both of them include functions that are not needed for CIDS operations as they are listed in Table 4-1.

The model features multiple entity definitions, these entities or classes and the need for the given operations are all described in shortly.

5.1.1 CategoryManager

This entity is used for category administration purposes. It can be used by the administrators to add a new category or delete an existing one. The AddCategory function receives as input the full path to the new signature database. Then it redefines its variables to comply with the cloud requirements and finally deploy the signature rules set to the publish point that is read by the intrusion detection daemon.

5.1.2 Category

This is where the details of the categories are stored and retrieved. For these purposes, the LoadDetails, CountRules, GetName, and GetDescription are implemented. The final function is the ListDetails which is a static member that can be used to obtain a visual list of all the available categories for the users to review.

5.1.3 AlertManager

The AlertManager, as its name implies, is used to reach the Alerting database and retrieve the details of the detected alerts. The function getAlertSummary and GetAlerttByUser can obtain the list of all detected attacks and the list of detected attacks of a certain service subscriber. The other two functions, i.e. ConvertIpToLong and ConvertLongToIp are used to convert the integer representation of the IP addresses to the A.B.C.D notation and vice versa. This is required for this implementation since the addresses are stored as numeric values of long datatypes. Each detected alert is represented as an instance of the Alert entity. The Alert entity stores the details of every alert, which include the source and destination addresses, the signature type and identifier, the name of the subscriber whose assets were targeted by the detected attack, and finally a simple utility method for representing the alert as HTML statement in order to display the alert directly on a special web page.

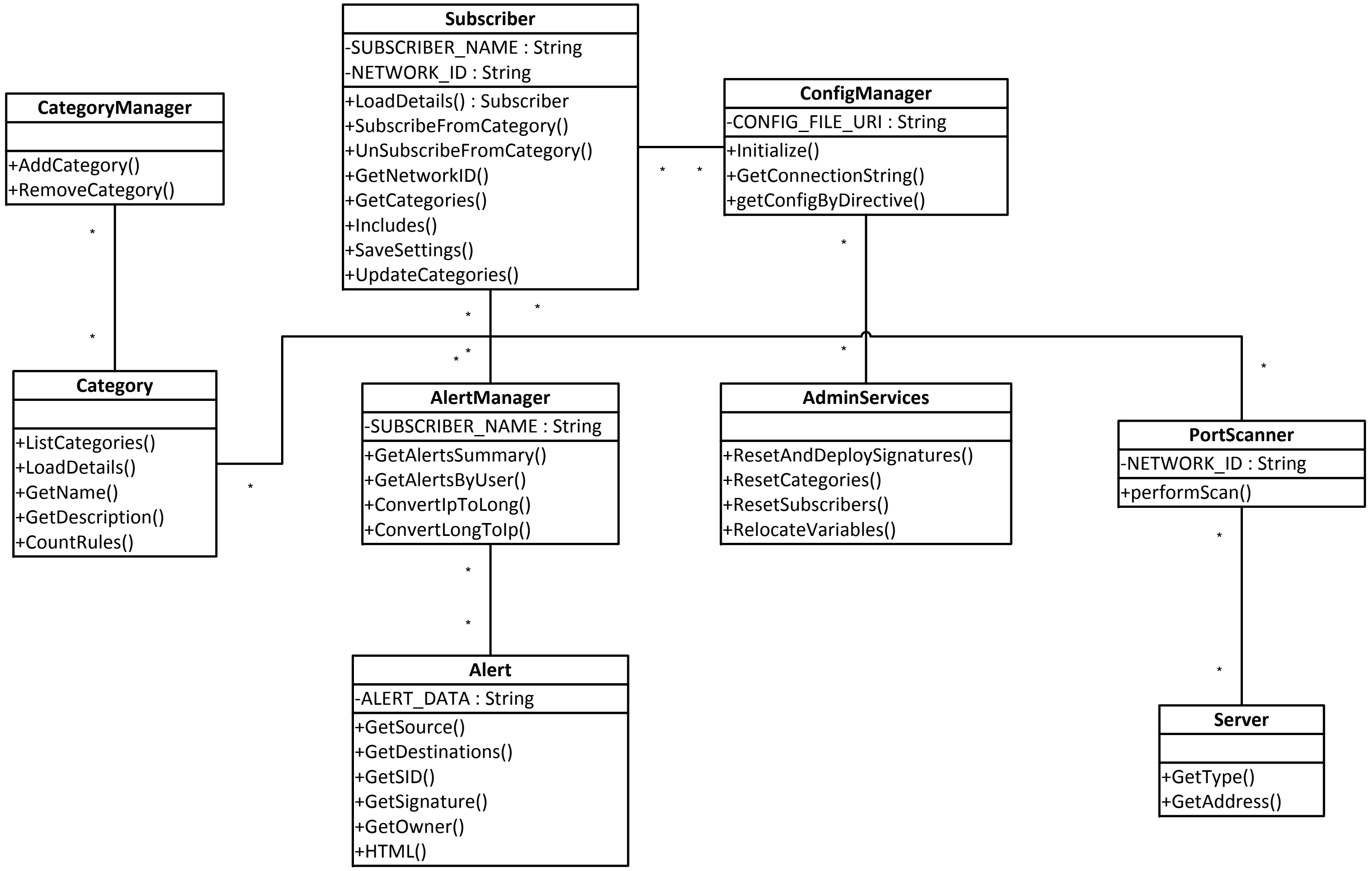


Figure 5- 1: UML Class Diagram Model for Entities in the CRE Implementation

5.1.4 PortScanner

This class utilized the famous Network Scanner “nmap” [**45**]. This tool can be used to scan the network and obtain a list of all open ports on all of its active hosts. The original specification didn’t include this function, but the implementation required such a function to facilitate the configuration and optimize the settings. In short, the “Snort”® IDS that we used as the intrusion detection process contains a set of rulesets. Within these rule sets , multiple special variables are defined like SMTP_SERVERS, TELNET_SERVERS, SQL_SERVERS, HTTP_SERVERS, and others. The traditional “Snort”® deployment requires that a user specifies the values of these variables manually. However, in a cloud network where the service usability is a main concern, users may not be able to enter these exact values. This is where the PortScanner class comes in handy, the class contains a single function PerformScan which do the actual scanning using nmap where the final output is a list of Server entities, where each Server entity represents a server. Each server contains the IP address of the server and the port number on which it listens, a single host that contains an HTTP service and an SMTP service is represented by two Server instances.

5.1.5 AdminServices

This entity was used during the testing period, it contains the functions required to *flush* the CRE system. By flush we mean reset the subscription information, re-initialize the variables list, remove the deployed signature databases, and reset all the category information to the original settings where no subscribers exist. This class is not required, still, it can be used by any cloud provider who wishes to install and utilize the CRE engine on their cloud environment.

5.1.6 ConfigManager

This entity is used to read the CRE configuration. It then parses the configuration directives and stores them as a HashMap. The entity provides two functions. The first is the GetConfigByDirective which is a static member that is used to obtain the value of a specific configuration directive. The second static member is the GetConnectionString, this is really not needed as a series of GetConfigByDirective calls can be used to replace it. However, because it is frequently called and because the connection string contains multiple directives, it was implemented as an independent function. A typical connection string includes the [host name, port number of database management system, and the database name].

5.2 Implementation Environment

As a proof of concept implementation, The CIDS was implemented using the Java 2 Enterprise Edition (*J2EE*) [**46**]. The entities in the UML class diagram model was translated to Java-based classes. The web interface was created using Java Server Pages (*JSP*) [**47**] technology while the implementation code was written using *NetBeans 6.9.1 IDE* [**48**] installed on *Ubuntu 10.04* [**49**] Linux distribution.

5.3 CRE Web Interfaces

The CRE system model has successfully been implemented and tested. The final implementation contains two simple web folders for cloud administrators and cloud subscribers a like. Following is a description of the implemented functions within the web interface of the CRE. Appendix A illustrates the web interfaces for CIDS

Once the cloud administrator logs in to the control panel, he will see a list of choices that are available for administration procedures. These include the (Add Category, Delete Category, Subscribers List, Categories Available, and the summary of all the alerts detected including alerts detected by traffic destined to the subscribers networks or the cloud network itself. In this case the cloud is considered a subscriber. The categories that are activated by default are said to be the ones the cloud is subscribed with.

The Add category function available to the administrator provides a very simple form designed to receive as input the full path to the new category signature database. The form sends the content of the file along with the descriptions to the CIDS server, where the database passes through some processing to convert it to cloud-capable signatures database. Then the new database gets added to the list of available categories so that subscribers may be able to subscribe with it. Similar functions apply to the *Delete Category* function, which receives the name of the category to be removed and then performs some processing to eliminate the database from the ones available for subscription. After this, the CIDS revokes all subscriptions to the deleted category. This function can be used to quickly remove any buggy signature database that generates *false positives*. A false positive is defined as a positively detected attack while the actual traffic is not really an attack.

On the other hand, a client must first login or create a new account. An account contains the name of the subscribing user and the network segment that is desired to be included with the protection. Then, the subscriber may be able to subscribe to/unsubscribe from the available categories. He also can view a list of all the current categories. For this simple implementation, the client can view the category name as well as the number of signatures included and the number of subscribers who are currently subscribed to that category. The number of subscribers can be useful to give an indication to the clients on the popularity and reliability of the signature database.

The implemented interfaces for both the cloud administrator and the cloud user are made as a proof-of-concept only while the real implementation may include more complex and user-friendly interfaces. However, this is not the main concern of this research which focuses more on providing a cloud-capable service-based intrusion detection system.

5.4 Results and Discussion

Due to the unique requirements that are set before designing and implementing the CRE component of the Cloud Intrusion Detection System (CIDS), the simulation is based on two related scenarios. The first scenario is for the comparison of the CIDS design with the simple implementation of "Snort" for single client protection. This scenario aims at estimating the overhead obtained when implementing the CIDS Web Service using "Snort" as the base IDS. In this scenario, we follow similar performance analysis outcomes as the ones mentioned in Lo et. al. [**36**]. Specifically, we measure the "Snort" process size, the detection rate of extremely hostile traffic, and the time per packet that is required to analyze and detect attacks. The second scenario aims at comparing the resource consumption of CIDS Web Service when compared to other solutions which allocate distinct process for each user, similar to Virtualized IDS allocation in Roschke et. al. [**43**]. This scenario is created to compare the effectiveness of using shared ID process among subscribers to the case of using separate process for each subscriber. In this scenario we simply compare the process size for using single shared setting as in CIDS with the case of using the ID process with different protection profiles for each user, where the protection profile are in the form of customized protection packages.

As mentioned earlier, the main objective was to design and architect an intrusion detection system that can be run in a service-based manner. This service-based design would then allow the cloud users to be able to take advantage of the service to include their valuable assets with the available IDS. This can be classified as a Security-as-a-Service service model.

Table 5-1 reviews the comparison of CIDS with the relevant published researches on integrating intrusion detection systems with cloud computing networks. As stated in the table, the CIDS have enough advantages that the other cloud IDS solutions do not have. As it is shown in the table both Yee and Bosin works are designed for client protection. But as stated earlier, Bosin work focuses on managing access to multiple ID sensors by different vendors. Clients can't change the options of a any single ID process but rather change to another ID by another provider if the options are not suitable. On the other hand, Yee focuses of protecting the web service in the cloud. These web services are owned or used by cloud clients, but the protection is fixed and cannot be customized or accessed by the clients.

Table 5- 1: Comparing CRE with other cloud-based intrusion detection systems

System	CIDS	Roschke	Lo	Bakshi	Yee	Bosin
Service-Based	Yes	Yes	No	No	No	Yes
Customizable Subscriptions	Yes	Yes	No	No	No	No
Designed for Clients Protection	Yes	Yes	No	No	Yes	Yes
General Protection	Yes	Yes	No	No	No	Yes
Fully Parallelizable	Yes	No	Yes	Yes	Yes	Yes
IDS Separated from Networks	Yes	No	Yes	Yes	Yes	Yes

5.4.1 First Scenario

The first scenario consists of running the standard "Snort" process for a single class C network protection and comparing it to the case of using CIDS to protect multiple Class C networks. The number of subscribed rules varies from 200 rules to 1000 rules.

Figure 5-2 displays the results obtained when measuring the process size for each of the two cases. As clearly visible in the figure, the process size for using the CIDS to protect multiple networks is very similar to using "Snort" for protecting a single network. This behavior stays the same despite the number of subscribed signatures. The architecture of CIDS Web Service is expected to give such result since the only addition difference is length of the same number of signatures. The remaining process components are all the same. The process size overhead is very small compared to the original snort process size. This overhead is negligible because we are comparing the case of protecting a single network and the case of protecting multiple networks.

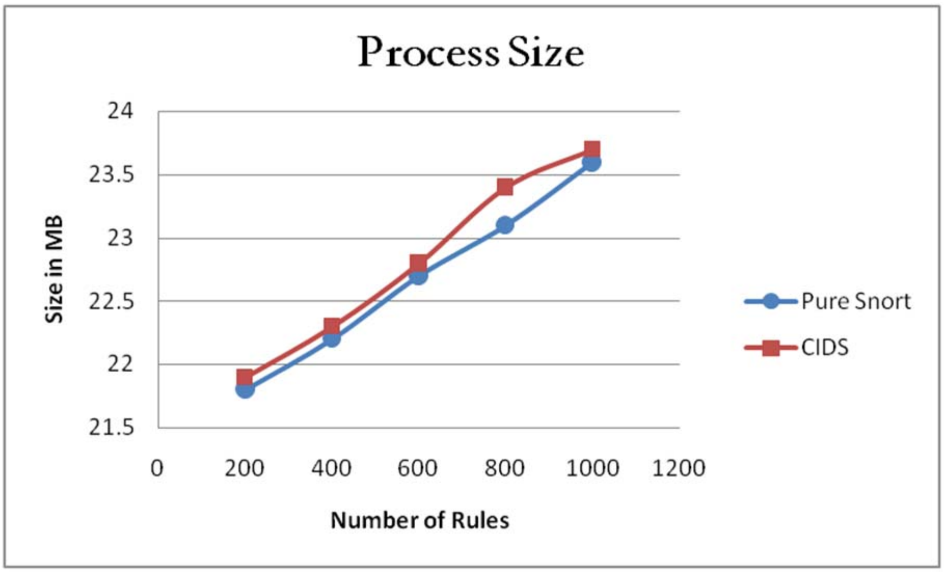


Figure 5- 2: Process size in (MB) for using CIDS and standard "Snort"

We now measure the detection rate of heavy and hostile traffic. Figure 5-3 shows the obtained results for the effective attack detection rate for the two cases mentioned earlier. As the figure illustrates, the average attack detection rate is very high for the two cases.

Despite the degradation in performance for the CIDS compared to the pure "Snort" implementation, the detection rate is still very high since we are dealing with CIDS protecting multiple networks. This result also proves that the CIDS is very effective when it comes to detecting attacks in real-time.

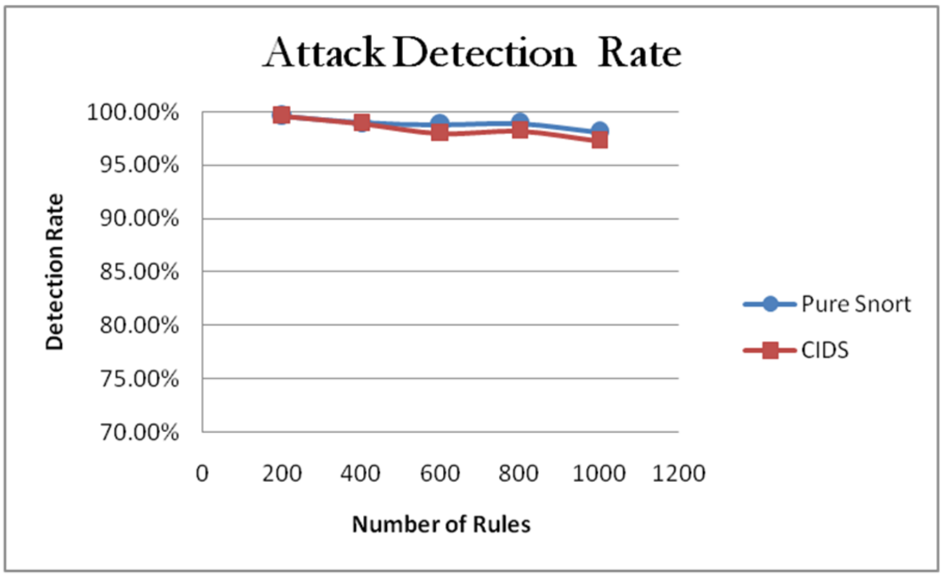


Figure 5- 3: Attack detection rate in (%)

Finally, we measure the average time required to fully analyze and detect attacks in network packets. This aims at estimating the time the ID process needs to detect an attack when protecting a single network and when protecting multiple networks using CIDS. Figure 5-4 shows these results.

The obtained results are very satisfying. Because it only takes 1 milliseconds extra time to detect an attack that targets any of the multiple protected networks when compare it to time required to detect attacks against a single network. This time is relatively small and can be considered an accepted compensation to enable large-scale attack detection against numerous networks at once.

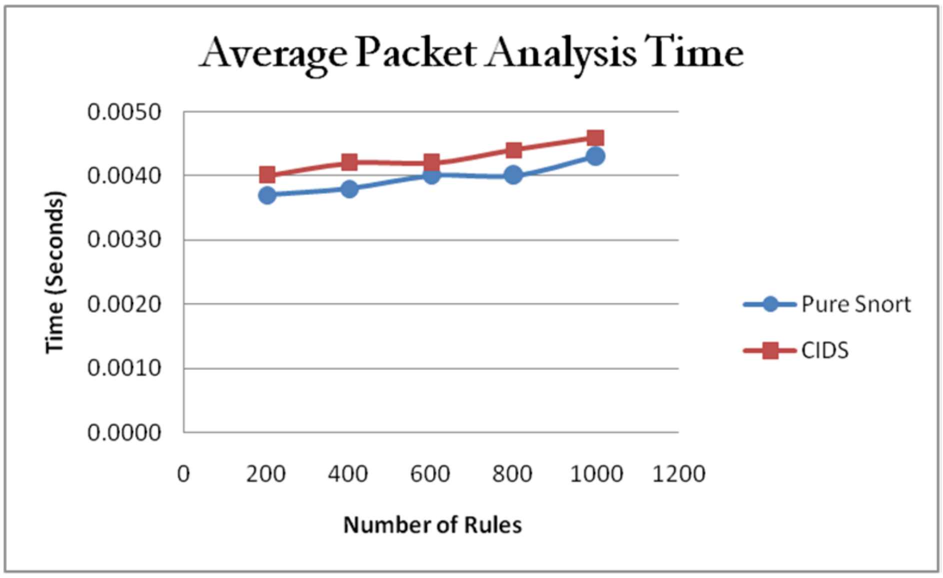


Figure 5- 4: Average packet analysis time

5.4.2 Second Scenario

As mentioned earlier, the second scenario is for measuring the expected improvement when using the CIDS architecture compared to using separate profile for each subscriber. In order to do this, we first compare the memory consumption for using CIDS when varying the number of subscribers from 100 to 500 subscribers with a protection profile consisting of 1000 signatures. Figure 5-5 displays the obtained results.

At first glance, the results may seem as way too good results. But we need to point out that this is only a demonstration of usefulness of the CIDS architecture that utilizes fewer ID processes to protect the requesting subscribers. The other configuration is similar to utilizing separate resources for each subscriber. Despite the fact that every subscriber will have relatively good results on his own 'virtualized IDS', the cost of supporting numerous and separate ID processes for subscribers is too high for the cloud provider.

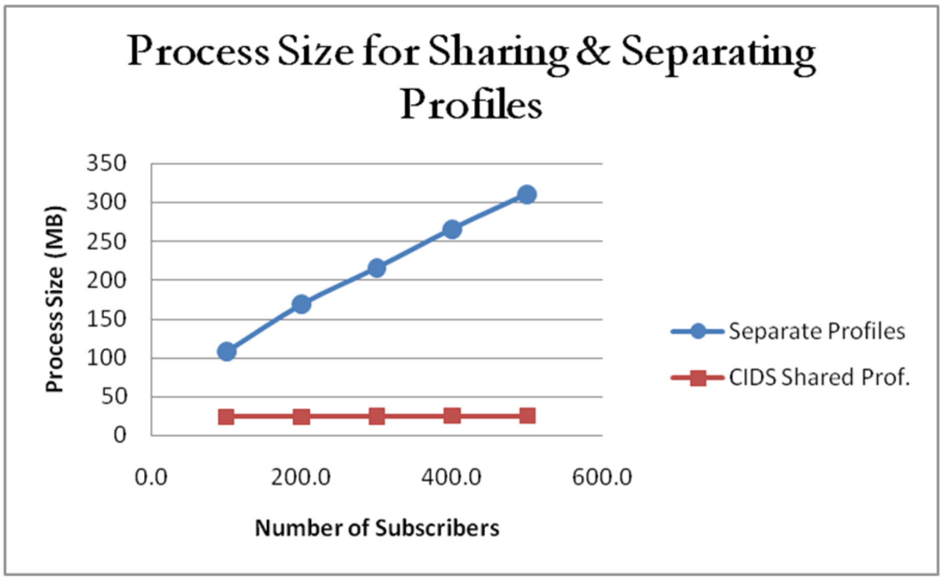


Figure 5- 5: Process size in (MB) for using Sharing & Separating profiles

We measure the other two performance metrics that are useful to the cloud clients as well as the provider. These are the attack detection rate and the average attack detection time. As mentioned earlier, these metrics were measured by Lo et. al. to prove the efficiency of their proposed IDS framework. We here extend these results to measure the relative performance of our CIDS Web Service with Roschke-like implementations of separate settings and processes for each client. Figure 5-6 displays the attack detection rate for CIDS and the Separate setting scheme.

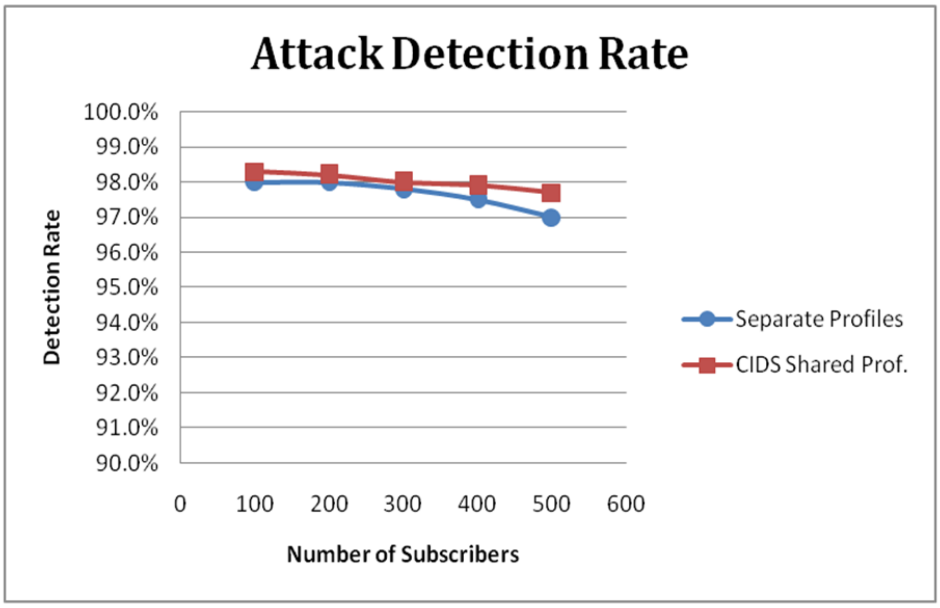


Figure 5- 6: Attack detection rates

The CIDS Web Service 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. 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.

Finally, we measure the average attack detection time, which aims at measuring the speed of the IDS Architecture in detecting attacks. Figure 5-7 views the results of this performance metric.

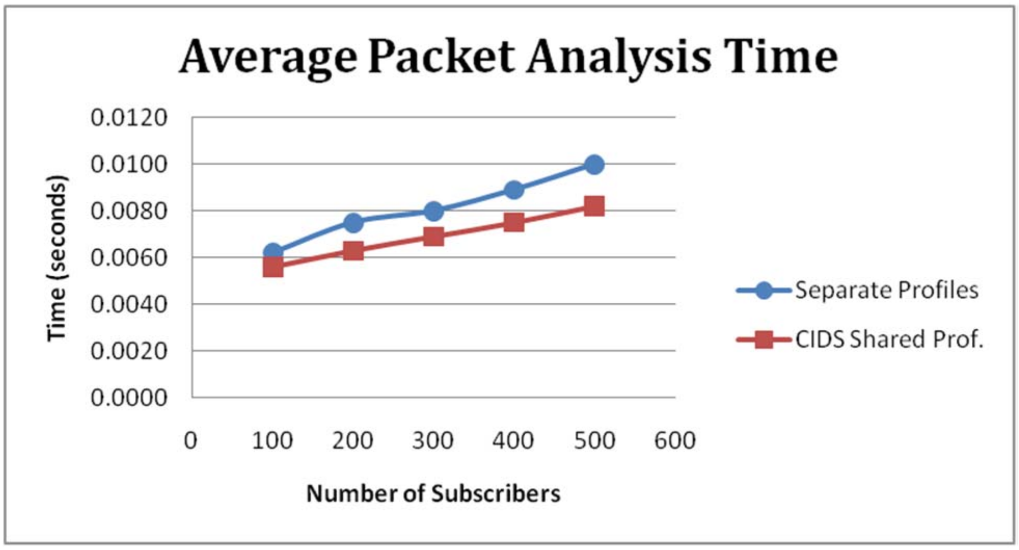


Figure 5- 7: Average packet analysis time

As shown in the results, the shared scheme of the CIDS Web Service 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.

The first scenario have proved that the sharing scheme in CIDS does provide very small overhead to the ID processes regarding to only using the ID processes for one network protection only. But as proved in the second scenario, this minimal overhead can be very advantageous to the cloud providers, who can save too many resources by applying CIDS Web Service in their cloud networks for providing intrusion detection services to the clients. On the other hand, the cloud clients and subscribers can also benefit from the CIDS Web Service as it provides better attack detection rate as well as detecting the attacks faster than the other scheme of separating resources and settings for each client.

Chapter 6 – Conclusion

6.1 Summary and Concluding Remarks

In this thesis, we have designed and implemented the Cloud Intrusion Detection Service (CIDS) which was proved to be a very effective solution to the problem of providing intrusion detection as a service for cloud environments. The system which consisted of three separate layers (User Layer, System Layer, and Database Layer), aims to develop a scalable intrusion detection Web Service that can be deployed by cloud providers to enable the clients to subscribe with the intrusion detection in service based manner. The system is reengineering of the existing intrusion detection system ("Snort") but can be implemented with other systems if required. The three layers are recommended to be separated to different machines.

The User Layer is the interface at which the users (clients and administrators alike) interact with the system. It consists of various web pages that receive users' subscription-related actions and translate them to configuration settings to the intrusion detection system at the System Layer. The System Layer needs to utilize highly efficient Network IDS (NIDS) to be able to analyze the enormous traffic volumes found in typical cloud networks. The CRE was developed as part of the core components in the CIDS to be the tool that generates the configuration files and manage attack signature databases. The CRE rewrites the categories that a client subscribes with and include the definition of his network segment with the signature itself. The signature end up to be a new attack-signature and is loaded to the IDS system afterwards.

A very important feature is the ability to reload the IDS process quickly. This is required because when new users subscribe to the system, the intrusion detection process needs to reload the signature databases so that it can include the new settings. By comparing CRE and regular profile-based subscription, we found that the load time is extremely superior in the case of CRE. An additional requirement was to have a memory-efficient system. The CRE was also proved to be very efficient to memory utilization. We can verify the superiority in memory utilization for the CRE compared to the other solution. Finally, we needed to make sure that the previous configurations can be efficient when placed in real environments. For this to be made, we retested the systems once again where this time we wanted to measure the rate at which the packets can be fully analyzed. We used a moderate 500-signatures database. These results give the final confirmation that the CRE can indeed generate a very efficient shared signature databases that would be equal in performance to the case when only one subscriber is concerned.

The final layer was the Database Layer, which is needed to keep track of all the current configurations and subscription information for later web access at the User Layer. As the name implies, this layer contains the subscription information stored in as database records. Also, this layer can be used to dig in the alert database and review the alerts related to each and every subscriber.

As a final conclusion, the CIDS have been a successful Web Service for managing intrusion detection systems in cloud networks and provide it as a service to the cloud clients. The system component of the CIDS is very scalable and extremely effective in memory utilization, and supports large volumes of traffic.

6.2 Recommendation and Future Work

Enabling access to IDS modules in the cloud in subscription-based manner is highly motivated. It is also highly recommended to separate the IDS process from the network assets controlled owned by the users. This enables the cloud providers to fully control the IDS process and its configuration. Additionally, it is of great advantage to enable the subscription on the level of protection packages. With this, the providers can make a huge list of protection options available to clients and at the same time update these packages overtime whenever required. So the subscribers will always be subscribed to packages that provide optimal performance and optimal protection coverage. Finally, it is also recommended that the protection packages be used as templates and that they are shared amongst all the subscribing clients. This would reduce the resource allocation required to deploy the IDS module and simultaneously, increase the attack detection rates and the time required to spot attacks.

Currently, only Snort NIDS is supported within CIDS. However, for future endeavors, it is of great advantage to include additional NIDS solutions to the system. Additionally, and since the CIDS depends on using a single ID process, it is very important to consider the fault-tolerance techniques. Possibly by parallelizing the architecture of CIDS.

References

[1] J. Li and Y. Yang, "Design and Realization of a Large-scale Distributed Intrusion Management," in *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008, pp. 537-540.

[2] R. Maggiani, "Cloud Computing is Changing How We Communicate," in *IEEE International Professional Communication Conference*, Waikiki, HI, 2009, pp. 1-4.

[3] George Reese, *Cloud Application Architectures*, 1st ed.: O'Reilly Media, 2009.

[4] B. P. Rimal, E. Choi, and I. Lumb, "A Taxonomy and Survey of Cloud Computing Systems," in *Fifth International Joint Conference on INC, IMS, and IDS*, 2009.

[5] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *Grid Computing Environments Workshop, GCE'08*, 2008.

[6] M. Hanaoka, K. Kono, and T. Hirotsu, "Performance Improvement by Means of Collaboration between Network Intrusion Detection Systems," in *7th Annual Communication Networks and Services Research Conference*, 2009, pp. 262-269.

[7] R. A. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview," *IEEE Security and Privacy Magazine*, vol. 35, no. 4, pp. 27-30, April 2002.

[8] Ch. Dougligeris and D. N. Serpanos, *Network Security: Current Status and Future Direction*, First Edition ed.: John Wiley & Sons, 2007.

[9] C. Endorf, E. Schultz, and J. Mellander, *Intrusion Detection & Prevention*, 1st ed., Marcia Baker, Lisa Theobald Andy Carroll, Ed.: McGraw Hill, 2004.

[10] J. McHugh, "Intrusion and Intrusion Detection," *International Journal of Information Security*, vol. I, no. 1, pp. 14-35, August 2001.

[11] T. Verwoerd and R. Hunt, "Intrusion Detection Techniques and Approaches," *Computer Communications*, vol. 25, no. 15, pp. 1356-1365, September 2002.

[12] M. Esposito et al., "Intrusion Detection and Reaction: an Integrated Approach to Network Security," *Advances in Information Security*, vol. Intrusion Detection Systems, 2008.

[13] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Martin_Roesch, Last Accessed: March 31, 2011

[14] SouceFire. [Online]. http://www.sourcefire.org/, Last Accessed: March 31, 2011

[15] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proceedings of LISA '99: 13th Systems Administration Conferemce*, Seattle, 1999.

[16] George Pallis, "Cloud Computing: The New Frontier of Internet Computing," *IEEE Internet Computing*, pp. 70-73, September/October 2010.

[17] A. T. Velte, T. J. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*.: McGraw Hill, 2010.

[18] Google. Google App Engine. [Online]. http://code.google.com/appengine/, Last Accessed: March 31, 2011

[19] 'In The Cloud' Security Services Hit the Peak of the Gartner Hype Cycle in 2009. [Online]. http://www.gartner.com/it/page.jsp?id=1179113, Last Accessed: March 31, 2011

[20] McAfee Security. Security as a Service. [Online]. http://www.mcafee.com/us/small/security_insights/security_as_a_service.html, Last Accessed: March 31, 2011

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

[22] Webroot Security. [Online]. http://www.webroot.com/, Last Accessed: March 31, 2011

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

[24] Panda Security. [Online]. http://www.pandasecurity.com/, Last Accessed: March 31, 2011

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

[26] Cliff Riggs, *Network Perimeter Security: Building Defense In-Depth*.: Auerbach Publications, 2003.

[27] Amazon. Amazon Elastic Compute Cloud (Amazon EC2). [Online]. http://aws.amazon.com/ec2/, Last Accessed: March 31, 2011

[28] SourceFire. (2010, July) Snort News & Events. [Online]. http://www.snort.org/news/2010/07/07/snort-now-available-on-the-amazon-cloud/, Last Accessed: March 31, 2011

[29] Priscilla Oppenheimer, *Top-Down Network Design, 3rd Edition*. Indianapolis: Cisco Press,

2011.

[30] B. R. Kandukuri, R. Paturi, and A. Rakshit, "Cloud Security Issues," in *IEEE International Conference on Services Computing*, 2009, pp. 517-520.

[31] K. Vieira, A. Schulter, C. B. Westphall, and C. M. Westphall, "Intrusion Detection for Grid and Cloud Computing," *IT Professionals*, pp. 38-43, July/August 2010.

[32] 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*, Sliema, Malta, 2009, pp. 175-180.

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

[34] 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.

[35] 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.

[36] Wikipedia. Denial-of-service attack. [Online]. http://en.wikipedia.org/wiki/Denial-of-servoce_attack, Last Accessed: March 31, 2011

[37] 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.

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

[39] 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.

[40] Ch. Yee, W. Shin, and G. Rao, "An Adaptive Intrusion Detection and Prevention (ID/IP) Framework for Web Services," in *International Conference on Convergence Information Technology*, Gyeongju , 2007, pp. 528-534.

[41] A. Bosin, N. Dessì, and B. Pes, "Service Based Approach to a New Generation of Intrusion

Detection Systems," in *Sixth European Conference on Web Services*, Dublin, 2008, pp. 215-224.

[42] 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.

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

[44] G. Lyon. Nmap Security Scanner. [Online]. http://nmap.org/, Last Accessed: March 31, 2011

[45] Oracle America. Oracle Corp. [Online]. http://www.oracle.com/technetwork/java/javaee/overview/index.html, Last Accessed: March 31, 2011

[46] Sun Microsystems. Sun Developer Network (SDN)). [Online]. http://java.sun.com/products/jsp/, Last Accessed: March 31, 2011

[47] Oracle Corporation. NetBeans. [Online]. http://netbeans.org/, Last Accessed: March 31, 2011

[48] Canonical Ltd. Ubuntu Linux. [Online]. http://www.ubuntu.com/, Last Accessed: March 31, 2011

[49] E. V. Breusegem et al., "Grid Computing: the next network challenge!," in *First Conference on Broadband Network*, Broadnets, 2004.

[50] P. Kokkinos and E. A. Varvarigos, "Resource Information Aggregation in Hierarchical Grid Networks," *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009.

Appendix A – Web Interfaces of CIDS

A.1 Sample Administration Web Interfaces

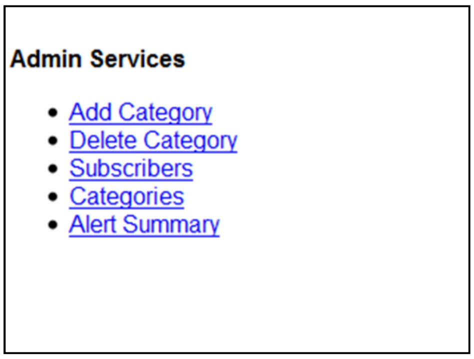


Figure A: 1 - Administrator Main Panel

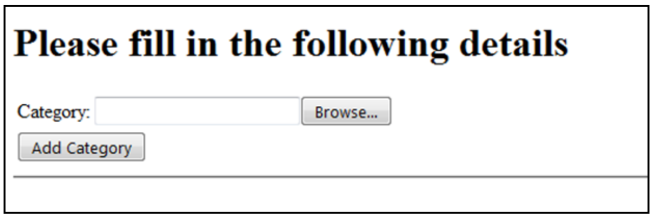


Figure A: 2 - Add Category Page

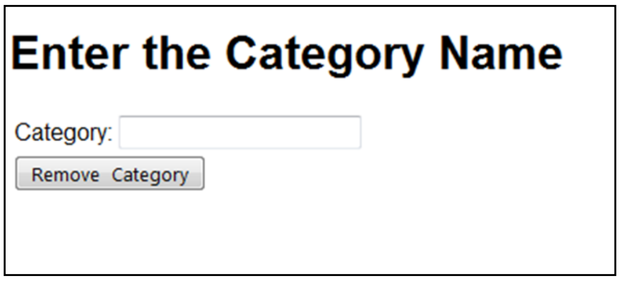
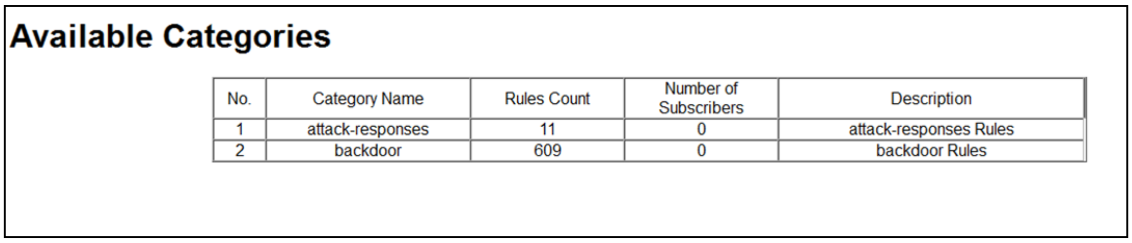


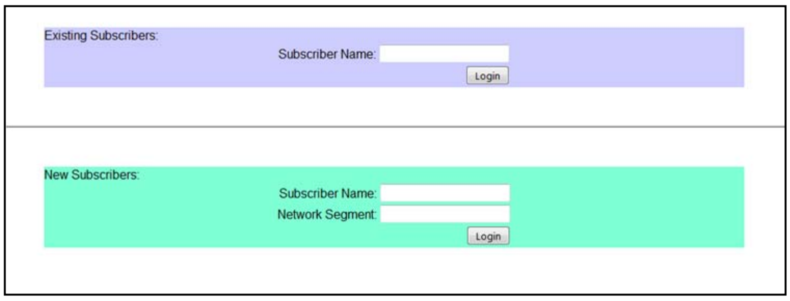
Figure A: 3 - Remove Category Page


Available Categories

No.	Category Name	Rules Count	Number of Subscribers	Description
1	attack-responses	11	0	attack-responses Rules
2	backdoor	609	0	backdoor Rules

Figure A: 4 - Available Categories Page

A.2 Sample Clients Web Interfaces


Existing Subscribers:

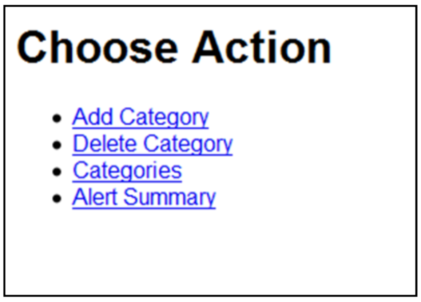
Subscriber Name:

New Subscribers:

Subscriber Name:

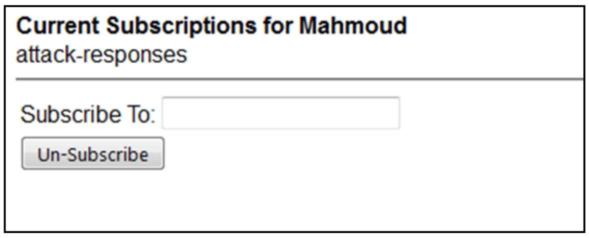
Network Segment:

Figure A: 5 - Clients Login Page


Choose Action

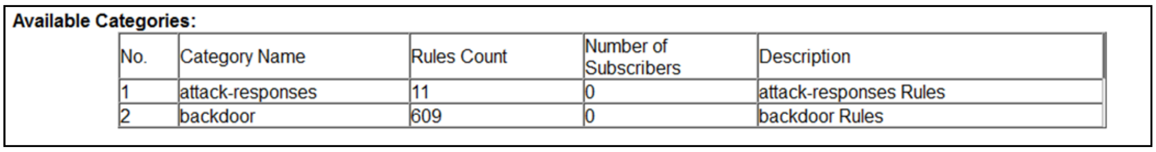
- Add Category
- Delete Category
- Categories
- Alert Summary

Figure A: 6 - Client Options Page


Current Subscriptions for Mahmoud
attack-responses

Subscribe To:

Figure A: 7 - Unsubscribe From Category Page


Available Categories:

No.	Category Name	Rules Count	Number of Subscribers	Description
1	attack-responses	11	0	attack-responses Rules
2	backdoor	609	0	backdoor Rules

Figure A: 8 - View Categories Page