Event-Driven Multi-tenant Intrusion Detection System

by

Mohamed HAWEDI

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, "OCTOBER,09, 2019 "

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

M. Chamseddine Talhi, Thesis Supervisor
Département de génie logiciel et des TI,École de technologie supérieure ÉTS

Mrs. Hanifa Boucheneb, Co-supervisor
Department of Computer Engineering and Software Engineering, Polytechnique Montréal

M. Chakib Tadj, President of the Board of Examiners
Département de génie électrique, École de technologie supérieure ÉTS

M. Mohamed Faten Zhani, Member of the jury
department of software and IT engineering, École de technologie supérieure ÉTS

M. Jamal Bentahar, External Independent Examiner
Concordia Institute for Information Systems Engineering, Concordia University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "DEFENSE DATE"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

FOREWORD

This thesis presents my research that was work has carried out between 2014 and 2019 under the supervision of Professor Chamseddine Talhi and Professor Hanifa Boucheneb. The main goal of the research is to address and provide solutions to the security issues related to cloud tenants.

The dissertation is organized in form of *integrated articles*. The articles are included in their original published and submitted form; no changes have been added to them.

For this research work, three articles are included. The articles are inter-related as they describe related solutions for different aspects of multi-tenant IaaS cloud. Separate chapters are devoted to the background and the literature review. The three journals are represented in Chapters 2, 3, and 4. Furthermore, in each article, a separate section is devoted to the background and the related work. Also, a special review is presented on the research problems and the contributions relevant to that particular work.

ACKNOWLEDGEMENTS

I would like to extend my sincere and heartfelt gratitude to everyone who has contributed to making my dreams come true by making this thesis possible. Indeed, without their support, this thesis would not have been achieved.

First of all, I would like to express my special appreciation to my advisors Professor Chamseddine Talhi and Professor Hanifa Boucheneb for their continued support of my Ph.D. studies and the related research. I am very grateful to you all. You gave me enthusiasm, motivation, guidance and patience. You also helped me with immense knowledge that enabled me to progress and to complete this thesis.

My special appreciation and thanks to Professor Chamseddine Talhi for providing his tremendous guidance throughout this research. I also would like to express my sincere gratitude for encouraging and guiding my research. Your immense knowledge and advice on both my research as well as on my career have been invaluable. Dear Talhi, your deep insights and fruitful discussions assisted me in reaching this stage. I will never forget your constant encouragement and wonderful smile that left a deep impression on me and always made me able to move forward and not give up. Really, you have been a great mentor.

My deepest sense of gratitude to Professor Hanifa Boucheneb for all what she has done for me. I really appreciate your integrated perspectives on research, motivation and support. Thanks for inspiring me with positive words and blessings.

Many thanks also go out to Libyan Ministry of Higher Education and Scientific Research for providing the financial support that made this work possible.

I would like to express my deep gratitude to the committee, Professor Chakib Tadj, Professor Mohamed Faten Zhani, and Professor Jamal Bentaha, for their efforts and insightful comments.

I am deeply indebted to my wife for everything she has given me and without her sincere prayers, understanding and dedication this Ph.D. would not have been possible. Special Thanks to my beloved children for cheering me up and bringing happiness to me.

I am deeply indebted to my beloved mother, father, brother and sisters for their sincere prayers and unconditional support during these years.

Finally, a special thanks to all my friends. The time we spent together and the unconditional support are always remembered.

SYSTÈME DE DÉTECTION D'INTRUSION MULTI-LOCATAIRE ORIENTÉ ÉVÉNEMENT

Mohamed HAWEDI

RÉSUMÉ

En raison de ses services créatifs offerts sur demande avec un bon rapport coût-efficacité et fiabilité, le « cloud computing » a été utilisé par des multi-locataires. Ces locataires se partagent, en effet, les ressources du Cloud, par exemple l'IaaS, le PaaS, le SaaS. Bien que ces services soient facilement fournis aux locataires à la demande avec des investissements d'infrastructure mineurs, ils sont considérablement exposés aux tentatives d'intrusion parce que les services sont offerts sous l'administration d'une supervision diverse sur Internet. Il est donc nécessaire d'avoir un mécanisme de sécurité capable de protéger les Cloud multi-locataire. Dans ce contexte, le fournisseur de Cloud s'est sérieusement efforcé de fournir des mécanismes de sécurité capables de protéger les ressources des locataires. Cependant, avec toute la sécurité offerte, malheureusement, les exigences des locataires ont été négligées. À partir de ces prémisses, nous proposons dans cette thèse un nouveau mécanisme de sécurité appelé Système de détection d'intrusion multi-locataire qui cible principalement l'IaaS du Cloud. La solution proposée comprend : (1) fournir un système IDS optimisé qui, d'une part, permet aux locataires de sélectionner les services de sécurité qui répondent à leurs besoins et, d'autre part, s'adapte automatiquement aux changements (activation/désactivation des services de sécurité) qui se produisent dans les environnements des locataires. (2) fournir un système de détection d'intrusion d'anomalie capable de détecter les attaques intrusives et d'assurer l'extraction d'informations précieuses du réseau surveillé, qui seront utilisées pour générer automatiquement de nouvelles signatures personnalisées. (3) fournir un IDS d'anomalie en temps réel orienté événement capable de surmonter les limites de l'IDS traditionnel en permettant une mise à jour continue de l'IDS d'anomalie pour détecter de nouvelles menaces. Il permet aux locataires ayant les mêmes intérêts de partager les nouveaux classificateurs IDS générés, chose qui permet d'optimiser les ressources et d'améliorer l'efficacité.

Mots-clés: Multi-locataire, exigences des locataires, système de détection d'intrusion (IDS), optimisation de l'IDS, Orienté événement, génération de règles ou de signatures, IDS d'anomalie, IDS en temps réel.

EVENT-DRIVEN MULTI-TENANT INTRUSION DETECTION SYSTEM

Mohamed HAWEDI

ABSTRACT

Due to the creative services offered on-demand with cost-efficiency and reliability, cloud computing has been used by multi-tenant providers. The tenants share cloud resources, for instance, the IaaS, PaaS, and SaaS. Although these services are easily provided to tenants on-demand with minor infrastructural investment, they are significantly exposed to intrusion attempts because the services are offered under the administration of diverse supervision over the internet. Thus, there is a need to have a security mechanism that is able to protect multi-tenant clouds. With this context, a cloud provider has to provide security mechanisms capable of protecting the tenant's resources. However, with all existing security approaches, tenants' requirements have been largely neglected. From these premises, in this thesis, we propose a novel security mechanism named Multi-tenant Intrusion detection System that targets primarily the IaaS cloud. The proposed solution involves: (1) providing an optimized IDS system that first enables tenants to select the security services that meet their needs and second, that automatically adapts to the changes (activating/deactivating the security services) that occur in the tenants' environments. (2) providing an anomaly-intrusion detection system that is capable of detecting intrusive attacks and ensuring the extraction of valuable information from the monitored network that is used to automatically generate new customized signatures. (3) providing an event-driven real-time anomaly IDS that is able to overcome the limitation of the traditional IDS by enabling continuous update to the anomaly IDS to detect new threats. It enables tenants who have same interests to share new generated anomaly IDS classifiers, which they use to detect new threats.

Keywords: Multi-tenant, Tenant Requirements, Intrusion Detection System (IDS), Optimize IDS, Event-driven, Signatures or Rules generation, Anomaly IDS, Real-time IDS

TABLE OF CONTENTS

Page

INTRODUCTION 1
0.1 Overview 1
0.2 Problem Statement 4
0.3 Research Aims and Objectives 6
0.4 Research Scope 7
0.5 Methodology 7
0.6 Technical Contributions 10
0.7 Publications 11
0.8 Thesis Organization 12

CHAPTER 1 BACKGROUND AND LITERATURE REVIEW 13
1.1 Cloud Computing 13
1.1.1 Fundamental concept of Cloud computing 13
1.1.2 Definition of Cloud Computing 13
1.1.3 Related Technologies 13
1.1.4 The architecture of cloud computing 14
1.1.5 Cloud Computing Business Model 16
1.1.6 The deployment models of Cloud Computing: 17
1.1.7 Features of the Cloud Computing 18
1.1.8 Cloud Services Providers 19
1.2 Security Challenges 23
1.2.1 Availability 23
1.2.2 Confidentiality 24
1.2.3 Privacy 25
1.2.4 Data Integrity 25
1.2.5 Audit 26
1.3 Attacks Classification 26
1.4 Intrusion Detection System(IDS) 28
1.4.1 Types of intrusion detection system 28
1.4.2 Intrusion detection techniques 30
1.4.3 IDS Components and Architecture 31
1.4.4 IDS Signatures Rules 32
1.5 Cloud orchestration 33
1.6 Machine Learning Algorithms 33
1.7 Literature Review 34
1.7.1 Signature based Detection Approaches 35
1.7.2 Collaborative Approaches 38
1.7.3 Anomaly based Detection Approaches 42
1.7.4 Hybrid Detection Approaches 45

CHAPTER 2 ARTICLE 1: MULTI-TENANT INTRUSION DETECTION SYSTEM FOR PUBLIC CLOUD TENANTS(MTIDS) 49
2.1 Introduction 50
2.2 Motivation 53
2.3 Background and Related Work 56
2.3.1 Fundamental concept of Cloud computing 57
2.3.2 Intrusion detection system(IDS) 58
2.3.2.1 Intrusion detection techniques 60
2.3.3 Related Work 60
2.4 Cloud Tenants Requirements and Perspectives 65
2.5 Cloud Tenant Topology Scenarios 66
2.6 MTIDS Architecture 68
2.6.1 MTIDS Mechanism 73
2.6.2 OIDS Process Flows 75
2.6.2.1 Definition of sets 75
2.6.2.2 Definition of relations 76
2.6.3 Optimized IDS (OIDS)algorithms 78
2.7 Implementation and evaluation results 81
2.7.1 Network Architecture Setup 82
2.7.2 Workload 84
2.7.3 Finding 86
2.8 Discussion and Future Work 90
2.9 Conclusion 92

CHAPTER 3 ARTICLE 2: MULTI-TENANT ANOMALY INTRUSION DETECTION SYSTEM(MAIDS) 95
3.1 Introduction 95
3.2 Related Work 99
3.3 Cloud Tenant Requirements 102
3.4 MAIDS Architecture 104
3.4.1 The MAIDS Mechanism 110
3.5 Implementation and Evaluation Results 112
3.5.1 Anomaly IDSs Training and Validation 112
3.5.2 Network Setup Scenarios 114
3.5.3 Finding 118
3.5.4 Discussion 124
3.6 Conclusion and Future Work 125

CHAPTER 4 ARTICLE 3: COLLABORATIVE REAL-TIME INTRUSION DETECTION SYSTEM (CRIDS) 127
4.1 Introduction 127
4.2 Related Work 131
4.2.1 Anomaly Based Detection Approaches 131

4.2.2 A hybrid IDS Approaches (Anomaly-Based & Signatures-Based Detection) 132
4.2.3 Collaborative IDS Approaches 133
4.3 Cloud Tenant Requirements 135
4.4 Proposed Approach 137
4.4.1 CRIDS Mechanism 142
4.5 System Evaluation 143
4.5.1 Environment setup 144
4.5.2 Data-Set Creation and Labeling 145
4.5.2.1 Labeling Data Set Traffic 146
4.5.3 Performance Metrics 148
4.5.4 Experimental Results 149
4.5.4.1 Experiments1 : Evaluate the performance of CRIDS and then compare the performance with that of an anomaly IDS 149
4.5.4.2 Experiments2: Measuring the Latency 153
4.6 Discussion 153
4.7 Conclusion and Future work 154

CONCLUSION AND RECOMMENDATIONS 155

BIBLIOGRAPHY 158

LIST OF TABLES

Page

Table 2.1 Ex.MTIDS Activation/Deactivation Policies 74

Table 2.2 Ex.Security Services 75

Table 2.3 Description of Virtual instances deployed on AWS..... 84

Table 2.4 Rules Sets 84

Table 3.1 Broker Policy..... 109

Table 3.2 Description of the Deployed AWS instances 111

Table 3.3 (CIDDS Set Attributes) Ring *et al.* (2018) 114

Table 3.4 Traffic Time 119

Table 4.1 Broker Policy to achieve sharing interests between tenants..... 141

Table 4.2 Data sets Traffic Type..... 145

LIST OF FIGURES

Page

Figure 0.1 Multi-tenant Intrusion Detection System(MTIDS) Architecture 8

Figure 0.2 MAIDS Architecture 10

Figure 0.3 CRIDS Architecture 11

Figure 1.1 Cloud Computing Architecture Overview.(Zhang *et al.*, 2010) 15

Figure 1.2 GrepTheWeb on AWS 20

Figure 1.3 OpenStack Architecture. 21

Figure 1.4 Mosaic API 22

Figure 1.5 Cloud Computing Models 30

Figure 1.6 IDS Components and Architecture 31

Figure 2.1 AWS instance cost 54

Figure 2.2 AWS Access Control List Example 57

Figure 2.3 Cloud computing Models 58

Figure 2.4 Cloud Scenarios (AWS based vision) 67

Figure 2.5 Multi-tenant Intrusion Detection System(MTIDS) architecture 69

Figure 2.6 Optimized IDS Controller Process 69

Figure 2.7 MTIDS Mechanism 72

Figure 2.8 Network Architecture 82

Figure 2.9 Network Architecture 86

Figure 2.10 Compares CPU Consumption between Snort vs OIDS 87

Figure 2.11 Compares CPU Consumption between Snort vs OIDS 88

Figure 2.12 CPU Consumption of the OIDS vs Snort 91

Figure 3.1 Cloud Tenant Security Preferences 102

Figure 3.2 MAIDS Architecture 104

Figure 3.3 Broker Mechanism 110

Figure 3.4 The MAIDS Mechanism 111

Figure 3.5 Network Architecture Set-up Scenario-1 113

Figure 3.6 Network Architecture Setup Scenario-2 115

Figure 3.7 Comparison of latency 1 subscriber and 1 publishers 121

Figure 3.8 Comparison of latency 1 subscriber and 3 publishers 122

Figure 3.9 Broker CPU Consumption 123

Figure 3.10 Broker Latency Performance 123

Figure 4.1 Tenant Security Requirement 136

Figure 4.2 The Architecture of CRIDS 138

Figure 4.3 DC Process in collaboration with Data Transmitter 139

Figure 4.4 Classifiers Model Broker Mechanism 141

Figure 4.5 The mechanism of the CRIDS 142

Figure 4.6 Network Architecture Set up 143

Figure 4.7 Comparison of *Accuracy* between CCRIDS based (SVM, DT, and RF) and traditional anomaly IDS 150

Figure 4.8 Comparison of *Recall* between CCRIDS based (SVM, DT, and RF) and traditional anomaly IDS 151

Figure 4.9 Comparison of Latency of CRIDS Models 152

LIST OF ALGORITHMS

Page

Algorithm 2.1 Tenant-Security-Requirements MSR $< Key, Value >$, SRQt) 78

Algorithm 2.2 ActivateAdaptationR (MSR, SRQt,SRt)..... 79

Algorithm 2.3 DeactivateAdaptationR (MSR, SRQt,SRt, ReqPriod) 79

LIST OF ABREVIATIONS

ETS	École de Technologie Supérieure
ASC	Agence Spatiale Canadienne
IDS	Intrusion Detection System
VMM	Virtual Machine Monitor
VM	Virtual Machine
NIDS	Network Intrusion Detection System
HIDS	Host-based Intrusion Detection System
AWS	Amazon Web Services
VPC	Virtual Private Cloud
TC	Traffic Classifier
ML	Machine Learning
NN	Neural Network
DOS	Denial-of-service
DDOS	Distributed Denial-of-service
CIDDS	Coburg Intrusion Detection Data Sets
CSP	Cloud Service Providers
DT	Decision Tree
RF	Random Forest
SVM	Support Vector Machine

MTIDS	Multi-tenant Intrusion Detection System
OIDSC	Optimized IDS Controller
OIDS	Optimized Signature IDS
TSRS	Tenant Security Requirements Supervisor
OAE	Optimized Analyzer Engine
AID	Anomaly IDS
NBA	Network Behavior Analyzer
OIDS	Optimized Intrusion Detection System
MAIDS	Multi-tenant Anomaly-based Intrusion Detection System
RP	Rule Publisher
RG	Rule Generator
TC	Traffic Classifier
RH	Rule Handler

INTRODUCTION

0.1 Overview

Cloud computing has emerged as the technology that enables access to computing resources such as networks, servers, storage, applications and service on-demand. Cloud computing changes the perspective toward inventing, deploying, scaling, updating, maintaining, and paying for information technology (IT) services. As a result, it has grown rapidly along with the trend of IT services. Cloud services are elastically offered over the Internet, enabling tenant to comfortably perform their daily jobs wherever they are. Tenant is a term that is commonly used to denote a client that uses a particular service from a cloud computing environment to satisfy an information technology (IT) need. The client might be an individual, an organization, or a business unit (Magar, 2012). With this regard, multi-tenancy concerns the concept of sharing or using the same set of resources by multiple clients. For instance, different companies use Cumulus to host a number of instances for deploying a number of customer-facing applications. In this case, Cumulus is considered a multi-tenant environment (Magar, 2012). These companies might have different units or branches (e.g., sales, and marketing business units) where they provide different services; hence, deploying different IaaS topologies.

Cloud computing allows tenants to decrease the cost, through pay-per-us as well as raises efficiency by offering different ways of consuming services by utilizing infrastructures of computing in a useful supplementary manner and a business paradigm for marketing the computing resources (Ficco *et al.*, 2012). In other words, the cloud computing model infrastructure-as-a-Service (IaaS) has become an acceptable solution that allows tenants to rent their hardware in the form of virtual machines (VMs)(Almorsy *et al.*, 2011). Thus, in the cloud, tenants are not required to purchase physical hardware, but only pay for use of the services.

Despite the significant advantages of cloud computing, security is a big challenge due to the distributed nature of the cloud computing environments. In addition, cloud resources become more vulnerable as they are presented over Internet, which, unfortunately makes them an attractive target for abuse; and thus an area of vulnerability (Zaman, 2009). Simultaneously, the tools used by the intruders or exploiters have been substantially improved along with advances in information technology.

This raised many questions for organizations regarding the safety of information when their data is transferred to the cloud.

Tenants need an adequate security safeguard to satisfy their needs or requirements. From a tenant's perspective, the security offered by the cloud provider is not well-trusted as the same hardware and hypervisor software are shared by all the VMs, which opens up the possibility for a compromised and exploited VM to compromise the other hosted VMs or even the hypervisor Ibrahim *et al.* (2011). In addition, tenants are not well-informed on the actual security protection level assured by the cloud provider to the hosted VMs. Also, since price is considered as an important aspect for choosing a security service provider, cloud security providers need to provide a security mechanism that is capable of delivering better security along with the cost reduction feature.

The cloud providers themselves do not have information about the contents of the VMs, because those VMs are run, controlled, and owned by the tenant. In simple terms, because cloud providers are not conscious of the architecture of the hosted services, they are not able to provide effective security controls. Therefore, from providers' perspectives, the hosted VMs cannot be trusted to host their supported security software, since the VMs can be compromised and exploited.

Collaboration, which is a way of allowing cloud providers to communicate with their tenants, is imperative to build trust as well as to improve the security. Collaboration allows both cloud providers and tenants to exchange knowledge regarding security, which will result in reducing the impact of security threats. Moreover, collaboration enhances the capability of the provider to meet the security requirements of their tenants efficiently. Thus, a new framework aiming at enhancing cloud computing security by taking into consideration the perspectives of both the cloud tenants and cloud providers is required.

Information security in the cloud plays a vital role in attracting more and more tenants to migrate their resources to the cloud as it is mainly targeted at preventing unauthorized users from accessing, using, disclosing, destructing, modifying or disrupting companies' data.

Because of this, intrusion detection systems (IDS) has been proposed, developed and used by both academic researchers and industrial groups to monitor networks and to raise alarms over suspicious activities; hence, to combat the rising number of attacks and to safeguard sensitive information. An IDS collects and analyzes incoming/out-going traffic within a host or a network to identify potential security breaches that can include attacks from outside/inside an organization. The IDS is designed to provide a defense layer against illegitimate users by sensing attacks and raising alerts. IDS-based systems have become essential security mechanisms for securing cloud computing environments since it is not possible to prevent all cyber-attacks (Tan *et al.*, 2014).

Although several innovative approaches and new models based on IDS have been proposed recently; they are still unable to provide an appropriate security mechanism that is able to overcome many shortcomings including: meeting the tenant's security requirements, reducing the cost, generating customized detection signatures, and enabling advance detection by sharing signatures between tenants based on their preferences. Furthermore, cloud

tenants want to have an optimized security arrangement that meets their requirements as they deploy or have different architectures on top of the cloud.

In this thesis, we aim to overcome the aforementioned shortcomings by developing a multi-tenant intrusion-detection framework that enables cloud tenants to exchange relevant information based on their interests by sharing their findings in real-time. Consequently, they can adapt their respective infrastructures and mechanisms to better reach their security objectives. These extensions are designed and implemented as collaboration functionality that will be integrated into the selected framework.

0.2 Problem Statement

Cloud providers (CPs) fail to deliver a high-quality security control as they are not aware of the architecture of the hosted service (Almorsy *et al.*, 2016). Since CPs provide their services to different tenants, they are faced with a lot of changes to their security requirements. The communication between CPs and tenants should be established based on negotiation and agreement before applying any security properties. This communication is a major challenge as there is no standard regulation in terms of the notations for security specifications that can be employed by both CPs and tenants to represent their offered or required security properties.

Basically, the security provided by cloud providers is very limited to securing cloud infrastructure and cloud platform services including the physical security of the data-center, network infrastructure, virtualization platform and infrastructure (Demchenko *et al.*, 2017). As an example, Amazon Web Service (AWS), an IaaS provider, states that tenants are responsible for securing their VMs as they are allowed to run any operating system (OS) or modify its services (Shawish & Salama, 2014). In the AWS public cloud, tenants are responsible for securing amazon machine instances (AMI), the OS,

applications, and data including data in motion, data at rest, and data stores as well as credentials, policies and configuration. Additionally, the tenant is specifically responsible for complying with the acceptable use policy (AUP), ensuring the correct use of the cloud platform, updating security, and patching the guest OS and installed applications (Demchenko *et al.*, 2017).

This raises many questions around the security provided by CPs. One critical issue is when an insider attacks occur; for example, when a cloud client abuses the cloud computing power and storage capacity or attempts to attack outside the cloud. According to (Patel *et al.*, 2013), Amazon Elastic Compute (EC2) was used by an attacker — an AWS customer — to attack Sony's online entertainment systems. Millions of customer accounts were compromised, which is considered as the biggest breach of data in the United States.

Meeting the tenant's security requirements is a serious impediment for CPs. As mentioned earlier, the security mechanisms offered by CPs do not take into consideration the different needs of the tenants as they provide the same security mechanism for all the tenants. For instance, some providers use a firewall to block external attacks while others use Access Control Lists (ACLs) on gateway routers (Adil & Ijaz, 2015). In fact, each tenant needs security policies based on his/her requirement. Thus, a network security system is individually needed for each tenant which secures them based on their needs. Another critical issue is cost management in terms of the security offered to the tenant. In some cases, the requested security may not be used for a period of time. As an example, let us assume that a tenant requests some protection against DoS, DDOS, and SQL. After a period of time, this tenant has changed his network topology by terminating the database server or it has been discovered that the traffic related to the database was not received. Therefore, based on this situation, the SQL security mechanism should be

temporally suspended to reduce resource consumption costs. Tenants impose their own special requirements to protect their VMs.

0.3 Research Aims and Objectives

This research is aimed toward proposing some security services in IaaS cloud. In particularly, it provides an adaptive intrusion-detection system for cloud tenants. The main aim of this study is to design and build a multi-tenant intrusion-detection-system framework that reduces overhead cost and provides a robust defense mechanism. The proposed work can cope with the dynamic nature of the cloud networks where information and the data within them are continuously changing. This enables an accurate and prompt intrusion detection. The proposed research will have a significant influence on the security of cloud computing industry through which we can expect increasing trust and paving the way for attracting more organization to migrate their resources to the cloud.

To summarize, we seek to meet the following sub-objectives:

- Present and implement a multi-tenant IDS as a service (MTIDS), which is an efficient framework that is aimed at reducing cost and meeting the tenants' requirements. The MTIDS should enable a higher-precision detection consensus by supplying advanced traffic analysis. Based on this analysis, a decision is taken to provide the desired security. Hence, tenants can make substantial gains in terms of efficiency, cost reduction, and security as it offers a flexible and adaptive security mechanism.

- Present an anomaly IDS that is trained based on the tenant's security requirements. For instance, a tenant who is interested in DOS protection will get a DOS anomaly-IDS model while a tenant that is interested in, for example, SQL injection protection, will

get a trained a SQL anomaly-IDS model. A tenant can have an anomaly IDS that has been trained for all attacks too.

- Extend traditional IDS with new rules that play a vital role in increasing the detection of the attacks that cause damage to the tenant's resources. Hence, automatically generate a customized set of rules that can significantly improve the efficiency of the IDS and ensure the extraction of valuable information from the monitored network.

- Enable cloud entities — which include tenants, CPs, and others — to work in collaborative manners and exchange relevant information which can be new signatures or classification models based on their shared preferences. This enables advance attack detection; it also reduces cost resulting from training the classification model and generate new rules.

0.4 Research Scope

This research focuses on designing a multi-tenant intrusion detection system that provides a low-cost SaaS framework for cloud tenants to deploy in order to protect their own cloud infrastructure. The multi-tenant intrusion detection system is an hybrid-based IDS (rule and anomaly-based) network intrusion detection system that offers appropriate security features for cloud tenants. It proposes to monitor the activities of tenants, capture abnormal activities, examine them to evaluate the probability of being an intrusion, and maintains the consistency of the security strategy. Furthermore, it is enabling a higher-precision detection consensus by supplying advanced traffic analysis. Based on this analysis, a decision is taken to provide the desired security.

0.5 Methodology

To achieve our objectives, we use the following methodology:

(a) Study the main solutions of infrastructure-as-a-service and the associated security approaches. We first identified the enforcement limitations of the existing security mechanisms managed by the providers and those managed by the tenants. We started with a comparative study of the existing multi-tenant IDS to identify their strength and weaknesses.

(b) Develop a security-as-a-service solution that targets cloud tenants. The proposed security service offered to cloud tenants and service providers and their infrastructure to restrain security intrusions. Throughout this work, various research questions have arisen and hence several objectives had to be set, which are highlighted throughout the articles presented in the following chapters:

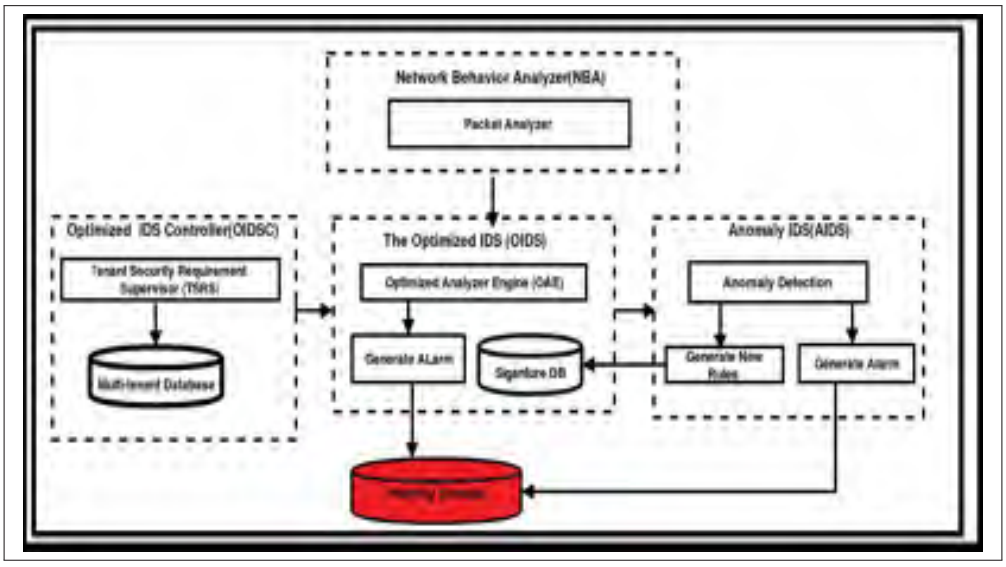


Figure 0.1 Multi-tenant
Intrusion Detection System(MTIDS) Architecture

The aim of the first article (Chapter 2) is to provide a flexible, on-demand, scalable, and pay-as-you-go multi-tenant intrusion detection system as a service that targets the security of the public cloud. Further, it is designed to deliver appropriate

and optimized security taking into consideration the tenants' needs in terms of security service requirements and budget. In this regard, we introduce Multi-tenant Intrusion detection System for Public cloud (MTIDS) as shown in Figure 0.1. It is proposed to monitor the activities of tenants, capture abnormal activities, examine them to evaluate the probability of being an intrusion, and maintain the consistency of the security strategy. The security strategy would be automatically changed based on the analyzed activities. The MTIDS would be able to activate/deactivate a set of rules, generate new rules, and automatically update the set of rules. Such services are regularly presented within a service-level agreement (SLA) context, which is aimed at ensuring the requested service quality.

(c) Developing security mechanisms allowing cloud tenants to detect intrusive attacks and that can automatically generate new signatures from the attacked system. Therefore in the second article (Chapter 3), we aim to answer the important research question that arises regarding the limitation of the signature IDS in detecting attacks; and updating the tenants with customs signatures. Accordingly, we propose Multi-tenant Anomaly Intrusion Detection System (MAIDS) as shown in Figure 0.2 with the following sub-objectives: Monitoring the tenant system activities and classifying them as either normal or abnormal; based on this, generating customs signatures.

(d) Developing security mechanism that is able to overcome the lack and the limitation of the traditional anomaly IDS by providing new anomaly-based IDS that is capable of adapting to changes occurring in network intrusion characteristics and patterns. Therefore in the third article (Chapter 4), we introduce a real-time anomaly-based intrusion detection, designed as a micro-services framework as shown in Fig 0.3. The aim of this work is to constantly train the classification model to overcome the mentioned problems and to detect the anomalies. It provides a light-weight,

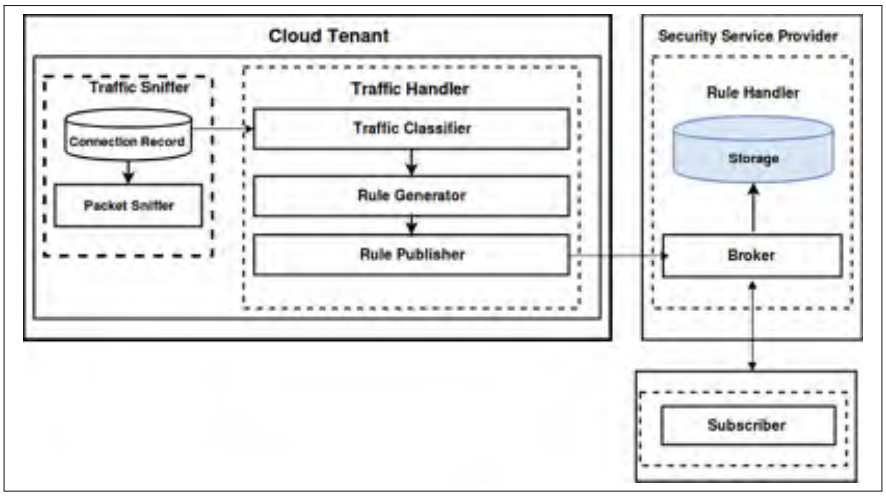


Figure 0.2 MAIDS Architecture

portable, reproducible and declarative security framework that is able to share more system resources. This results in cost reduction. The framework also minimizes the time and effort related to framework management. In addition to recognizing unseen attacks, the framework allows the generation of new classification models. These classification models will be shared between tenants based on their interests.

0.6 Technical Contributions

Through this thesis, we offer the following contributions:

- Adaptable and optimized intrusion detection framework for multi-tenant public cloud.
- Cost-effective anomaly intrusion-detection system that targets multi-tenant cloud providers.
- Safeguard security mechanism offering real-time intrusion-detection and enabling exchanging of relevant information.

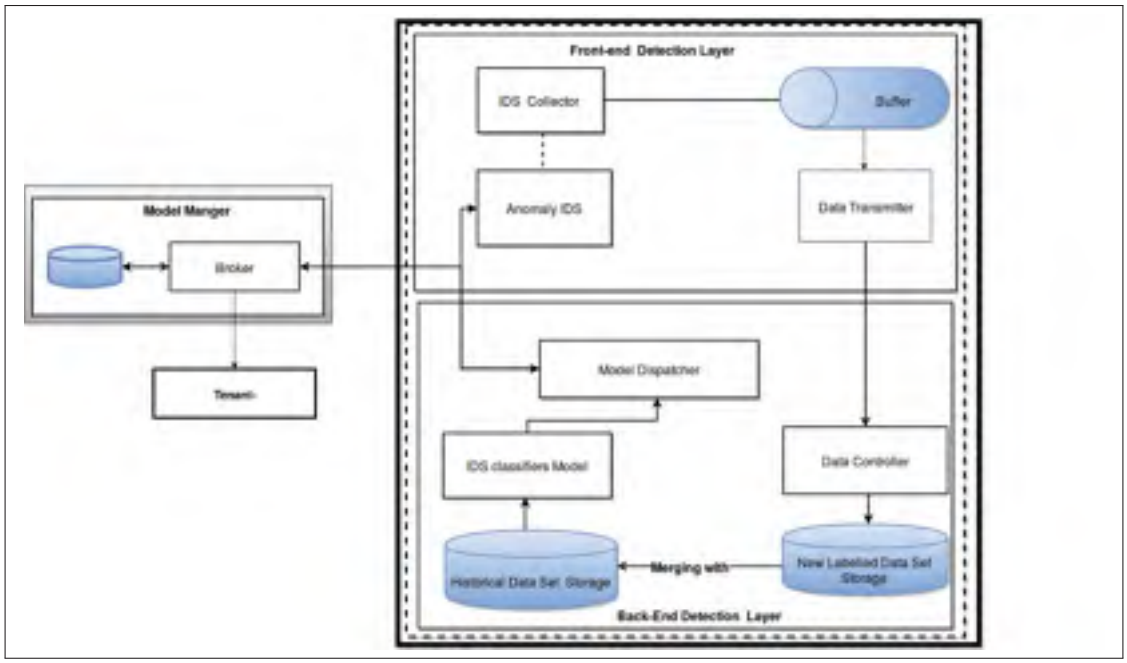


Figure 0.3 CRIDS Architecture

0.7 Publications

- Hawedi Mohamed, Chamseddine Talhi, and Hanifa Boucheneb. "Security as a service for public cloud tenants (SaaS)." Procedia computer science 130 (2018): 1025-1030.

- Hawedi Mohamed, Chamseddine Talhi, and Hanifa Boucheneb. "Multi-tenant intrusion detection system for public cloud (MTIDS)." The Journal of Supercomputing 74.10 (2018): 5199-5230.

- Hawedi Mohamed, Abdi Ramy, Chamseddine Talhi, and Hanifa Boucheneb. " Multi-tenant Anomaly Intrusion Detection System." Journal of Computers & Security.: Under Review.

- Hawedi, Mohamed, Abdi Ramy, Chamseddine Talhi, and Hanifa Boucheneb. "Collaborative Real time Intrusion Detection System(CRIDS) ." The Journal of Supercomputing.: Under Review.

0.8 Thesis Organization

Since this work is articles-based, we begin by discussing the general background and literature review. Thereafter, we provide details on each of our publications in dedicated chapters. Finally, in the last part, we conclude the thesis and draw some future directions out of the remaining open research questions.

CHAPTER 1

BACKGROUND AND LITERATURE REVIEW

1.1 Cloud Computing

1.1.1 Fundamental concept of Cloud computing

Cloud computing can be defined as a way of storing data or information permanently on servers over the internet and temporarily caching them on the client's side via laptops, sensors, computers, etc (Shawish & Salama, 2014).

1.1.2 Definition of Cloud Computing

According to the U.S. National Institute of Standards and Technology (NIST)," cloud computing can be defined as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011). As an example, users are able to use an application that runs on a remote data-center, which is named *cloud*, instead of installing the application on their system as is usually the case with the traditional technology.

1.1.3 Related Technologies

Cloud computing has not been created from scratch. There are many technologies that have paved the way to its development, and they have shared formal aspects (Zhang *et al.*, 2010):

- Utility Computing: it represents the provision of resources model upon request and charging the customers based on the consumption and not on a fixed price. With this

regard, cloud computing can be recognized as a implementation of utility computing as it has entirely adopt a scheme , utility-based pricing, for economic intents. In the cloud, service providers can truly scale up the utilization of the resource and reduce their operating costs through on-demand resource provisioning and utility-based pricing features.

- Virtualization: It plays an essential role in terms of designing the cloud (Modi & Acha, 2017). This is a technology that provides an abstraction of the physical hardware and virtualized resource details for high-level applications (Zhang *et al.*, 2010); and it is one of the most significant technologies utilized in IaaS. Virtualization increases efficiency in terms of performance, maintenance, and cost reduction for the computing services provided to users. This technology is used by different Cloud Service Providers **(CSP)** such as Amazon and Microsoft. In cloud computing, hardware and platform level resources are provided as services on-demand (Zhang *et al.*, 2010).

- Autonomic Computing In 2001, IMB invented autonomic computing. its main goal is building computing systems that are able to provide self-management include responding to internal/external surveillance without human intervention).Autonomic computing is developed to tackle the complexity of management issues of today's computer systems.

1.1.4 The architecture of cloud computing

This subsection designates three elements, as shown in Figure 1.1: the cloud computing layered models, business, as well as the deployment model of cloud computing. (Zhang *et al.*, 2010).

- Cloud Computing Layered Model

    (i) The hardware layer: it aims at managing the cloud physical resources: physical servers, routers, switches, power and cooling systems. Typically this layer is

Figure 1.1 Cloud Computing Architecture Overview.(Zhang *et al.*, 2010)

implemented in data centers, which is commonly comprised of thousands of servers that are systematized in racks and interconnected through switches, routers or other devices.

(ii) The infrastructure layer: it is also acknowledged as the virtualization layer. Through partitioning the physical resources using virtualization technologies, the infrastructure layer creates a pool of storage and computing resources, for instance, Xen (XEN, 2018), and VMware (VMWARE, 2018). The infrastructure layer is a key component of cloud computing, as many crucial features, such as dynamic resource allocation, are merely made obtainable through virtualization technologies.

(iii) The platform layer: it is built on top of the infrastructure layer. It contains both the operating systems (OS) and the application frameworks. It aims at reducing the burden of deploying applications precisely into containers or virtual machines. For instance, Google App Engine uses the platform layer to provide API support that intends to implement storage, database and business logic of idealistic web applications.

(iv) The application layer: the application layer contains the real cloud applications. Contrary to the conventional applications, cloud applications are able to

take advantages of the characteristic ,automatic scaling feature, for achieving availability, better performance, and lower operating charges.

1.1.5 Cloud Computing Business Model

Cloud computing uses a service-driven business paradigm. Straightforwardly, cloud computing provides the hardware and platform-level resources as services based on demand. Conceptually, every layer of the architecture explained in the previous section could be employed as a service. Conversely, a scientific point, three services are offered by clouds: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS):

1. Infrastructure as a Service(IaaS):

   The first model — Infrastructure as a service — is considered the foundation for the cloud computing environment and the tenants do not have first-hand access to it (Zargar *et al.*, 2011). IaaS is referring to the provisioning of infrastructural resources on- demand, normally with regard to virtual machines. The concept behind the IaaS is that servers, network, and storage are arranged to be available to end users as a service based on their needs. Basically, IaaS refers to the allocated infrastructural resources on demand in the form of VMs. IaaS provider refers to the cloud proprietor who offers IaaS. For instance, Amazon EC2 (Amazon, 2015), Microsoft Windows Azure(Azure, 2018).

2. Platform as a Service(PaaS) Platform as a Service(PaaS) refers to providing the platform layer resources that include OS and software frameworks (Rosado, 2012). Thus, the main objective of PaaS is to decrease the cost of the platform, such as OS or the development framework, that applications and services are developed or deployed (Oktay & Sahingoz, 2013). Azure Platform and Google App Engine are considered examples of PaaS (Alharkan & Martin, 2012).

3. Software as a Service(SaaS)

Software as a Service (SaaS) refers to offering applications on-demand on the internet such as SaaS providers (Zhang *et al.*, 2010). Here, the IaaS provider is responsible for infrastructure, platform, and service maintainability (Oktay & Sahingoz, 2013) while cloud users or customers are only allowed access to the application setting. In simple terms, users are only using applications that are running in the cloud (Almorsy *et al.*, 2011). The software is available in the cloud instead of a software installment in your computer; for instance, Google, Microsoft Word, etc. Whereas building a cloud-based information system involves matching the selection of the deployment model to the requirements; for example tanacy, security, performance, management and privacy (Oktay & Sahingoz, 2013).

1.1.6 The deployment models of Cloud Computing:

The deployment models of cloud computing are categorized with regard to the management and usage of the resources to private, community, public and hybrid cloud (Oktay & Sahingoz, 2013).

(a) **Private Cloud** is offered for the individual organization. The infrastructure could be situated in either the organization or in a third-party data center. In addition, they are responsible for the maintenance.

(b) **Community Cloud:** this kind of cloud is offered to more than one individual organization from a particular community that shares the same concerns. In the cloud community, the Infrastructure is typically placed in the diverse organization either in third party's data center or the community cloud. The data center organizations and the third party share responsibilities in terms of management and operational tasks.

(c) **Public Clouds:** public clouds are global as they are deployed over the Internet, by which they can serve users. A third party provides all the provisioning infrastructure, platform, and software. Users and providers provision and share

the management, operational and security requirements based on a Service Level Agreement (SLA).

(d) **Hybrid Clouds:** Hybrid Clouds are the combination of more than one cloud deployment model among those listed above. In this type of cloud, the infrastructure, platform, and software have flexible and transportable features that enable them to switch between the deployment models in the hybrid architecture that is consistent with its requirements.

1.1.7 Features of the Cloud Computing

Many significant features are offered by the cloud computing model to attract more and more enterprises towards transferring their existing IT assets to the cloud. Indeed, end-users can benefit from one or more cloud features based on their application domain regardless of the cloud deployment model. These features are designed to achieve fast development of applications, self managing workload adjustments, and financial cost saving.

- Elastic Scalability Scalability and elasticity are core capabilities needed to handle the change in a service workload. In particular, scalability is the capability of a service to maintain a changeable workload by meeting the quality of service (QoS) requirements, probably through consuming a variable amount of underlying resources while elasticity is the capability of a service to rapidly provision and de-provision the underlying resources on the fly (Ferry *et al.*, 2014).

- High-Availability In cloud, tenants are able to initialize, relocate, and cancel large number of virtual resources based on their demand. This reinforces the need for having a stable and reachable services. One of the key feature of the cloud is ensuring accessibility even with failure of some of its assets. In some cases, there could be a fail in the individual services for one or more customer, but the system continues to function for other services or a different set of tenants.

- Utility-based Service Utility-based service has its most important advantage in terms of financial cost saving. Cloud offers pay-as-you-go subscription. This enable the cloud tenants to meet their demands in terms of getting the functionality on a powerful infrastructure and solely pay for what they consume.

1.1.8 Cloud Services Providers

This section lists some cloud providers including the private and public cloud and the existing cloud computing technologies.

1. Amazon Web Services (AWS) AWS presents a compute power, content delivery, storage, and other functionality that allow companies to deploy applications, and services cost-efficiently with elasticity, expandability, and dependability. AWS is a dynamic, rising business element in Amazon.com. AWS started proposing IT infrastructure services to companies in web services since 2006. Currently, it is universally recognized as cloud computing platform. Due to the elasticity of AWS, customers can design their application architecture as they deem fit. Figure 1.2 shows the architecture of AWS. The first component is *Amazon Elastic Compute Cloud* (Amazon EC2). It is a web service interface that enables customers to gain and form capability with a slight effort, and this web service offers innovative compute capability in the cloud. Second, *Amazon Simple Queue Service* (Amazon SQS) is a message queuing service that is rapid, reliable, resizable, and entirely manageable. Using SQS, customers are able to transfer any volume of data, at any throughput level, without dropping messages or demanding other services to ensure availability. Third, *Amazon Simple Storage Service* (Amazon S3) is created to make the developer satisfied with web-scale computing, and it is used as a storage for the Internet. A simple web services interface is offered by Amazon S3, which can store and retrieve any volume of data (Mathew, 2014).

2. OpenStack

Figure 1.2 GrepTheWeb on AWS

Zhang (Zhang *et al.*, 2013) states that *OpenStack* refers to an open-source IaaS projects provided and established via both NASA and *Rackspace*. Yadav (Yadav, 2013) defines *OpenStack* as a group of open-source software projects that enable developers and technologists of cloud computing to set up and operate their cloud compute and storage infrastructure. Nebula platform, which is created by NASA, and object storage component, which is originally the *Rackspace* cloud files platform, are the *OpenStack* computing components. As is demonstrated in figure 1.3, *OpenStack* contains three main components: Glance (image service component), Nova (computing component), and Swift (object storage component) (Zhang *et al.*, 2013).

3. Eucalyptus

EUCALYPTUS refers to an acronym for Elastic Utility Computing Architecture used for linking your Program to the useful system. It is an open-source software that the University of California, Santa Barbara established for cloud computing in order to employ Infrastructure as a Service. It is compatible with Amazon Web Service API for deploying on-premise private cloud. Eucalyptus consists of five high-level components. The first is *Cloud Controller* (CLC), which is the login point for the end user, administrator project developers and manager into the private cloud and its aims in run-virtualized resources. The second is *Cluster Controller* (CC), which

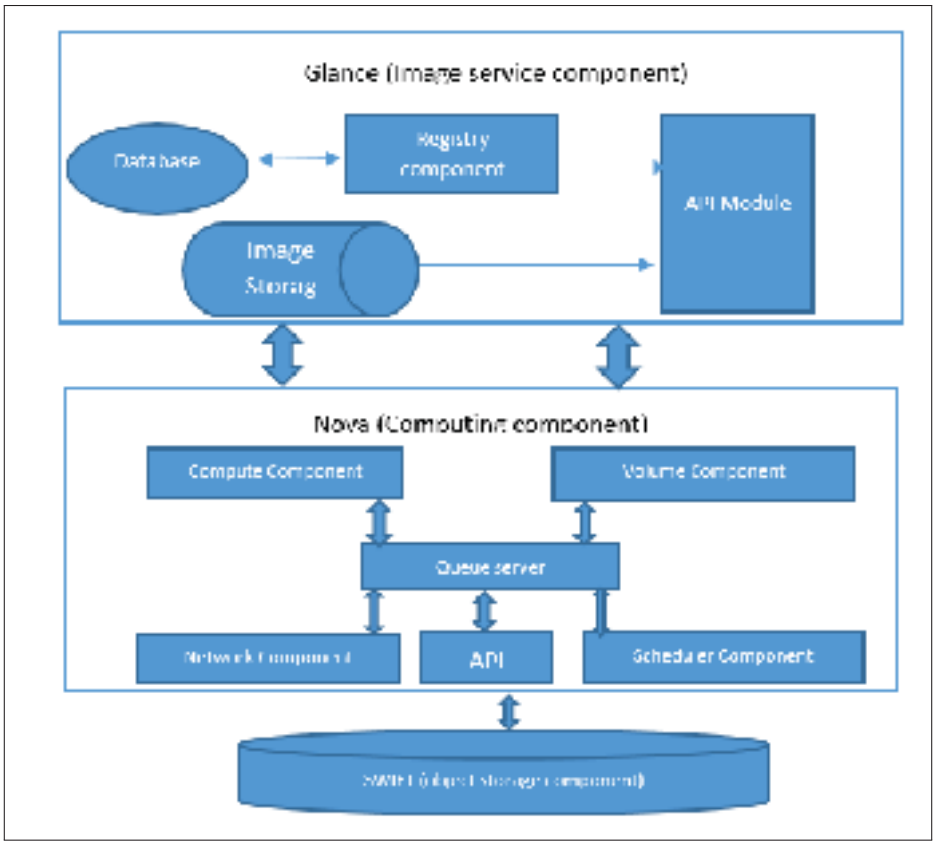


Figure 1.3 OpenStack Architecture.

is the component that is responsible for managing the network of virtual machines. The third is *Storage Controller* (SC) that offers block-level network storage, it even includes support for *Amazon Elastic Block Storage* (EBS) semantics. The fourth is *Node Controller* (NC), which is placed in each node for controlling the activities of the virtual machines, including the implementation, examination, and VM instances termination (Yadav, 2013).

4. OpenNebula

In 2005, Ignacio M. Liorente and Ruben S. Montero created Open Nebula as a research project and, in March 2008, released it to the public. Open Nebula could be used as a virtualization tool in order to run virtualized infrastructure in the data center or cluster, which is commonly signified by a private cloud. It can support hybrid cloud, merge local infrastructure with public cloud-based infrastructure, which enables

extremely scalable hosting environment. Also, it supports public cloud by providing cloud interfaces to expose its functionality to the virtual machines, and to storage as well as to the network management (Yadav, 2013).

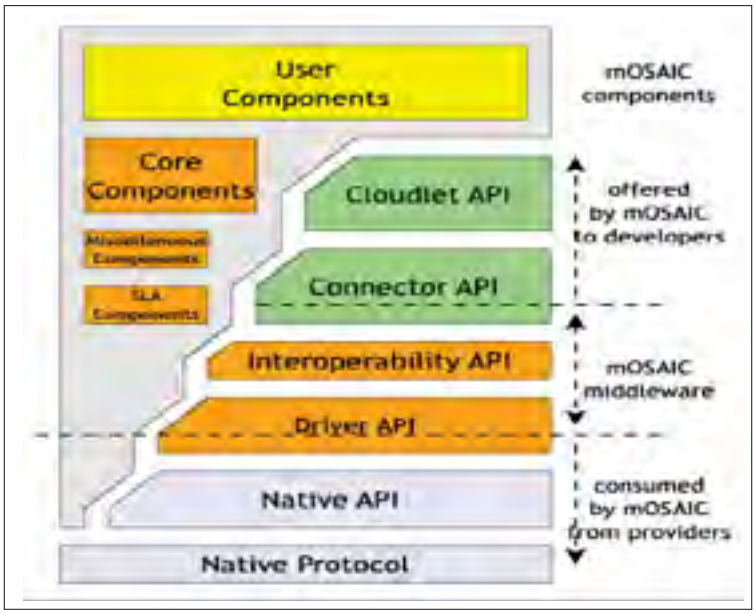


Figure 1.4 Mosaic API

5. mOSAIC

The *mOSAIC* solution is a consequence of a multi-national team effort as part of a funding contract with the European Commission in the frame of the FP7-ICT programmer. In September 2010, the execution of the project had begun and in March 2013, the complete software was released (Petcu *et al.*, 2013). The *mOSAIC* proposes to offer an open framework and APIs, which together aids cloud applications to be developed and deployed. Cloud application is designated as a set of components that have the ability to interact, as well as act as consumer cloud. As shown in Figure 1.4, the architecture of *mOSAIC* is comprised of two separate principal components, namely *Software Platform* and *Cloud Agency*. The software platform facilitates the implementation of applications that are created by employing *mOSAIC* APIs while *Cloud Agency* handles provisioning and brokering resources from a cloud provider's

federation. The *mOSAIC* application is a set of components that build blocks capable of connecting each other via communication resources of cloud (e.g., queues), for instance, AMQP, Amazon SQS (Ficco *et al.*, 2013).

1.2 Security Challenges

In spite of the advantages of cloud computing, organizations are still reluctant to embrace it because of security issues and challenges that are associated with cloud computing. A cloud computing environment offers its services including software as a service, platform as a service, and infrastructure as service on the Internet. Offering these services over the Internet potentially opens the door to hackers to attack the system. For instance, distributed denial of service attacks (Kulkarni *et al.*, 2012). Security is a major concern that is limiting the adoption of cloud in the industry. A cloud supplier offers the public cloud computing environments and it is responsible for ensuring that a resolution of cloud computing meets an organization's security and privacy needs (Ficco *et al.*, 2013). The following subsection highlights some cloud computing security issues and threats.

1.2.1 Availability

The objective of cloud computing availability is to guarantee the use of cloud computing resources including computing infrastructure and applications anytime and anywhere. Hardening and redundancy are two strategies used to improve the cloud system's availability or the availability of the applications hosted on it. Based on virtual machines, numerous cloud computing system sellers offer cloud infrastructure and platforms. Virtual machines can provide on-demand services — regarding consumers' individual resource requirements — to an enormous number of users (Ashktorab *et al.*, 2012). Several motives should be considered by the users with regard to the availability of their valued assets in the cloud. Firstly, most cloud providers do not own the computing and data-center infrastructure; they lease them from other providers. Hence, if the cloud infrastructure gets affected and becomes unavailable, most possibly, other providers will be affected as

well, thus making the resources unavailable to the wider users. The isolated traditional IT networks suffer less from this issue. Secondly, the possibility that a cloud provider can file for insolvency and close the business is another issue to consider; when this happens, the cloud resources become unavailable. Finally, in the cloud, the infrastructure and services are shared by multiple tenants; this makes the entire system more vulnerable, which, in turn, can affect the availability of the resources (Onwubiko, 2010). Denial of services is an example of a security issue regarding the availability in the cloud computing environment.

1.2.2 Confidentiality

Confidentiality can be defined as saving the sensitive information from unauthorized disclosure; this means that unauthorized users cannot access the sensitive information (Zaman, 2009). Cloud providers should ensure that users' data are always protected to ensure their confidentiality. The standard ISO/IEC 27018 (The International Organization for Standardization (ISO)) was developed to create a uniform, universal approach to data confidentiality. The aim is to protect the confidentiality of personal data that is stored in the cloud. Through ISO/IEC 27018, enterprises will be able to guarantee robust confidentiality protection across geographies and vertical manufacturing zones. ISO 27018 ensures the protection of privacy of customer information by several methods. First, it allows the customer to control their data by merely processing personally different information based on the instructions that the customer provided. Second, it ensures the transparency of the policies provided relating to the return, transmission, and removal of personal information that the customer stores in the data centers. Consequently, the tenants would be able to know what is going on with their data by informing them who has access to the data, and in the case of loss, disclosure or alteration of customer's information. Third, ISO 27018 is committed to not advertise the consumer's data. Finally, in the policy of ISO 27018, the consumer will be informed when the government accesses his/her data (Microsoft, 2015).

1.2.3 Privacy

Privacy is an essential issue for cloud computing from the legal obligation and users' concern perspectives. This need should be considered at every design stage. Reducing privacy risk and ensuring legal compliance are the major challenges that software engineers face in designing cloud services. More and more data is placed in the cloud for which the privacy is impacted. For instance, the best database buyers are using cloud support for their databases, such as Oracle using Amazon's cloud service platform EC2 to run and consequently more data is being transferred to the cloud. Concerns over privacy will keep growing as these databases often store secret and private information belonging to enterprises and persons (Pearson, 2009). Privacy concerns in the existing cloud computing offerings are observable and real.

Since users' personal data is placed in unknown locations — data centers — that are controlled by third party cloud providers, the cloud infrastructure may not be adjustable, and could transcend geographical boundaries that influence both statutory and regulatory requirements of the data being transmitted or stored. This concern is legitimate and faithful (Onwubiko, 2010).

1.2.4 Data Integrity

Data integrity in the cloud refers to the preservation of information integrity, which means unauthorized users are not allowed to modify or delete the data (Ashktorab *et al.*, 2012). Protecting data integrity should be considered when outsourcing data services from cloud providers. Outsourcing data services seems attractive from an economical point of view, especially, in the long term and for large-scale storage, it does not assure data integrity and availability. Therefore, this issue should be suitably addressed to avoid impeding a successful cloud architecture deployment (Ren *et al.*, 2012) The technique commonly used for examining data integrity is digital signage. The regularly adopted distributed file systems normally split large data volumes precisely into a set of blocks with a default

size. A digital signature is attached to a block of the data when it is physically stored. Hence, the digital signature can be tested for data integrity, and can be used to recover from a fraud (Ashktorab *et al.*, 2012).

1.2.5 Audit

Audit refers to the observation of events in the cloud system. Audit-ability is a kind of an additional layer that might be added above the virtualized operation system, or virtualized application environment, which is hosted on the virtual machine for providing services to watch the activities occurred in the system. It is better to place an additional layer on virtual machines than on the applications or in the software themselves. By following such a procedure, more ability to watch the entire accesses will be available. Nevertheless, the monitoring should not be invasive; and it should be finite with regard to the cloud provider needs (Ashktorab *et al.*, 2012). With this regard, every transaction of data has to be recorded for ensuring data integrity in cloud computing. Therefore, cloud providers should implement internal monitoring controls specifically designed for an external audit process. Despite cloud computing environment presenting new challenges from both perspectives of an audit and agreement, the current solutions for outsourcing and audit might be sufficient (Ren *et al.*, 2012)

1.3 Attacks Classification

An intrusion can be defined as any set of activities attempts at compromising the confidentiality, integrity, or availability of the resources by obtaining privilege to access to confidential data. This is can be done by either exploiting vulnerabilities of the system to access sensitive data, or by relying on an authorized user to break or compromise the the system. In deed, the increasing skills of the intruders in regard to understanding of how systems work make intruders a professional at figuring the weaknesses of the systems and exploiting them to make system resources access-able. The intrusion patterns used by intruders are not easy to be traced and identified as they uses set of feints before

compromise or breach the target systems and rarely indulge in sudden bursts of suspicious or abnormal activity. In addition, another important activity for the intruders is covering their tracks so that their activity is difficult to be discovered on the penetrated system (Zaman, 2009).

Generally, attack are categoried into four main types :

- Probing: surveillance.
- U2Su/U2R: Unauthorized access to Local Super user root privileges.
- DoS: Denial of Service.
- R2L: Unauthorized access from a Remote machine

(i) Probing: it is a category of attack that is used to scan a network to collect information or to identify known vulnerabilities. The probes fall into different types including the some that abuse the system by taking advantages of the legitimate features and others that use social engineering techniques. IPsweep, Saint, and Satan are an example of this kind of attack.

(ii) DoS Attacks : it is a category of attack that overloaded the computing or memory resource and make them unable to handle legitimate requests, consequently denying authorized users to access to a resources or machines. it is a category of attack that overloaded the computing or memory resource and make them unable to handle legitimate requests, consequently denying authorized users to access to a resources or machines. DDoS, Pingflood, SYN flood are examples of DoS attack.

(iii) U2Su Attacks : User to root (U2Su) exploit. This category of attack begins to access to legitimate users accounts and then exploits a system vulnerabilities to obtain root access privilege . Buffer overflows is considered as the most common exploits in this kind of attack. Eject, Fdformat, Loadmodule, and Perl are examples of the U2Su Attacks.

(iv) R2L attacks : in this category of attack, the attackers attempts to exploits the system's vulnerabilities to illegally obtain local access as a user by sending packets over the network to a machine . FTP-write, Sendmail, and Xlock are examples of this category of attack.

1.4 Intrusion Detection System(IDS)

Intrusion detection refers to the process used for monitoring the events that occur in a computing system. It is used to monitor computers and networks and inspecting them for signs of the potential occurrence of accidents, which are violations or impending violation threats of computer security policies, appropriate-use policies, or standard security practices. An intrusion-detection system (IDS)can also be used to function as an Intrusion Detection and Prevention System **(IDPS)**. Intrusion prevention refers to the process used for performing intrusion detection and for attempting to stop the detected potential incidents (Scarfone & Mell, 2007). Intrusion detection systems (IDS) can be either software, hardware or a hybrid collection of both of them that gathers data from the protected system, and send alarms to the administrators in diverse ways either through logging, emailing or preventing the system against the detected intrusion. Though there are different kinds of IDS, host-based, network-based are the main types (Oktay & Sahingoz, 2013).

1.4.1 Types of intrusion detection system

- Network-based intrusion detection system (NlDS): NIDS is used for observing, examining and analyzing certain components for precise and predefined divisions of network traffic (Oktay & Sahingoz, 2013). NIDS observes network traffic for certain network segments or devices and inspects the activities of the network to detect any suspicious activities. It is usually located at the networks' borders, that is, close to firewalls or behind routers, virtual private network, remote access servers, and wireless networks (Scarfone & Mell, 2007). Network-based IDSs can overcome the weaknesses

of host-based IDSs. Configuring and managing every host is not required as a single IDS sensor in a network segment can be responsible for performing all the analyses. Furthermore, if one host in the network is attacked, IDS would not be affected as it is deployed outside the hosts. Finally, network-based IDS do not use the hosts' resources, as it utilizes private resources for its functionality (Zarrabi & Zarrabi, 2012). As an example of network-based IDSs in cloud, NIDS deployed on AWS monitors all VMs that belong to the virtual private cloud.

- Host-based intrusion detection system (HIDSs): Host-based intrusion detection system (HIDSs): HIDS is used for observing and analyzing the host system's characteristics once an incident happens such as suspicious activity like system calls, processor threads, entity and configuration access or modification. Usually, the HIDS is placed in the most important host that is used to store private and vital information. Also, it is possible for the host to perform some of the NIDSs' functions if it is placed in one host and configured to detect network behaviors (Oktay & Sahingoz, 2013). For example, in the AWS cloud (Amazon, 2015), host-based IDSs are deployed in each of the launched VMs. Host-based IDSs can use both system logs and operating system audit trails for system state monitoring such as detecting a particular program that accesses specific resources. Despite the fact that host-based IDSs provide many benefits, some drawbacks limited their adoption (Zarrabi & Zarrabi, 2012):

  - Information has to be configured and managed for every host monitored, which make managing host-based IDSs very hard.
  - Host-based IDs' information sources and analysis engines are placed in the same monitoring targeted virtual machines, which can cause them to be exposed or to be controlled or disabled by attacks.
  - Host-based IDSs influence performance cost on the monitored systems as they employ the computing resources of the hosts they are monitoring.

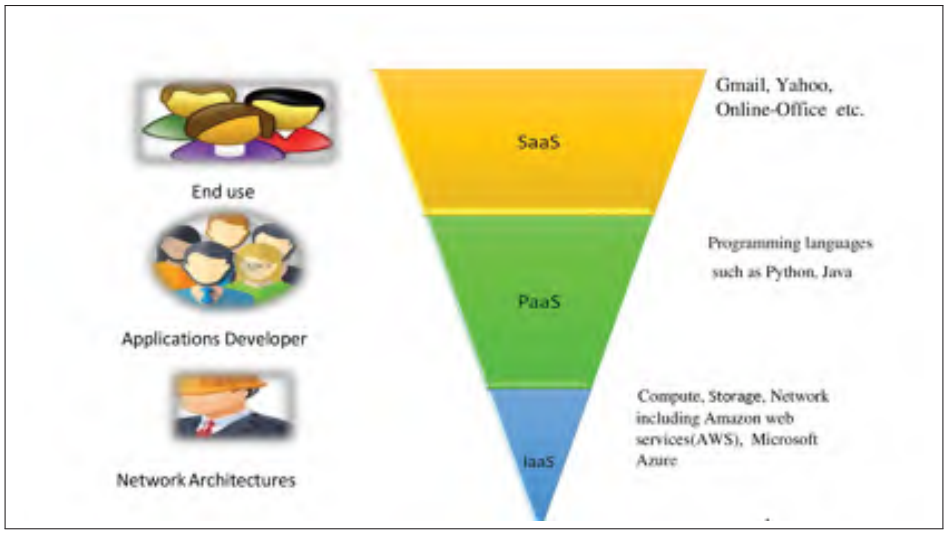


Figure 1.5 Cloud Computing Models

1.4.2 Intrusion detection techniques

Different techniques can be used to analyze and detect attacks. These techniques are categorized into various principal categories (Oktay & Sahingoz, 2013), as described in the following section:

- Signature-based IDSs: Signature-based techniques are also known as knowledge-based or misused-based techniques
  (Keegan *et al.*, 2016).
  Misuse detectors aim at analyzing system activities by searching for an event or a group of events that match the signature that identifies known attack types. Patterns that match known-attack types are referred to as signatures. Misuse detection is occasionally known as signature-based detection. This technique is used to determine intrusion attempts based on pre-defined rules against known attacks(Oktay & Sahingoz, 2013). Its main goal aims at detecting intrusion attempts by searching for the pattern of the known attacks. One key features of the Signature-Based is the ability of producing a low volume of false positives alerts.

- Anomaly-based IDSs: IDSs particularly focus on monitoring and analyzing system behavior. Anomaly detectors aim at identifying unusual and abnormal behavior in the host and network. They assume that abnormal activities are different from authorized activities. Hence, these activities can be detected using systems, which can distinguish normal from abnormal activities. Anomaly detectors learn themselves by constructing profiles representing the habitual behavior of users, hosts, and the network. The profiles are made from data gathered for a period of time and over a stage of normal operation. The anomaly detector function then collects data events and employs several measurements to determine when monitored activity moves away from the common habit (Zarrabi & Zarrabi, 2012). In contrast to the signature-based IDSs, anomaly-based IDSs can detect unknown attacks like, a new and evolving malware, which consumes computing resources through originating processes or multiple logs on the network (Petcu *et al.*, 2013).

- Stateful protocol: This technique uses the characteristics of the known protocols as a detection mechanism. Therefore, it's akin to the signature-based types as its intrusion detection with defined formations of protocols.

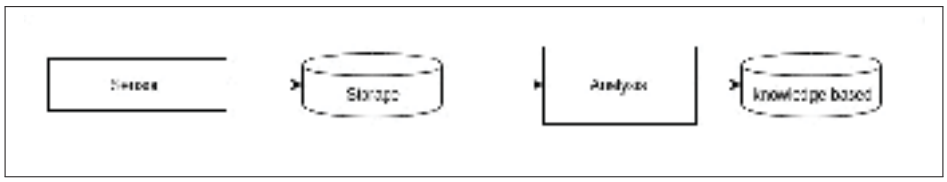


Figure 1.6 IDS Components and Architecture

1.4.3 IDS Components and Architecture

There are four main elements that all Intrusion detection systems share regardless of their nature. These components include the sensors or agent, storage, the analysis and the knowledge-based components as shown in Figure 1.6. The first element, that is, the sensor is designed to collect the network traffic and store them in the

storage component. In particular, sensors or agents are responsible for monitoring and analyzing activities. Sensors are usually used for IDSs that aim at monitoring networks, comprising network-based, wireless, and analysis technologies of network behavior. Storage is a central component that receives information from the sensors or agents and manages the received information. The analysis component analyzes all the gathered events stored in the storage component. The Knowledge-based encompasses some sets of signatures or rules that are used by the analysis component to identify or detect any suspicious activities.

1.4.4 IDS Signatures Rules

IDS rules used to define the patterns and criteria that are employed to examine potentially known malicious traffic on the network. The following Example 2 illustrates Snort IDS signature rules.

Example 1. *alert tcp EXTERNAL_NET 80 − > HOME_NET 7070 (msg:"ICMP test" sid:1000001; rev:1; classtype:icmp-event;)*

- This is explained below: the rule examines any traffic directed toward the network and it will generate an alert whenever the IDS detects traffic headed inbound from the outside to the network over the port (Source) 80. Basically, Snort rule contains two parts: namely *Rule header* and *Rule Options*. The syntax of the rule is explained as follows:

  - **Rule Header:** In the rule header `(alert tcp EXTERNAL_NET 80 -> HOME_NET 7070)`, `alert` represents the rule action to be taken by the IDS. The IDS will generate an alert if the condition is met; the string `EXTERNAL_NET` represents the source IP and `80` is the port number. All the external traffic from the source IP over port number 80 will be considered by the IDS. The arrow (`->`) indicates the direction from the source to the destination. The `HOME_NET 7070` refers to the destination IP (`HOME_NET`) and its port number (`7070`).

- **Rule Option:** In this example, the rule option is represented as `(msg:ICMP test; sid:1; rev:1; classtype:icmp-event;)`, where the message (`msg:`) is `(ICMP test)`, will be included with the generated alert. The string `sid:1;` represents the rule number or ID, which should be unique. The string `rev:1` gives the revision number. This option facilitates the maintenance of the rule. Lastly, the `classtype:(icmp-event)`, in this example, is used to classify the rule as an 'icmp-event'. This classification helps in the organization of the rules.

1.5 Cloud orchestration

Cloud orchestration refers to the automation that is involved with organizing and managing of complicated cloud-enabled software and services. Cloud orchestration is about automating the software configuration, coordination, management, and interactions within the cloud environments. Cloud application orchestration or tools and specifications are used to express the service topologies. Many tools — such as *CAMP*, *TOSCA*, *CloudFormation*, and *Heat* — can be used in a cloud environment to deal with cloud application orchestration (Katsaros *et al.*, 2014).

1.6 Machine Learning Algorithms

This section provides a general overview on the machine learning algorithms that are used to detect novel attacks that the signature-based IDS are unable to detect.

1. **Decision Tree (DT)**: DT is a type of supervised learning algorithm that significantly is used in data mining, more specifically in classification problems, as a predictive model that can be used for representing classifiers. In DT, data is represented in form of a hierarchical tree that involves a group of nodes that describes a problem with various solutions. DT encompasses of three elements: nodes, edges, and leaves. First, DT repetitively splits the training data set into subsets based on its attributes until the stopping conditions are satisfied. Since the decision nodes has the same

data set, it is possible to specify the attribute that best splits the data set into its classes. In fact, each decision tree node contains various edges whose main purpose is to specify the potential value ranges or values of the selected attributes on the node. The specification of the edges play a role in splitting the data node into subsets. a child node is created by the DT for each subset. The splitting process is repeated and it is terminated whenever the node satisfies the stopping rules or no future unique attributes can be determined. Finally, a node is labeled by DT and after labelling it called a leaf.

2. **Neural Network (NN)**: NN is a collection of connected units that follow a particular topology. Each neuron is described by a unit that contains an input and an output. Two neurons are connected such that the output of one of the neurons is connected to the input of the other. In a neural network, each connection has a weight correlated to it. The topology of the neural network, the training methodology for weight adjustment and the connections between the different neurons are the main aspects that define the kind of the corresponding neural network (Bouzida & Cuppens, 2006).

3. **Random Forest (RF):** RF is a supervised machine learning classification algorithm that is used to overcome problems related to classification and regression. It uses multiple decision-tree-learning models to get better predictive results. Simply, the RF model creates a forest with the number of decision trees to get the best solution.

1.7 Literature Review

Security in cloud computing environment is considered as the most significant challenge that makes many researchers propose security as a service for either cloud provider, tenant or service provider.

This section illustrates an overview literature review of the research topic in relation to the adoption of the traditional intrusion detection system (IDS) for enhancing security in cloud computing. Each work focused on various cloud service modules or various

objectives. Essentially, an intrusion detection system encompasses a verifying process that examines the entity's behavior looking for attack signatures that are precise patterns which indicate malicious or suspicious intent.

1.7.1 Signature based Detection Approaches

A lot of the proposed research concentrates on providing security as a service for cloud providers as part of their infrastructure which they can offer to tenants.

Alharkan and Martin (Alharkan & Martin, 2012) proposed a scalable intrusion-detection system as a service (IDSaaS), which can detect malicious or suspicious intent. IDSaaS is a network-based IDS that targets use to protect IaaS. It is built on top of the AWS cloud and is used to monitor and log suspicious activities of the network between tenant VMs. This framework uses the signature-based technique to determine the validity of events. Particularly, the traffic is captured and analyzed based on signatures that have been pre-defined and can be updated on a systematic basis. IDSaaS consists of two main components namely:*IDSaaS Manager, and IDS Core* . The IDSaaS Manager acts as a security administrator access point for performing various supervision tasks. While the IDS core is a network IDS deployed on the public subnet and acts as a goal keeper to the private VMs. It is mainly inspects all incoming traffic using the Intrusion Engine component.

(Nikolai & Wang, 2014) proposed a Hypervisor-based Cloud Intrusion Detection System (HCIDS) to detect denial of service attacks within a cloud environment. The main goal is to improve performance over data residing in a VM by taking advantages of the hyper-visor capabilities. In their work, they examine system metrics of the cloud instances directly from the Virtual Machine Monitor (VMM) or hyper-visor, which hosts the virtual machines to find any potential misuse patterns. And these metrics include, network data transmitted and received , and CPU utilization, etc. The framework consists of three high level components namely: a controller node, end point nodes, and a notification

service. firstly, the end point nodes component retrieves or gather data in near real-time intervals(each second) from the virtual machine hyper-visor and then dispatches the data to the to the controller node where it does real time analysis of the received data to confirm whether there exists an attack or not. if any potential attack is discovered, controller component sends an alert to a notification service.

Similarly, Patel, and Sonker (Patel & Sonker, 2016), proposed a rule-based NIDS designed to detect Dos attacks and to generate Port scan detection rules. The proposed approach enables safeguarding networks from any illegitimate access. In this rule-based detection system, network traffic which are passed over the network are captured and then inspected looking for any suspicious activities (intrusions). The system generates an alarm if any packet matches the signature.

Varadharajan and Tupakula (Varadharajan & Tupakula, 2014) have proposed safeguarded security as a service model based on the intrusion detection system that targets the infrastructure and is offered by the cloud provider to its tenants . The proposed service model provides baseline security to the cloud provider for protecting its cloud infrastructure. It also offers some flexibility: tenants can specify their security functionality that is tailored to their needs. The main aim of the approach is to protect a tenant from internal attacks by deploying the proposed mechanism on the hypervisor (that is, at the highest privilege level), making it possible to monitor the VMs. Their proposed approach's architecture contains different defense components:

The *Host-Based Security Tools* (HBST) acts as the primary defense layer; the layer allows tenants to run their own HBST on the VMs provided by the cloud provider. The *Service Provider Attack Detection* (SPAD) and the *Tenant Specific Attack Detection* (TSAD) components are the other important defense layer that are proposed to augment the first layer of defense. SPAD receives the tenant VM's traffic and then enforces the security baseline policies required by the cloud service providers. The tenant's virtual machine is isolated. An alert is generated if a tenant VM's traffic violates any of the security policies

in the SPAD. If a tenant requires more traffic investigation, TSAD is available on-demand as an additional defense layer. Whenever such a request is made, the traffic is forwarded by SPAD to TSAD; TSAD enforces the tenant's specific security policies on the received tenant's traffic. The security policies in TSAD are decided by the tenant at the time of registration. TSAD forwards the traffic to its destination if the traffic does not violate any of the security policies enforced by TSAD. This approach is effective: it drops errant traffic. However, due to a lack of a mechanism for regular updates of the rule database by the approach, the system may be subjected to kernel manipulation attacks.

Gul and Hussain (Gul & Hussain, 2011) introduced a distributed multi-threaded paradigm IDS that targets the security of cloud computing. The multi-threaded IDS model is designed to handle a huge flow of network traffic. The traffic is then analyzed and an alert is raised if there is any suspicious activity. This approach suggests that the raised alert should be sent to a third party. The third party is responsible for sending alert reports to a cloud user in case there is an intrusion attempt, as well as advice to the cloud provider regarding the occurred intrusion. It is composed of three components including: *Capture and Queuing, Analysis and Reporting.* The first component captures the traffic. Next is queuing and then the traffic is sent for analysis. A signature technique is used for detecting any suspicious activity. If the traffic violates the signature, an alert will be generated and sent to the report component. Hereafter, the Report component will send the generated report to the third party. Then, it will be sent to the cloud user and cloud provider.

A network intrusion detection system (NIDS) was presented by Gupta (Gupta & Kumar, 2017) aiming at securing Cloud resource against distributed denial of service attacks. The*Cloud Manger* is the main component where it is centrally controlled and manage the NDDS instance that deployed at the client environment.The cloud manager manages configuration files and shares the client VMs log files at the client VMs. Particularly, tenant VM profile is used for predicting the DDOS. In essence, The profile of the tenant is used for classifying DDoS the signatures of the attack as well as for performing back

off based detection of signatures. Various important processes phase were proposed in this approach. The first phase is initialization and rule update, which is created to set the threshold for the attack signature existing in a client virtual machines profile. In detection phase, packets are examined for detection attack that meet the signature. In Alert and Response Generation phase, alerts with will be send to the VM client if suspicious activities were detected.

1.7.2 Collaborative Approaches

Other researchers concentrate on deploying the intrusion detection system in a collaborative manner for detecting suspicious activities.

(Ficco *et al.*, 2013) proposed Intrusion-detection architecture. It is designed for detecting distributed attacks in cloud computing environments. Their approach is based on the observation of the cloud computing and aims at detecting when a computing resource has been compromised. It is capable of detecting distributed attacks. They developed a framework that is composed from three components: *probe*, *agent*, and *security engine*. A probe represents the component for detecting intrusions. It does this by monitoring some security configuration parameters at different cloud architectural levels. It can monitor the hypervisor, the entire infrastructure, the platform, and the applications. An agent is a piece of software that is responsible for forwarding the security-related data collected by probes to the security engine. In addition, agents perform a normalization process to enable different types of probes to generate events using a domain-specific language. A security engine correlates multiple streams of event data in real-time and decides whether the information received represents a potential attack pattern based on some specific correlation rules. In their approach, security engines are organized into an hierarchy using three architectural layers. The engines perform different correlation tasks at different levels of abstraction, forwarding the aggregated data to a higher layer. At the base layer, the security engines are responsible for correlating the unprocessed security data collected by probes that can be offered as a service by the cloud provider. The

service can be IDSs (e.g. Snort), log analyzers (for syslog); it could also be some specific security arrangement provided by the cloud platform. The probes can be configured to collect data tailored to the security requirements of the customer. At the highest level, the cloud provider is responsible for enabling additional IDSs and for attaching them to independent VMs or physical machines. For instance, in a network-based IDS, the provider can enable probes for monitoring the infrastructure and platform layers. In a host-based IDS, an IDS is attached to each physical machine; so a probe can be enabled at each host machine. Specific probes that collect complementary information — such as access to the hypervisor — can also be enabled in the specific cloud environment.

The work introduced by (Zargar *et al.*, 2011), proposed a framework (DCDIDP) which focuses on coping with an attack using intrusion detection and prevention systems (IDPSs). The proposed intrusion detection framework (DCDIDP) allows all cloud service providers to cooperate in a distributive way at different operational levels to reply to attacks and offer universal IDPSs. Global databases are shared among cloud providers on which they can detect sophisticated cooperative intrusion. DCDIDP encompasses of three level architectures including: network, host and global infrastructure. The Network and host architecture maintain local database of policy and rules and contribute to global database. The global database shares information regarding intrusions among different clouds. The main features of DCDIDP include being distributed (policies are distributed among hosts), collaborative (hosts collaborate with each other to stay synchronized for information sharing) and data driven (dynamic evaluation of rules and access list). DCDIDP can be implemented in IaaS, PaaS and SaaS and provides effective intrusion prevention.

Chi-Chun Lo et al (Lo *et al.*, 2010) proposed an intrusion-detection system for monitoring cloud computing regions. In their approach, IDSs are deployed in each cloud region to detect intrusion attempts. The IDSs send alerts to each other whenever an intrusion is detected. They also assess the reliability of the received alerts. In the approach, many components are involved in detecting an intrusion. Some of the components are: alert clustering, threshold check, intrusion response agent, intrusion blocking agent, intrusion

collaborative agent. Alert clustering module gather alerts generated in other regions; the collaborative agent in different regions exchange the alerts gathered by the alert clustering component. The riskiness of the alerts are estimated to determine whether they true or false alerts. If the alerts are considered true, a new blocking rule is added to the blocking table. This enables an early intrusion detection and provides a means for the IDSs deployed in the cloud computing region to resist an attack, even from a victim IDS. Each IDS has three modules. The first is the block module; the second is the communication module; and the third is the cooperation module. The block module is used to drop bad packets sent out from the source node. The communication module sends warning messages on some specific attack detected by the IDS to the other IDSs in the region. The cooperation module gathers alert messages, which could be either true or false alerts.

The authors of (Alruwaili & Gulliver, 2014) suggested the concept of a collaborative IDS for protecting the cloud infrastructure layer against known and unknown threats with a combination of a signature-based technique and an anomaly-based technique. Their proposed approach is for protecting service providers and their tenants against loss of services that are caused by known or unknown threats. They developed a framework that enables cloud service providers to improve the performance, reliability, accuracy of their intrusion detection and prevention system in the SaaS and PaaS service models. The framework can work in a global, collaborative manner to achieve a sophisticated monitoring of the cloud resources. The architecture of the framework is composed of several modules including:

Threat Detection Agent (TDA), *Intrusion Detection Sensor* (IDS), *Data Collector Agent* (DCA), *Data Inspection Analysis* (DIA), *Security and Network Management* (SNM), *Keep-Alive* (KA), *Cooperative Agent* (CA), and *Administrative Interface Console* (AIC). The sensor is an integral part of every component of the IaaS model; it aggregates the activity logs and can be placed in different layers. It can be placed in: 1) the network layer — where it will be responsible for monitoring the incoming, outgoing, and local

network activities; 2) the storage layer — where it will monitoring the file integrity and unauthorized access to files to provide protection; 3) the server layer — for monitoring CPU usage, process activities, memory, and input/output (I/O) utilization; and 4) the virtualization layer (hypervisor) — for monitoring configuration files and all process instances and the activities of the VMs. DCA receives all packets from the deployed IDSs in the IaaS components based on some predefined scheduling intervals. The aggregated data is then passed to DIA for inspection and auditing. DIA receives packets collected by DCA; processes them using the anomaly and signature databases. The combined analysis and inspection process is separated from the detection engine via some dedicated resources for improved performance. The results are passed to TDA for decision making. AIC is the interface that enables the CCIPS administrator to manage and access all the CCIPS components. AIC enables access to the activity monitors, system logs, and the threat patterns in the anomaly database. CA is responsible for the coordination and collaboration of the CCIPS signature and anomaly databases among all connected or participating CCIPSs.

(Roschke *et al.*, 2009) proposed virtual machine (VM) IDS architecture which is an extensible IDS management architecture that build based on a novel mechanism of event gatherer component. The main goal is preventing the VMs from being compromised. The architecture consists of several IDS sensors deployed on each VM as well as a remote controller that performs the management task of these sensors. Each deployed sensor is responsible for detecting and reporting any abnormal behavior and to dispatch the triggered events to the event gatherer component, whose main function is to collect the anomalies and store them in the event storage unit or a database for an additional analysis. In this approach, IDMEF standard were utilized for sharing the the alarm information. Furthermore, the authors design a standardized interface enables to users to view the result reports. By combining the system-level virtualization technology and some known VM Monitor (VM) approaches, the new IDS management system can be used to handle most of VM-based IDSs

1.7.3 Anomaly based Detection Approaches

The authors, (Velmurugan & Thirukumaran, 2012), have proposed an anomaly IDS to overcome the limitations of the traditional IDSs. In their work, they proposed two approaches based on two key ideas: *Performance and Information*. The performance approach is built on artificial neural network algorithms. Its main aim is to detect any attempts to exploit system vulnerabilities. It contributes to automatic discovery of new attacks. The performance approach has an event auditor. It receives profiles that include user behaviors captured by the event auditor node. It also classifies the received data; any deviation observed is considered an anomaly. The information approach uses a database, which consists of pre-configured rules or information regarding specific vulnerabilities and attacks. Hence, the information approach can have a high discovery rate of known attacks.

An anomaly detection system was proposed by (Pandeeswari & Kumar, 2016). It works at the *hypervisor* — Virtual Machine Monitor(VMM) — layer and is thus named *hypervisor detector*. It is designed to detect abnormal behaviors on virtual networks by analyzing network events within the virtual machines. The main goal of this approach is to improve the accuracy of the intrusion detection system by using an hybrid algorithm, which is a combination of *Artificial Neural Network (ANN) and Fuzzy C-Means clustering (FCM) algorithms*. The proposed approach follows three stages. In the first stage, the data-set is split into training subsets by using a fuzzy clustering technique. For the experiments, DARPA's KDD cup data-set 1999 was used. In the second stage, different ANNs are trained based on the data sets that were obtained after the splitting in the first stage. Then, the fuzzy aggregation module is used for re-learning and combining the generated ANN modules into a single ANN module to eliminate the errors at different ANNs. In the third phase, the fuzzy aggregation module is again used to aggregate the ANN module's results into a unique module by which it will be able to eliminate the detected errors.

Wang, Zhijian and Zhu, Yanqin (Wang & Zhu, 2017), propose a centralized host-based Intrusion detection system to enable cloud users or tenants to decrease the resource utilization. Generally, the Framework uses agents to collect the virtual machines' logs. Then it stores them in a centralized location for analysis purpose. The detected results, obtained after logs analyses, are then sent to each virtual machine. The proposed framework is built on OpenStack (Openstack), which is an open source source cloud platform used to built a private cloud environment. The framework encompasses of four main modules including:*Data collection , Data pre-prosing , Data detection , and Alarm report module* . Data Collection Module utilizes agents(Logstash) to collect data information from virtual machines. Thereafter, the collected data is stored into Elasticsearch cluster for farther analysis by the detection center.Data Pre-Prosing Module is used to prepossess the collected logs by prepares unified and effective data for Data Detection Module, which uses a decision tree (DT) for classifying the data and send an alert to the victim virtual machine If an abnormal events are detected.

(Roschke *et al.*, 2009) proposed virtual machine (VM) IDS architecture which is an extensible IDS management architecture that build based on a novel mechanism of event gatherer component. The main goal is preventing the VMs from being compromised. The architecture consists of several IDS sensors deployed on each VM as well as a remote controller that performs the management task of these sensors. Each deployed sensor is responsible for detecting and reporting any abnormal behavior and to dispatch the triggered events to the event gatherer component, whose main function is to collect the anomalies and store them in the event storage unit or a database for an additional analysis. In this approach, IDMEF standard were utilized for sharing the the alarm information. Furthermore, the authors design a standardized interface enables to users to view the result reports. By combining the system-level virtualization technology and some known VM Monitor (VM) approaches, the new IDS management system can be used to handle most of VM-based IDSs

Similarly, (Ganeshkumar & Pandeeswari, 2016), proposes an anomaly-detection-system framework that targets the hypervisor level. The framework is also named hypervisor detector. Its main purpose is to detect abnormal activities in the cloud. The proposed work was developed to minimize the false negative rate and to provide high detection accuracy. The hypervisor detector model is built on adaptive neuro-fuzzy inference system (ANFIS). This is an integration of fuzzy systems with the adaptation and learning proficiency of neural network. It was developed to monitor the activities of the virtual machines and to provide flexibility in terms of detection. It detects abnormal activities belonging to a host and network.

From the foregoing review of some previous work in intrusion detection, the following research gaps and shortcomings have been identified:

- The approaches that deploy security solution at the hypervisor layer have an issue on making the security solutions cloud architecture and platform dependent. Placing the IDS solution in the hypervisor as a centralized IDS may cause bottleneck when traffic flow increases. In addition, assigning the security control to the cloud provider may lead to a loss of security control as the same hardware and the hypervisor software are shared by VMs, which opens the possibility of a compromised and exploited VM to affect the other hosted VMs or even the hypervisor itself.
- The security approaches that are based on signatures are not effective: it is not capable of detecting unknown attacks. Moreover, relying on attack signatures in detecting attacks is not resource-efficient. This is because some attacks, DDOS, for instance, can cause an enormous amount of signature packets in a very short period of time. Further, the discussed signature-based approaches do not take into the consideration signature maintenance (updating the signatures).
- Collaborative security solutions are developed to provide an advance detection as they enable exchanging of alerts that help in recognizing and avoiding similar attacks at other cloud nodes. They are not, however, capable of identifying unknown attacks as

they are signature-based dependent. They suffer from the limitations of the signature-based approaches mentioned earlier. They do not have adequate mechanisms for sharing alerts between nodes. First, the alerts needed to be pre-processed before shared. Second, the mechanism does not include clustering of the IDS nodes based on their security functionality and needs. For instance, some IDSs target the DOS attacks while others target SQL-injection attacks. In this case, any DOS alerts triggered by the nodes needed to be sent to only the IDS nodes that targeted DOS; this is also true for the SQL injection. If this happens, it will reduce the number of the alerts shared between nodes. And, as a result, causes a reduction in the cost since the security model will not require a large amount of computational and network communication resources.

- In anomaly-based and hybrid intrusion-detection approaches, various algorithms have been proposed. They concentrate on detecting attacks, more specifically, they focus on detection rates and with less focus on generating new signatures and measuring the latency in terms of sharing the new signatures. The discussed approaches do not take into consideration faster generation of signatures in automated manners. Furthermore, updating the detection model by continuously training the model so that it can learn new threats and update the model is not considered.

1.7.4 Hybrid Detection Approaches

Modi, Chirag and Patel, Dhiren (Modi & Patel, 2018) proposed a new IDS framework for detecting intrusion in virtual networks for a cloud environment. In their work, they present an hybrid NIDS to examine and detect intrusion in network traffic and in a cloud environment. They deployed the hybrid-NIDS sensors at each cloud host machine and region to monitor the traffic coming from external networks to the VMs, as well as, to monitor the traffic between the VMs. This security arrangement can help in monitoring simultaneously multiple VMs deployed at the same host. It also ensures the protection of the host machine and the VMs from network intrusions. In this approach, a centralized

location is used to collect and store the alerts generated by the deployed sensors. The alerts are then used to identify the distributed attacks. The hybrid-NIDS sensor has several components including a packet capture component, a signature-based detection component, the network-traffic feature extractor, an anomaly-detection component, a score calculation and an alert system. The packet capture component captures the inbound or outbound network traffic of the VMs and the host machine. It inspects them in real time using a signature-based-detection technique. The signature-based detection component uses both the signature-based techniques and the derived-attack rule database, which is generated using the signature Apriori algorithm that takes the captured packets and known attack rules as an input and generates derived-attack rules to check these packet. If an intrusion is detected, the signature-based-detection component determines the nature of the attack and sends an alert to the score-calculation component. Afterword, the captured packets are then applied to the network traffic feature extractor; it is the responsibility of the extractor to generate network traffic profiles from the captured network packets. The generated network traffic profiles are sent to the anomaly detection components where three classifiers including Bayesian, associative and decision tree are trained and used to predict the class label (intrusion or normal) of the given network traffic profile. The results are sent to the score-calculation component. The score-calculation component determines whether the intrusion detected by any classifier is also detected by other classifiers; the information is used for the making final decision. The alert system component generates for an intrusion that has been determined either by *Snort* or by score calculation. Information on the intrusive connection (e.g., the IP of the attacker, IP of the victim VM, source port, destination port, protocol, and detection result) is included in the alert message. The alerted intrusion is stored in the central log.

(Tupakula *et al.*, 2011) targeted security cloud IaaS by integrating the signature-intrusion-detection approach with the anomaly-intrusion-detection technique. They deployed their hybrid-intrusion-detection systems on top the Virtual Machine Monitor (VMM) to be

able to monitor all incoming/out-going packets and to identify the malicious entity that attempts to compromise the virtual machines.

(Modi & Patel, 2013) merge the traditional signature-based intrusion detection system with the anomaly-based intrusion detection system to enhance the detection accuracy and consequently enhancing the capability of detecting the network attacks. Their approach uses different machine-learning approaches including Bayesian, associative, and decision tree to build the framework. In their work, the incoming/out-going traffic passing through the signature IDS is forwarded to an anomaly IDS for classification.

(Aljawarneh *et al.*, 2018) proposed a hybrid classification-based intrusion detection model and a feature selection to help in detecting attacks over the network. First, the feature selection method is applied to NSL-KDD data set. Later, the n-intrusion detection model based on machine learning approach is built and used to find attacks. Moreover, the captured data is used to improve intrusion detection.

After study of previous work discussed above these research gaps has been identified

- the collaboration among different clouds requires an extensive trust management, which does not exist in the current DCDIDP framework. It does not provide approaches to promote trust among cloud users beyond what is stated in SLA. Information sharing among different clouds is also dependent on the structure of each cloud. Finally, DCDIDP has not been evaluated and verified through practical implementations.
- the discussed collaborative approaches architectures have many shortcomings; firstly, central management and processing of the data which represents a single point of failure.

CHAPTER 2

ARTICLE 1: MULTI-TENANT INTRUSION DETECTION SYSTEM FOR PUBLIC CLOUD TENANTS(MTIDS)

Mohamed Hawedi[1], Chamseddine Talh[1], Boucheneb, Hanifa [2]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Computer Engineering and Software Engineering, Polytechnique Montréal
2900 Edouard Montpetit Blvd, Montreal, QC H3T 1J4

Article Published « The Journal of Supercomputing (Springer) » 1 September 2018.

Abstract:

Cloud computing is an innovative paradigm technology that is known for its versatility. It provides many creative services as requested and it is both cost efficient and reliable. More specifically, cloud computing provides an opportunity for tenants to reduce cost and raise effectiveness by offering an alternative method of service utilization. Although these services are easily provided to tenants on demand with minor infrastructure investment, they are significantly exposed to intrusion attempts since the services are offered under the administration of diverse supervision over the internet. Moreover, the security mechanisms offered by cloud providers do not take into consideration the variation of tenants' needs as they provide the same security mechanism for all tenants. So, meeting tenants' security requirements are still a major challenge for cloud providers. In this paper, we concentrate on the security service offered to cloud tenants and service providers and their infrastructure to restrain intruders. We intend to provide a flexible, on-demand, scalable, and pay as you go Multi-tenant Intrusion Detection System (MTIDS) as a service that targets the security of the public cloud. Further, it is designed to deliver appropriate and optimized security taking into consideration the tenants' needs in terms of security service requirements and budget.

2.1 Introduction

A new phenomenon that paves the way to change the perspective towards deploying, scaling, updating, maintaining, and paying for information technology (IT) services is universally acknowledged as cloud computing. It is deemed as a new economic paradigm in the computing environment. Peter Mell and Timothy Grance (Mell & Grance, 2011), define cloud computing as " a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". Cloud computing provides various services that are elastically offered to users over the internet. This enables users to comfortably perform their daily jobs regardless of their locations. The significant advantages of cloud computing have attracted a number of companies such as (Amazon, 2015), (Azure, 2018), (Rackspac, 2018), (Arba, 2018) to get involved in this market.

However, the number of cyber-attacks against enterprises has grown tremendously, which impacts cloud computing negatively in terms of cloud resource protection. This has warned companies about the safety of their information when their data is transferred to the cloud. It has raised the following questions: 1) Does the security provided by Cloud Providers (**CPs**) meet the tenants' requirements?

2) Can tenants have full control over the security of their virtual infrastructure?

Despite the fact that CPs offer some security services, depending on these services can lead to serious security issues. Moreover, cloud providers fail to deliver high quality security control as they are not aware of the architecture of the hosted service (Almorsy *et al.*, 2016). Since CPs provide their services to different tenants, CPs are faced with a lot of changes to their security requirements. The communication between CPs and tenants should be established based on negotiation and agreement before applying any security properties. This communication is a major challenge as there is no standard regulation in terms of notations of security specifications that can be employed by both

CPs and tenants to represent their offered or required security properties. Basically, the security provided by cloud providers is very limited to securing cloud infrastructure and cloud platform services including the physical security of the data-center, network infrastructure, virtualization platform and infrastructure (Demchenko *et al.*, 2017). As an example, Amazon Web Service (AWS), an IaaS provider, states that tenants are responsible for securing their VMs as they are allowed to run any operating system (OS) or modify its services (Shawish & Salama, 2014). In the AWS public cloud, tenants are responsible for securing Amazon Machine Instances (AMI), OS, applications, and data including data in motion, data at rest, and data stores as well as credentials, policies and configuration. Additionally, the tenant is specifically responsible for complying with the Acceptable Use Policy (AUP), ensuring the correct use of the cloud platform, updating security, and patching the guest OS and installed applications (Demchenko *et al.*, 2017). This raises many questions around the security provided by CPs. One critical issue is when an insider attack such as a cloud client abuses the cloud computing power and storage capacity or attempts to attack outside the cloud. According to (Patel *et al.*, 2013), Amazon Elastic Compute (EC2) was used by the attacker, an AWS customer, to attack Sonys online entertainment systems. Millions of customer accounts were compromised which is considered as the biggest breach of data in the United States.

Meeting tenants' security requirements is still a major challenge for CPs. The security mechanisms offered by CPs do not take into consideration the different tenants' needs as they provide the same security mechanism for all tenants. For instance, some providers use a firewall to block external attacks while others use Access Control Lists (ACLs) on gateway routers (Adil & Ijaz, 2015). In fact, each tenant needs security policies based on their requirement. Thus, a network security system is individually requested for each tenant which secures them based on their requisites. Another critical issue is cost management in terms of the security offered to the tenant. In some cases, the requested security may not be used for a period time. Let's assume that a tenant requests protection against DoS, DDOS, and SQL. After period of time, this tenant has changed his network

topology by terminating the database server or it has been discovered that the traffic related to the database was not received. Therefore, based on this situation, the SQL security mechanism should be temporally suspended to reduce resource consumption costs. Tenants impose their own special requirements to protect their VMs. Thus, adopting traditional protection mechanisms like Intrusion Detection Systems protect the cloud environment. The intrusion detection system is the most significant technology that has recently been used to monitor networks and detect cyber-attacks. The IDS is designed to provide a defense layer against illegitimate users by sensing attacks and raising alerts. The IDS has become an essential security mechanism for securing cloud computing environments since it is not possible to prevent all cyber-attacks (Tan *et al.*, 2014).

Based on the aforementioned problems, we strongly believe that tenants should have a new security as a service mechanism (SaaS) that can tackle these issues. In this paper, we propose an elastic, scalable, and efficient multi-tenant intrusion detection system framework that reduces overhead cost and provides a robust defense mechanism. The primary contribution of this paper is a multitenant intrusion detection system that provides a flexible, efficient, and low-cost SaaS framework for public cloud tenants to deploy in order to protect their own cloud infrastructure. Third party security service providers can offer this framework to their tenants too. The MTIDS is a hybrid-based IDS (rule and anomaly-based) network intrusion detection system that offers appropriate security features for cloud tenants. It proposes to monitor the activities of tenants, capture abnormal activities, examine them to evaluate the probability of being an intrusion, and maintain the consistency of the security strategy. The security strategy would be automatically changed based on the analyzed activities. The MTIDS would be able to activate/deactivate a set of rules, generate new rules, and automatically update the set of rules. Such services are regularly presented within a Service Level Agreement (SLA) context, which is aimed at ensuring the requested service quality. In a nutshell, we will seek to meet the following objectives:

- Presenting and implementing a Multi-tenant IDS as a service (MTIDS), which is an efficient, agile, scalable framework that aimed to reducing cost and meeting the tenants' requirements

- Enabling a higher precision detection consensus by supplying advanced traffic analysis. Based on this analysis, a decision is taken to provide the desired security.

- Enabling tenants to make substantial gains in terms of efficiency, cost reduction, and security as it offers a flexible and adaptive security mechanism.

The remainder of the paper is organized as follows. We demonstrate the motivation in section 2.2. We give a brief background and discuss related work in section 2.3. In Section 2.4, we describe the cloud tenants' requirements. In section 5, we assume different scenarios where public cloud tenants build their topology. The architecture of MTIDS is described in detail in section 2.6. Section 2.7 explains the experiment setup and demonstrates the results of our experiments. We conclude the paper and present future work in section 2.8, and 2.9.

2.2 Motivation

Basically, tenants choose their security mechanism after they evaluate different requirements. This includes risks such as knowing what type of attack they are against, budget in terms of how much they are able to pay for obtaining security, network topology and the type of traffic they could receive. These requirements change permanently as more and more users with different access rights are using the tenant's environment. The following list explains these requirements extensively.

- Infrastructure network topology can help tenants specify their security requirements as it illustrates all the entry points where security should be located. Different languages can be used for designing the topology such as TOSCA (Oasis, 2017), Cloud formation...etc. However, tenants' topology can change for many reasons. For instance,

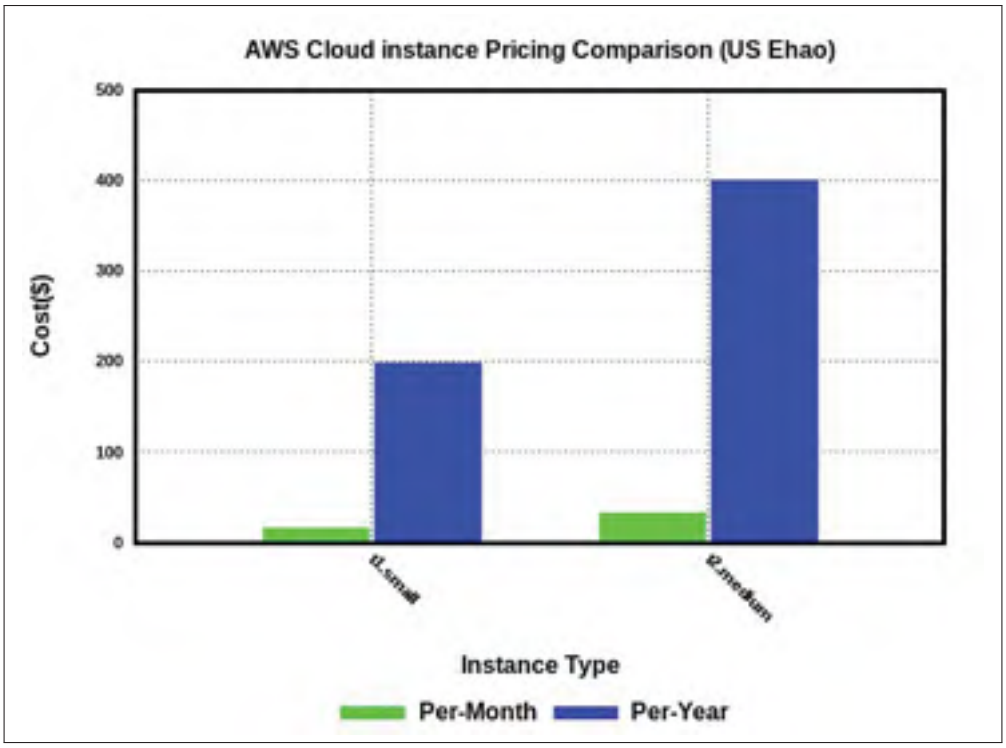


Figure 2.1 AWS instance cost

a tenant needs to scale VMs up and down as the number of users increases or decreases. In this case, the topology of the tenant will change in terms of design. They might have to delete or add subnets and change the configuration of traffic flows. Moreover, migration can be a significant reason for changing the tenants' security requirements.

- The tenant's budget helps to determine the security mechanism and limitation. Saving money while receiving the security mechanism is one of the tenants' requirements. This makes it imperative for security service providers to give a service that meets the tenants'needs. Service security providers can give an estimate of the cost for security if they can figure out resource consumption for the security provided. For instance, since it is possible to know the CPU consumption of each set of rules, security service providers could determine the feature of VM that is suitable for certain situations. This allows tenants to be informed about advanced information regarding the service cost. The MTIDS would be able to reduce the cost as it is designed to stop overspending on security. As we can see in figure 2.1, hosting an instance in a public cloud with one

core costs about 199$ a year and 400$ for a two-core CPU. This is a significantly high cost for the same period of time. Security service cost reduction can be achieved by using VMs with the least amount of resources when possible.

- Tenants' resources are exposed to different and changeable activities or traffic including inbound/outbound traffic such FTP, HTTP, SMTP, etc. The firewall provided by the CP could be used as input for determining which security mechanism should be applied. Figure 2.2 depicts an example of AWS ACL configuration in terms of accepted and denied services. The optimized IDS can build its decision in providing the security mechanism that meets the tenant's needs based on the ACL. Moreover, the Optimized Intrusion Detection System(OIDS) cloud feeds the ACL within the illegitimate traffic should be blocked after the traffic is analyzed.

To sum up, a traditional security mechanism such as IDS is still unable to provide adequate security as it is incapable of adapting to the changes occurring in the tenant's environment. Moreover, the traditional IDS has limited capabilities in terms of handling a large volume of traffic. In fact, IDS starts ignoring the packets if the traffic volume exceeds its capabilities. Here, we point out that each set of rules contains a number of rules that have unique signatures that characterize the patterns of attacks. Thus, as the number of rules in the set increases, the utilization increases, and the system becomes overloaded. Subsequently, this results in an exhaustion of resources allocated to the tenants as each inbound/outbound packet needs to be compared to the signatures specified in each rule. EX. 2 shows an example of rules. Moreover, having an optimized IDS that can adapt with the occurred changes in the tenant's environment would strengthen the security mechanism and encourage more tenants to move their environment to cloud. To this end, having knowledge of these factors helps the MTIDS to take the right decision in terms of determining the following:

1. Which security mechanism should be assigned to the tenant?
2. When set the security mechanism should be activated/deactivated?

3. Where should we place the security mechanism?

4. What will happen if the requested security is not used for a long time?

5. Is the tenant's budget enough to get appropriate security?

Example 2. *alert tcp EXTERNAL_NET 80 − > HOME_NET 7070 (msg:"ICMP test"; sid:1000001; rev:1; classtype:icmp-event;)*

Generally, this rule examines any traffic directed toward the network and it will generate an alert whenever the IDS detects traffic headed inbound from outside to the network over the port (Source) 80. Basically, Snort rule contains two parts: namely Rule header, and Rule Options. The syntax of the rule is explained as follows:

- **Rule Header:** Rule Header: (alert tcp $EXTERNAL_NET 80 -> $HOME_NET 7070). **Alert** which represents the rule action and the alert will be generated in case of the condition met. **$EXTERNAL_NET 80** represent the **source IP and the Port Number** receptively. The IDS will consider all external traffic(source IP) over port number 80. **The direction** from the source to destination is represented by **->** . ($HOME_NET 7070) refer to **the destination IP**($HOME_NET) and its **port number** which is 7070 .

- **Rule Option:** (msg:ICMP test; sid:1000001; rev:1; classtype:icmp-event;) **Msg:** Message will be included with the generated alert(ICMP test). **Sid:**:1000001 represents the rule number or ID. **rev:1** Represents the Revision number. Using such option enables easier maintain for the rule. **classtype:**icmp-event – it basically used to categorize the rule as an "icmp-event". This categorization helps with the organizations of the rules.

2.3 Background and Related Work

This section discusses the existing studies related to our proposed approach.

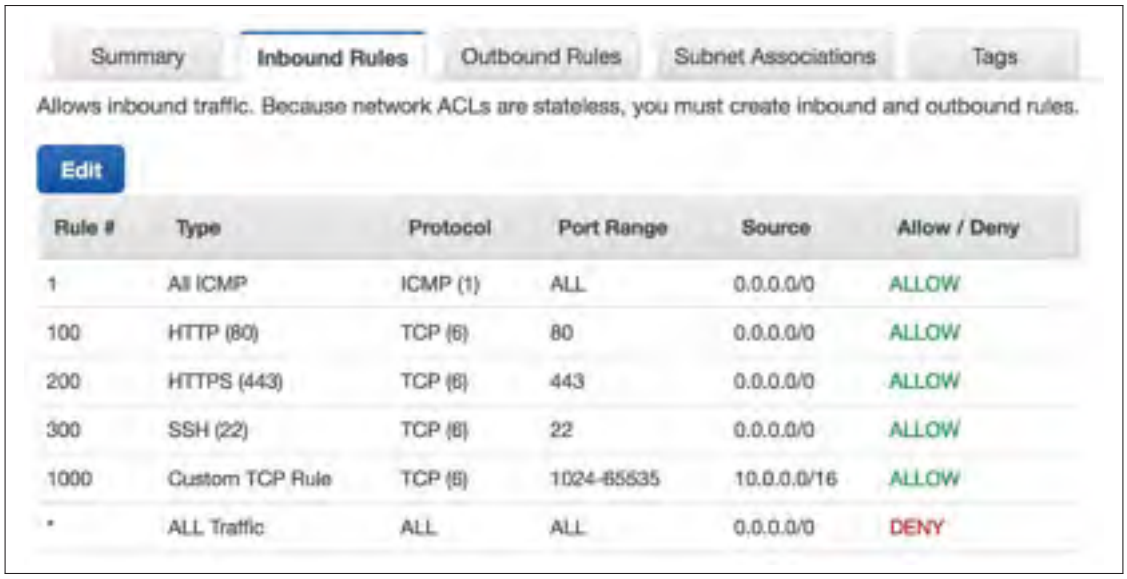


Figure 2.2 AWS Access Control List Example

2.3.1 Fundamental concept of Cloud computing

Cloud computing can be defined as the way to store data or information permanently on servers over the internet and temporarily caching them on the client side via laptops, sensors, computers, etc (Shawish & Salama, 2014). In fact, cloud computing has not been created from scratch and several technologies have paved the way for the cloud. Virtualization plays an essential role in terms of designing the cloud (Modi & Acha, 2017). This is a technology that provides an abstraction of the physical hardware and virtualized resource details for high-level applications (Zhang *et al.*, 2010)], and it is one of the most significant technologies utilized in IaaS. It increases efficiency in terms of performance, maintenance, and cost reduction for the computing services provided to users. This technology is used by different Cloud Service Providers **(CSP)** such as Amazon and Microsoft Azure etc. In cloud computing, hardware and platform level resources are provided as services on-demand (Zhang *et al.*, 2010). Basically, these services can be categorized into three models (Osanaiye *et al.*, 2016), (Park *et al.*, 2016): Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Infrastructure as a Service (IaaS) refers to the allocated infrastructural resources on

demand in the form of VMs. The concept behind the IaaS is that the servers, network and storage are arranged to be available to the end user on-demand, as a service. An IaaS provider refers to the cloud proprietor who offers IaaS such as Amazon EC2 (Amazon, 2015), Microsoft Windows Azure(Azure, 2018). Platform as a Service(PaaS) refers to providing platform layer resources that include OS and software frameworks (Rosado, 2012). For example, Microsoft Windows Azure. Software as a Service (SaaS) refers to offering applications on-demand on the internet such as SaaS providers (Zhang *et al.*, 2010). Salesforce (salesforce, 2017) is an example of SAAS. Figure 2.3 shows these services.

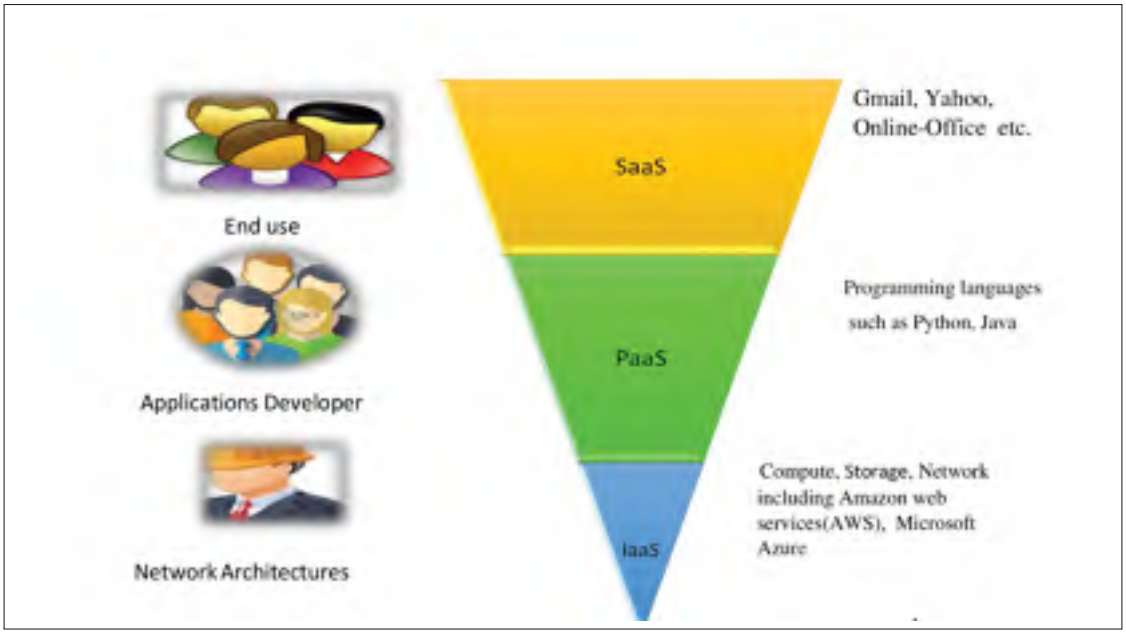


Figure 2.3 Cloud computing Models

2.3.2 Intrusion detection system(IDS)

IDS can be used to function as an Intrusion Detection System or Intrusion Detection and Prevention System **(IDPS)**. Intrusion detection refers to the process used for monitoring all events that occur in a network or computer system. It also includes inspecting them for signs of potential accidents, which are violations or impending threats to the computer

security strategy, policies, or standard security practices. Further, IDPS refers to the process utilized for performing intrusion detection and it makes an effort to stop detected potential incidents (Scarfone & Mell, 2007). Thus, we can define the intrusion detection system (IDS) as a software that provides automation of the intrusion detection process. IDS is categorized into two types: Host-based and Network-based (Oktay & Sahingoz, 2013).

- Network-based intrusion detection system (NlDS): NIDS is employed for observing, examining and analyzing certain components for precise and predefined divisions of network traffic (Oktay & Sahingoz, 2013). NIDS observes network traffic for certain network segments or devices and inspects the activities of the network to figure out any suspicious activities. It is usually located at networks' borders, asinas in closeness to boundary firewalls or behind routers, virtual private network, remote access servers, and wireless networks (Scarfone & Mell, 2007).

- Host-Based intrusion detection system (HIDSs): Host-based intrusion detection system (HIDSs): HIDS is used for observing and analyzing the host system's characteristics once an incident happens such as suspicious activity like system calls, processor threads, entity and configuration access or modifications. Usually, the HIDS is placed on the most important host that is used to store private and vital information. Also, it is possible for the host to perform some of the NIDSs functions if it is placed on one host and configured to detect network behaviors (Oktay & Sahingoz, 2013). For example, in the AWS cloud (Amazon, 2015), host-based IDSs should be deployed in each of the launched VMs. Host-based IDSs can use both system logs and OS audit trails for system state monitoring such as detecting a particular program that accesses specific resources.

2.3.2.1 Intrusion detection techniques

Indeed, different techniques can be used to analyze and detect attacks. These techniques are categorized into various principal categories (Oktay & Sahingoz, 2013), as described in the following section:

- Signature-based lDSs: Signature-based techniques are also known as knowledge-based or misused-based techniques (Keegan *et al.*, 2016). Misuse detectors aim to analyze system activities by searching for an event or a group of events that match the signature that identifies known attack types. As identical match-patterns to known attacks type are called as signatures, misuse detection is occasionally named signature based detection. The commercial product is the most common form of rule-based IDS. This technique is used to determine intrusion attempts based on pre-defined rules against known attacksOktay & Sahingoz (2013).
- Anomaly-based IDSs: IDSs particularly focus on monitoring and analyzing system behavior. Anomaly detectors aim at identifying unusual and abnormal behavior on the host and network. They assume that abnormal activities are different from authorized activities. Hence, these activities can be detected using systems, which can determine normal or abnormal activities. Anomaly detectors learn themselves by constructing profiles representing the habitual behavior of users, hosts, and the network. Also, the profiles are made from data gathered for a period of time and over a stage of normal operation. The anomaly detector function then collects data events and employs several measurements to determine when monitored activity moves away from common habit (Zarrabi & Zarrabi, 2012).

2.3.3 Related Work

In this section, we review relevant work that adopted the traditional intrusion detection system (IDS) for enhancing security in cloud computing. Each work focused on various

cloud service modules or various objectives. Essentially, an intrusion detection system encompasses a verifying process that examines the entity's behavior looking for attack signatures that are precise patterns which indicate malicious or suspicious intent.

A lot of the proposed research concentrates on providing security as a service for cloud providers as part of their infrastructure which they can offer to tenants.

Varadharajan and Tupakula (Varadharajan & Tupakula, 2014) have proposed safeguarded security as a service model based on the intrusion detection system that targets the infrastructure and is offered by the cloud provider to its tenants . The proposed service model provides baseline security to the cloud provider for protecting its cloud infrastructure. It also offers elasticity to tenants for gaining further security functionalities that match their security needs. Basically, this approach aims at protecting the tenant from internal attacks by deploying the proposed security mechanism on the hypervisor in privilege 0 by which they have the ability to monitor the VMs that are gained.

Alharkan and Martin (Alharkan & Martin, 2012) proposed a scalable intrusion detection system as a service (IDSaaS), which can detect malicious or suspicious intent. IDSaaS is a network-based IDS that targets to protect IaaS. It is built on top of the AWS cloud and it is used to monitor and log suspicious activities of the network between tenant VMs. This framework used the signature based technique to determine the validity of events. Particularly, the traffic is captured and analyzed based on signatures that have been pre-defined and can be updated on a systematic basis.

Gul and Hussain (Gul & Hussain, 2011) introduced a distributed multi-threaded paradigm IDS that targets the security of cloud computing. The multi-threaded IDS model is designed to handle a huge flow of network traffic. This traffic is then analyzed and an alert is raised if there is any suspicious activity. This approach suggests that the raised alert should be sent to a third party. This third party is responsible for sending alert reports to a cloud user in case there is an intrusion attempt as well as advice to the cloud provider regarding the occurred intrusion. It is composed of three components including

capture and queuing, analysis and reporting. The first component captures traffic. Next is queuing and then it is sent for analysis by the analysis component. A signature technique is used for detecting any suspicious activity. If the traffic violates the signature, an alert will be generated and sent to the report component. Hereafter, the report component will send the generated report to the third party. Then, it will be sent to the cloud user and cloud provider.

Patel, S.K., Sonker (Patel & Sonker, 2016), proposed a rule-based NIDS designed to detect Dos attacks and to generate Port scan detection rules. The proposed approach enables safeguarding networks from any illegitimate access. In this rule-based detection system, network traffic which are passed over the network are captured and then inspected looking for any suspicious activities (intrusions). The system will generate an alarm if any packet matches the signature.

A network intrusion detection system (NIDS) was presented by Gupta, S., Kumar, P (Gupta & Kumar, 2017) aiming at securing Cloud resource against distributed denial of service attacks. In this approach, a tenant Virtual Machine(VM) profile is used for predicting the DDOS. In essence, The profile of the tenant is used for classifying DDoS the signatures of the attack as well as for performing back off based detection of signatures. Various important processes phase were proposed in this approach. The first phase is initialization and rule update, which is created to set the thresholds for the attack signatures existing in a client VM profile. In detection phase, packets are examined for detection attack that meet the signature. In Alert and Response Generation phase, alerts with will be send to the Virtual Machine of the client if suspicious activities were detected. Wang, Zhijian and Zhu, Yanqin (Wang & Zhu, 2017), propose a centralized host-based Intrusion detection system to enable cloud users or tenants to decrease the resource utilization. The proposed framework uses agents to collect the virtual machines' logs. Then it stores them in a centralized location for analysis purpose. The detected results, obtained after logs analyses, are then sent to each virtual machine.

Modi, Chirag and Patel, Dhiren (Modi & Patel, 2018) s suggested a new IDS framework

aims at detecting intrusion in virtual networks for cloud environment. In this paper, the researches presented a Hybrid-NIDS to examine and detect intrusion in the network traffic and in cloud environment. They have deployed the Hybrid-NIDS sensors on each cloud host machine and region to monitor traffic coming from external networks to VMs, as well as, traffic between VMs. This kind of deployment aids monitoring multiple VMs deployed on the same host simultaneously, and to ensure protecting the host machine and VMs from network intrusions. In this approach, a centralized location is used to collect and store alerts generated by the deployed sensors to identify the distributed attacks.

Other researchers concentrate on deploying the intrusion detection system in a collaborative manner for detecting suspicious activities.

The authors of (Alruwaili & Gulliver, 2014) propose the concept of a collaborative IDS for protecting the cloud infrastructure layer against known and unknown threats by using signature based and anomaly techniques. Notably, the proposed approach is aimed at protecting service providers and their tenants against loss of services caused by both known and unknown threats. Moreover, the framework can work in a global, collaborative manner to achieve comprehensive monitoring of the cloud resources.

The work done by Ficco et al. (Ficco *et al.*, 2013) presented a distributed architecture to provide intrusion detection in cloud computing. The proposed approach helps to observe the CP and detects if certain CP resources have been compromised by other providers. It also identifies scalable distributed attacks as opposed to the federated cloud systems. A collaborative IDS distributed in the cloud layers including the infrastructure level, the platform level, and the application level to help in detecting coordinate attacks was introduced in (Gul & Hussain, 2011). This framework allows cloud providers to implement distributed IDSs for detecting intrusion attempts and it enables cloud tenants to monitor their applications. Moreover, network intrusion detection named multi-threaded distributed IDS deployed in the cloud computing environment (Gul & Hussain, 2011) was developed to deal with widespread network access traffic, and administrative data control and applications in the cloud. The proposed IDS can manage a huge flow of data

packets, and then produce reports after analyzing those data packets. Also, the proposed model suggested that cloud users should buy the third party for managing and monitoring. This third party is responsible for sending warning reports to cloud users in case there is an intrusion attempt as well as giving advice to cloud providers regarding the occurred intrusion.

The work introduced by Taghavi Zargar et al. (Zargar *et al.*, 2011), proposed a framework (DC- DIDP) which focuses on coping with an attack using intrusion detection and prevention systems (IDPSs). The proposed intrusion detection framework (DC- DIDP) allows all cloud service providers to cooperate in a distributive way at different operational levels to reply to attacks and offer universal IDPSs. Global databases are shared among cloud providers on which they can detect sophisticated cooperative intrusion.
Chi-Chun Lo et al. (Lo *et al.*, 2010) proposed an intrusion detection system based on the cloud computing framework to monitor cloud regions. In this approach, IDSs are deployed in each cloud-computing region for detecting intrusion attempts. They send alerts to each other in case intrusions are detected, and judge the honesty of these alerts. The proposed architecture consists of many components that involve intrusion detection, alert clustering, threshold check, intrusion response and blocking and cooperative agent. The concept beyond proposing the alert clustering module component is to gather alerts generated via other regions and then through collaborative agent component alert messages that are exchanged among IDSs. The judgment regarding whether these alerts are true or false is determined by estimating the riskiness of the gathered alerts.
Overall, all reviewed studies have some drawbacks. Most of the proposed approaches suggest that the security offered for tenants should be controlled by the CPS. Hence, this will not help to meet the tenant's security requirements. In addition, many reviewed studies propose deploying their security mechanisms on top of the VMM. This imposes many risks to the tenants' VMs as the VMM can be compromised. Moreover, some of the revised researches merely target detecting all known attacks. For the approaches that propose the collaborative IDSs, they are limited to exchanging information between CSPs.

In fact, tenants' security requirements are not considered.
Our proposed approach fills the gap of the aforementioned work. In fact, there are several aspects that differentiate our work from the discussed work.

- In our work, we take into consideration meeting the tenant's needs regarding security by providing Optimized IDS as a service. This is fully controlled by the cloud tenant and it is adaptable to the tenants' environment. Multi-tenant IDS(MTIDS) is an efficient, flexible and scalable solution that offers different security levels to meet the tenant's needs.

- Unlike the aforementioned work, which solely concentrated on providing security solution without considering the cost reduction, the MTIDS contributes to the cost reduction, which results from optimizing the resources consumption. And this considered as a significant feature in this proposed approach.

- Besides the anomaly detection, the proposed approach success in providing adaptable or optimized signature IDS where the security services (Set of rules) or signatures are activated/deactivated based on the traffic. The incoming/out coming packet got analyzed looking for the port number that meets the MTIDS policies which illustrated in Table 2.1 . Indeed, the anomaly IDS is designed to be offered as a service that capable to meet the tenants requirements as it provides different classification models.

- Unlike the other approaches which rely on a set of generic rules or signatures, our approach offer an automatic creation of custom rules based on tenants' environment.

2.4 Cloud Tenants Requirements and Perspectives

Due to heterogeneity of the cloud architecture and different security perspectives of the tenants, the security requirements tend to vary. This variation in points of views is definitely possible since each tenant has own infrastructure typologies, policies, and services for users. Details of these differences are listed as follows:

- As cloud tenants utilize and pay for the services offered by the cloud provider, tenants should obtain a robust security mechanism that meets their needs to monitor their virtual resources against attacks such as SQL injection and DoS/DDoS. As we have previously mentioned , the existing security provided by cloud providers does not meet the tenant's needs as cloud providers offer a standard security mechanism for all tenants.

- Some tenants have a lack of trust for the existing security mechanisms provided by cloud mechanisms as these mechanisms are configured and managed by the cloud provider. Thus, enabling the tenant to even partially manage and identify their own customized security strategies for each component being monitored will ensure more trust.

- Tenant needs to know whether their VMs and services are being employed to attack other victims. Hence, the security mechanism offered to cloud tenants must provide a more transparent information about attacks.

- The tenant requires the cloud provider to ensure that attacks are detected on their cloud infrastructure and offered services. However, as we have mentioned above, cloud providers are given the responsibility to protect the tenant's environment.

To sum up, offering a reliable security mechanism that meets the requirements of the tenants enhances trust and encourages more tenants.

2.5 Cloud Tenant Topology Scenarios

As we discussed above, since different tenants deploy their environment on top of the cloud, the security perspective tends to be distinct. In fact, each tenant has different cloud architecture, which is composed of topology policies and so on. Hence, the ability of understanding and categorizing the tenant's needs can determine how and where to provide support for tenants in the underlying designed architecture. Figure 2.4 depicts

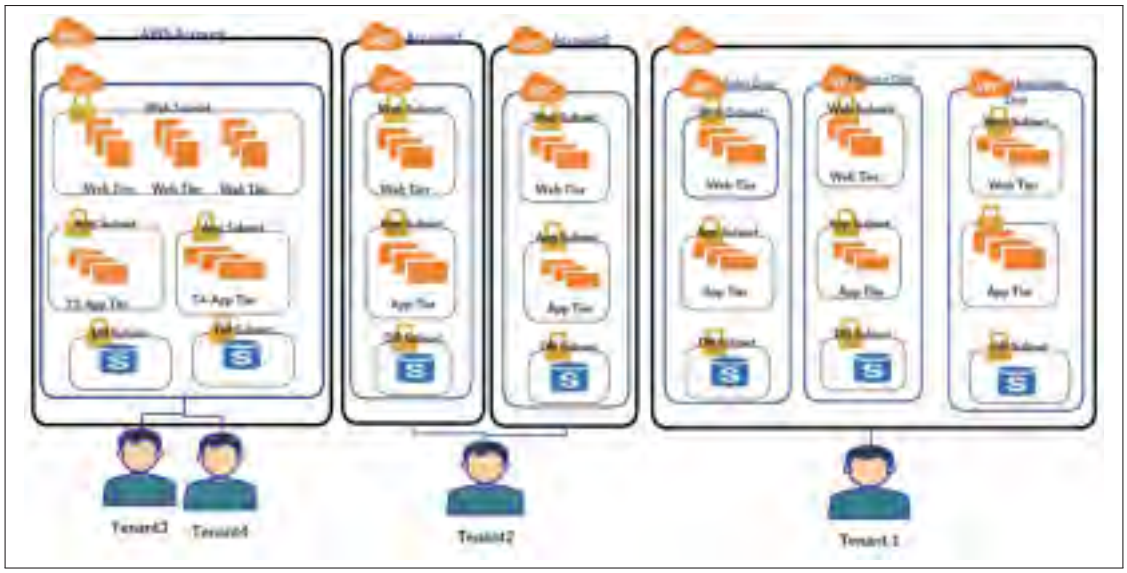


Figure 2.4 Cloud Scenarios (AWS based vision)

how tenants deploy their cloud resources on top of the public cloud (i.e. AWS) (Amazon, 2015). In this proposal, we assume that some tenants have a single account while others possess many accounts. This can only occur when tenants own large organizations where a large number of machines need to be operated. Tenants can have complete control over their compute resources using Elastic Compute Cloud, **EC2**. **EC2** is a web service that provides re-sizable compute capacity in the cloud [30]. Each AWS account has a single EC2 where a tenant is able to deploy their compute environment. With this assumption, we contemplate multiple scenarios for cloud resource deployment. In the first scenario, we assume that **Tenant 1** represents a small organization that has a few departments (i.e. sales, finance and hospitality), each of which want to deploy an infrastructure on top of a single AWS account. These departments can be isolated as VMs are deployed on different virtual private clouds **(VPC)**. Tenants are able to share information by connecting these VPCs. As can be seen in figure 2.4 , the web and application tier are delivered into different subnets to all architecture designs. In the second scenario, we stipulate that one of our tenants may have a large organization consisting of several branches. These branches need multiple accounts to ensure high availability of the service. Hence, **Tenant2** deploys its infrastructure on top of two AWS accounts. The third scenario targets conjoint

organizations. In this case, we assume that **Tenant3** and **Tenant4** have a joined account to deploy and manage their resources. In this infrastructure, isolation is accomplished at different tiers of the architecture. As we can see in figure 2.4, for each tenant, the application tiers and databases are isolated whereas the web tier is delivered as a shared subnet. Based on this assumption, there is a need to provide a new security mechanism for the public cloud tenant. Our approach enables tenants to make substantial gains in terms of efficiency, cost reduction, as well as security. In addition to that, it assists tenants to ensure proper cost allocation as they receive optimized security protection that meets their needs and adapts to the changes of the tenants' requirements over time. Moreover, the proposed security framework presents flexibility in terms of monitoring the tenants' resources and it automatically changes the security mechanism based on actions or events facing the tenants' resources. These features provide significant gains for tenants who manage multiple public accounts. Our framework, in general, enables cloud tenants to define their needs. It enables tenants to define their requirements by which it determines and categorizes the Optimized IDS that meets the tenants' needs. Based on this requirement, the OIDS will assign an appropriate optimized IDS that meets the needs. Moreover, the Optimized IDS(OIDS) is designed to be able to adapt to change occurs in the tenant's environments.

2.6 MTIDS Architecture

This section presents the architecture of the MTIDS. The main idea is to build a framework that takes into account the tenants' security requirements and offers a flexible security mechanism with a reduction in cost. As depicted in figure 2.55, the MTIDS consists of multiple main components: the Optimized IDS Controller (OIDSC), Packet Analyzer, Optimized Signature IDS (OIDS), and Anomaly Intrusion detection component (AID). These components are complimentary to each other.

1. Optimized IDS Controller(OIDSC) consists of two component including:

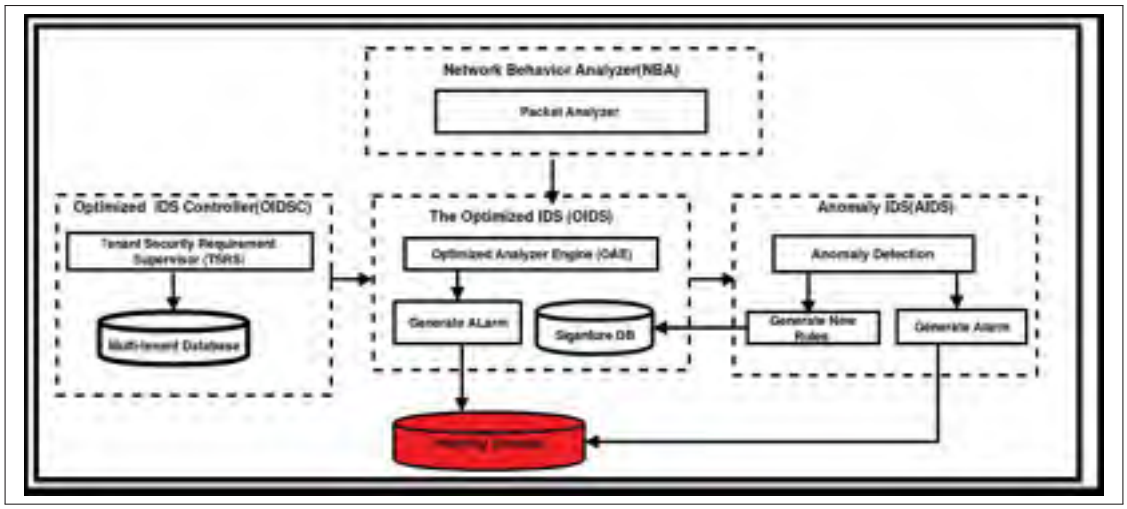


Figure 2.5 Multi-tenant Intrusion Detection System(MTIDS) architecture

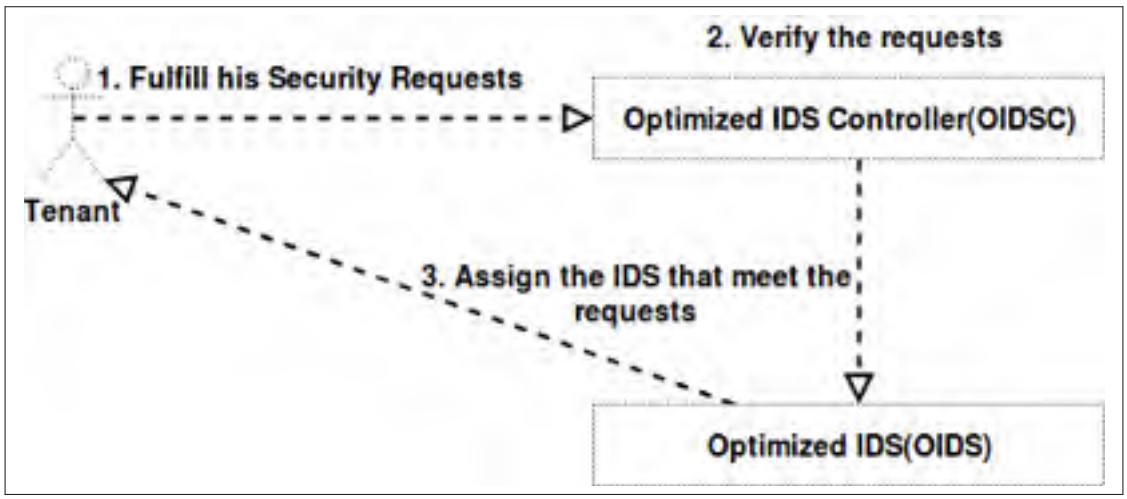


Figure 2.6 Optimized IDS Controller Process

- The Tenant Security Requirements Supervisor(TSRS). This component is responsible for receiving the tenants' security requirements, topology, and orchestration of the tenant application by which the (OIDSC) can assign a specific Optimized IDS that meets the tenants' requirements. This component allows public cloud tenants to specify the security requirements and the underlying topology of their virtual infrastructures. The optimized IDS will be assigned based on these security

requirements. This provides benefits to the tenants in terms of cost reduction and receiving an efficient security mechanism. Figure 6 illustrates the process of security specification that a tenant can follow to meet their need. The second element of OICDS is a multi-tenant IDS database 2.6.

- The multi-tenant IDS database uses a multi-tenancy concept for saving information regarding the tenants including tenant activities, login info, and so on. The saved information could be used to identify the behavior of each tenant by using machine learning techniques.

2. Network Behavior Analyzer(NBA): this component is responsible for analyzing network traffic which enables the Optimized IDS (OIDS) component to assign or activate the optimal security service. Various attributes including port numbers, services name are used to distinguish between different services that run over different transport protocol (iana, 2016). Based on the attributes, the OIDS can adapt to changes occur on the tenant environment. The NBA component is complementary to OIDS. Algorithm 2.2, and 2.3 demonstrate how these components work together to satisfy the tenants' needs.

3. The Optimized IDS (OIDS) is built on top of each tenant's public cloud environments. OIDS is a hybrid-based IDS. Rule based and anomaly based techniques apply all known attacks' signatures. They monitor system activity and classify it as either normal or anomalous. The OIDS is in charge of targeting a specific threat to meet the needs of the tenants. Moreover, the OIDS is designed to adapt to the changes of tenants' security requirements. The following example represents different scenarios of tenant security requirements. In the first scenario, **Tenant A** predicts an attack occurrence (i.e. SYN flood protection attack) in their environment.**Tenant A** requests the IaaS attack protection service against the predicted attack. In the second scenario, **Tenant B** requests security service against a SQL injection only. In this case, the OIDSC will assign the OIDS with SQL Injection to **Tenant B**. Nevertheless, these tenants will still have the right to change their requirements at any time and they

will still be able to have the security mechanism that meets their needs. Afterwards, **Tenant B** changes the IaaS topology. Hence, the security mechanism should adapt to this change. We have other scenarios where tenants are not aware of what security service to choose. In this case, the OIDSC will assign an IDS loaded with all known attack signature detection and anomaly detection. Subsequently, based on the network behavior analyzer's monitored activities, the OIDS will activate/deactivate a security mechanism. The optimized intrusion detection system consists of:

a. Optimized Analyzer Engine Component (OAE): It functions to deter- mine the threat level by analyzing the captured traffic based on certain rule-sets or pre-defined signatures. It then generates an alert in case the captured traffic matches the activity. The OAE is in constant communication with a central controller to keep the signature set updated. The set of rules of the OIDS are activated/deactivated automatically based on the traffic compromising the tenants' infrastructure.

b. Alert Dashboard: The Alert Dashboard is developed to allow tenants to gain information about the monitored VMs. It is used as a graphical user interface (GUI), which allows tenants to look into the generated alerts and correlate them. Tenants can generate reports with different characteristics including time/date, the attack source, or type of threat...etc.

c. The signature Database: The signature Database contains up-to-date attack signatures either through downloading them from public community services or through a subscription service including the Sourcefire VRT (Snort, 2017). Moreover, this database will be updated new rules generated by the anomaly IDS.

4. Warning storage: t is responsible for gathering the warning events generated by the Analyzer Engine Component. This facilitates the tracking process for the notification source and offers flexible management. Further, it is responsible for gathering the alerts generated by the Anomaly IDS component.

5. Anomaly IDS(AID) Component: this component is considered as a part of the OIDS. The main goal of this component is to classify the network traffic forwarded from the OIDS as normal or anomalous. Different methods including heuristics or rules instead of signatures are used to accomplish this classification. Machine learning such as Random Forest (Breiman, 2001) ,Decision tree(DT) (Kevric *et al.*, 2017) will be used to build the anomaly IDS. This component is used to find the beneficial patterns that can describe the behavior of the cloud tenants and use these patterns to build classifiers to acknowledge normal and anomaly intrusion. Hence, it will enhance the performance of the IDS in terms of the detection rate and speed processing.

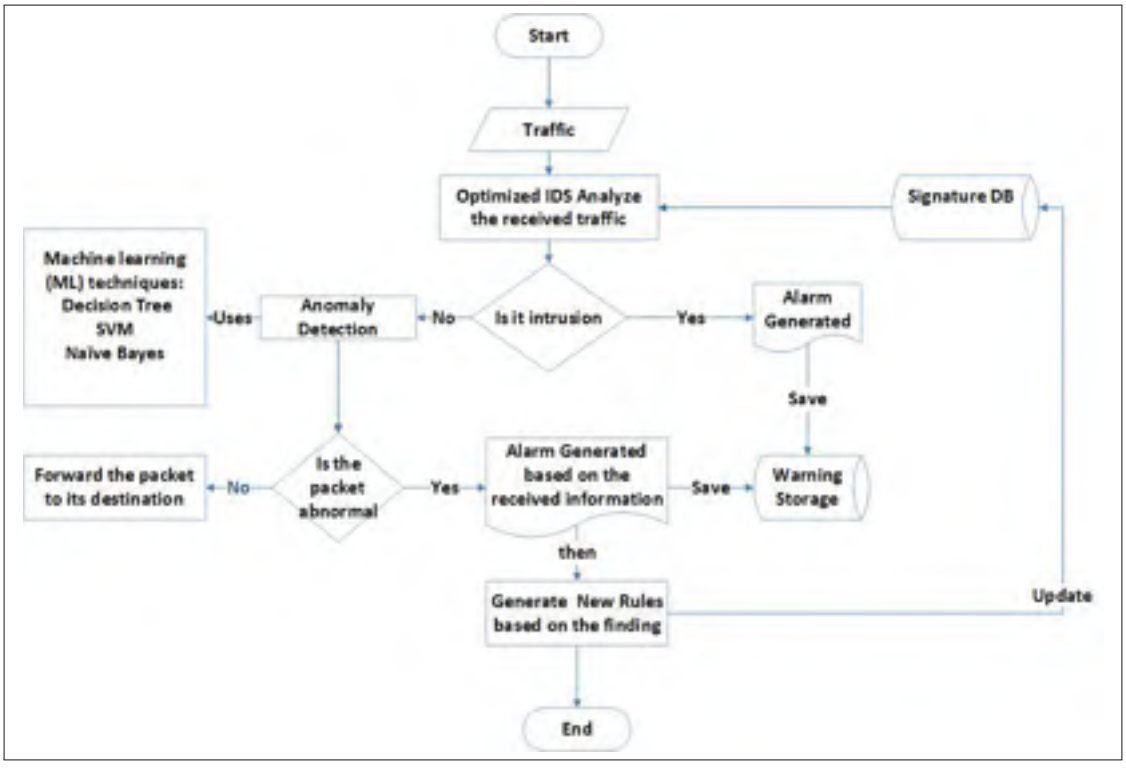


Figure 2.7 MTIDS Mechanism

2.6.1 MTIDS Mechanism

The main purpose for creating the intrusion detection system (IDS) is to monitor incoming and outgoing network traffic and look for any suspicious activity that is probably an indicator of an attack that will compromise the system. Signature IDS refers to the method of comparing predefined rules against captured events to detect any intrusion attempts. The rules consist of information that allows the IDS to take a precise decision including sending an alert. The IPS is responsible for blocking an attack or dropping illegal packets. Rule based systems are considered as the most extensively deployed IDSs. In the rule based system, an accurate signature information improves attack detection. Nevertheless, the rule based IDS is unable to detect unknown attacks. Hence, anomaly-based detection becomes indispensable to detect zero attacks and to generate new rules (Man & Huh, 2012).

In section 2.6, we discussed the architecture of the MITDS process. This subsection provides detail of the MTIDS process. It discusses how MTIDS components work to mitigate the attack. As we previously mentioned, the MTIDS uses signature and anomaly based techniques in order to provide better attack detection. The MTIDS provides a flexible security mechanism that is able to automatically adapt to the change based on tenants' variable security strategies. Different constraints such as tenant security requirements, topology, tenant activities and budget can play an important role in providing the security mechanism. Hence, it is extremely important for the tenant to accurately weigh their security requirement considering these constraints. In the proposed mechanism, we conducted two different scenarios. The first scenario is where the Optimized IDS required by the tenant receives traffic as expected. The OIDS will work normally in case the constraints explained above have not been violated. As we discussed, tenants should identify their requirements to receive a particular Optimized IDS (OIDS) that meets their specific needs. There are a lot of languages that can be used to perform this task. However, OASIS Topology and Orchestration Specification of Cloud Applications (TOSCA) (Oasis, 2017) have been used as a language for the Topology

and Orchestration Specification. As it is demonstrated in Figure 2.7, the optimized IDS examines the incoming/outgoing traffic and then generates alerts. In the second scenario, the OIDS signature does not recognize the attack pattern. In this case, an anomaly based mechanism should be involved to evaluate traffic activities in order to identify the traffic pattern and detect malicious activity. Thereafter, a new rule will be generated based on this detection. The existing set of rules will be updated subsequently. In this work, we have proposed an algorithm as illustrated in subsection 2.6.3 to cope with these scenarios. This algorithm uses the hash tables technique (Yan *et al.*, 2008) to speed up the execution of rule set management.

Table 2.1 Ex.MTIDS Activation/Deactivation Policies

Service	**Protocol**	**Port Number**	**Set of rules**
ftp	TCP	21	ftp
sql	TCP	139/1433/53/445/1434	mysql & sql
tulent	TCP	23	tulnet
tftp	UDP	69	tftp
virus	TCP	25	virus
dos	Ip/UDP/TCP/ICMP	7070/8080/161/6004/80/2048 /515/179/135:139/6789:6790	dos
dns	TCP/UDP	53/any	dns
delete	UDP/TCP	79/80/143/2140/60000/21/111	delete
snmp	UDP/TCP	161-162	snmp
ddos	TCP/UDP/TCP	18753/20433/31335/27665 /27444/6838/10498/12754	ddos
backdoor	TCP/UDP/ICMP	16959/27374/20034/2140/80 /146 /666/34012	backdoor
chat	TCP	6666/1863	chat
netbios	TCP/UDP	135/445/139	netbios
p2p	TCP/UDP	8888/6699/7777/5555/1214 /4242/41170	p2ps
smtp	TCP	25	smtp
imap	TCP/UDP	220/143	Internet protocol

Table 2.2 Ex.Security Services

Key	Value
DOS	DOS SR
DDOS	DDOS, SQL SR
SQL	SQL SR

2.6.2 OIDS Process Flows

Providing security services to meet the requirements of the cloud tenant would encourage more users towards the use of cloud. Thus, the main objective of the proposed MTIDS is to provide a security mechanism that is intended to meet tenants' requirements in terms of the quality of offered security service and the cost. MTIDS is a Multi-tenant IDS as a service that is intended to enable cloud tenants to specify their security requirements. Moreover, it provides an optimized IDS(OIDS) that capable to adapt with the changes occurs in the tenant's environment such as inbound/outbound traffic changes.
This section presents some formal description to represent the semantic scheme of the OIDS. The process of OIDS rely on important constraints, which are the tenants, their security requirements, the traffic they are facing, and the security service polices. In this context, various sets including The Set of Rules, Set of Tenants, Set of Packets, and Set of Security are defined. Moreover, the mathematical relation between them is clarified.

2.6.2.1 Definition of sets

In this subsection, various sets will be defined as the following:

(i) Set of Rules R: Represents the all rules that can be used by IDS.

(ii) Set of Tenants T: Represents the set of all tenants.

(iii) Set of Packets P: Represents inbound/outbound packets in the network.

(iv) Set of Security Services SS: Represents the security services like SQL, DOS, DDOS, ICMP, etc.

2.6.2.2 Definition of relations

This sub-subsection demonstrates the mathematical relations(mathematical function[1]) between the defined sets aforementioned in sub-subsection 2.6.2.1.

- **Security Requirement of the tenant(SRQt):**

$$\begin{aligned} SRQt :& \mathbb{N} \to P(Service) \\ & i \to SRQt(i) \end{aligned} \tag{2.1}$$

This mathematical function defines the security requirements of the tenant i. For instance, the tenant 1 can has the following security requirements:

$$SRQt(i) = \{Dos, DDOS, SQL\}$$

where, P(Service) is the powerset [2] of the set Service.

For simplicity, we consider the departure set is natural number which represent the id of tenants.

---

[1] "A function is a process or a relation that associates each element x of a set X, the domain of the function, to a single element y of another set Y (possibly the same set), the codomain of the function" (Wikipedia, 2018a).

[2] "The power set (or powerset) of any set S is the set of all subsets of S, including the empty set and S itself of the set R" (Wikipedia, 2018b).

- **Activated Set of Rules of the tenant(SRt):**

$$SRt : \mathbb{N} \rightarrow SR$$
$$i \rightarrow SRt(i) \tag{2.2}$$

This mathematical function gets the set of activated rules for the tenant i.

- **Packet Type(PacketType):**

$$PacketType : P \rightarrow SS$$
$$p \rightarrow PacketType(p) \tag{2.3}$$

This mathematical function defines that each packet has type which corresponds to a service that is an element of the security requirements SS. Here, the type of the packet is defined based on the port number. For instance, the SQL requests are likely to be received on TCP Port numbers (139/1433/53/445/1434). Table 2.1 , depicts the policies in terms of defining the port number and their corresponding security service. Basically, OIDS would examine the port number of the captured packet; then it matches them to OIDS policies. If the packet triggers the policies, the security service would be either activated or deactivated.

- **Set of Rules of Service(SRS):**

$$SRS : Service \rightarrow P(R)$$
$$Pt \mapsto SRS(Pt) \tag{2.4}$$

This mathematical function defines the set of the rules corresponding to the services. where, P(R) is the powerset of the set of the rules R.

- **The Security Service Targeted by Rule(STR):**

$$STR : R \rightarrow SS$$
$$r \mapsto STR(r) \tag{2.5}$$

This mathematical function defines the services(DOS, SQL. etc) of the rule r. An example of rule was mentioned in Example 2 where it shows the port number as an important element of the rule structure.

Algorithm 2.1 Tenant-Security-Requirements MSR $< Key, Value >$, SRQt)

```
1 SRt = φ
2 t ← Now
3 begin
4 |   foreach SRQi ∈ SRQt do
5 |   |   SRt ← SRt ∪ MSR.getValue(SRQi);
6 |   end
7 |   Activate (SRt)
8 end
```

2.6.3 Optimized IDS (OIDS)algorithms

This Subsection presents the optimized algorithms, which are based on the defined function discussed in Section 2.6.2.1,in further details. This algorithm is designed to provision security services based on tenants' demand. It also adapts to the changes of tenants' requirements, which alter as the topology changes. Moreover, this algorithm monitors the traffic activities and then provides an appropriate security service based on the monitored activities.

Algorithm 2.2 ActivateAdaptationR (MSR, SRQt,SRt)

```
1  newSR = φ
2  while True do
3  |  pk=CapturePacket()
4  |  typePk ← PacketType(pk)
5  |  if typePk ∉ SQRt then
6  |  |  NewSR ← MSR.getValue(typePk)
7  |  |  SRt ← SRt ∪ NewSR
8  |  |  Activate(NewSR)
9  |  |  SRQt ← SRQt ∪ {typePk}
10 |  end
11 end
```

Algorithm 2.3 DeactivateAdaptationR (MSR, SRQt,SRt, ReqPriod)

```
1  △ t
2  timecapture Map < Key, Value >
3  initiate( timecapture)
4  t ← now()
5  while True do
6  |  pk ←CapturePacket()
7  |  typePk ← packetType(pk)
8  |  Timecapture.update(typePk, now())
9  |  if now()−t > △t then
10 |  |  foreach SRQi ∈ timeCapture.getkey() do
11 |  |  |  time ← timeCapture.getValue()
12 |  |  |  if | now() - t | > ReqPeriod.getValue(SRQi) then
13 |  |  |  |  dSR ← MSR.getvalue(SRQi)
14 |  |  |  |  SRt ← SRt\dSR
15 |  |  |  |  SRQt ← SRQt\SRQi
16 |  |  |  |  deactivate(dSR)
17 |  |  |  end
18 |  |  end
19 |  end
20 |  t ← now()
21 end
```

Overall, the three Algorithms 8, 2.2, 2.3 are complementary to each other. Algorithm 8 is designed to offer security service based on the tenant's requirements whereas Algorithm

2.2 and Algorithm 2.3 are designed to enable turning on/off the given security services based on the activities that OIDS faces.

Algorithm 8Tenant-Security-Requirements Description:

This Algorithm is designed to meet the security requirement of the tenant by activating a set of rules corresponding to their requirements. First of all, the algorithm input maps composed of MSR(key, value). While the key donates to the security service, the value represents the corresponding set of rules. Table 2.2 illustrates an example of the MSR. In addition, SRQT which is based on the formula 2.1, represents the service security requirements of the tenant. While, SRT(line 1), encompasses of rules that is intended to be activated by the tenant security requirements. Lines (3 to 5) check the list of tenant security requirements (SRQT) and get the corresponding Set of rules from the map by using getValue function call. Finally, the algorithm would activate the set of rules as explained in line 7.

performed either by being activated (line10) or by being deactivated (line16). Then, the index is updated (lines 11 and 17).

Algorithm 2.2 ActivateAdaptationR Description: As aforementioned this algorithm activates the security services based on the monitored traffic. In this algorithm, MSR and SRQt are inputs which are also defined as input for the algorithm refAlgo:Security requirment. Furthermore, SRt(tenant's set of rule) is also input which is based on the forumla 2.2. The set newSR will contain of the new set of rules that will be added to tenant's set of rule (SRt). As shown in lines 2, and 3, the algorithm starts monitoring the incoming/outcoming traffic(packets) based on Packet function defined on formula 2.3. It will mainly concentrate on an attributes in each monitored packet which is the port number as each port number corresponds to a security service as revealed in Table 2.1. For instance, the SQL security service is linked to ports (139/1433/53/445/1434). Based on this context, the SQL security service will be activated if the algorithm discovers a new packet with such ports. Before activating any set of rules, the algorithm will double

check whether the set of rules is listed in the activated set of rules defined by tenant as shown in line 5. If not, the algorithm will get the set of rules corresponding to the type of packet and the MTIDS policies 2.1 as explained in the formula 2.3. Then this set of rule/rules would be activated.

Algorithm 2.3 DeactivateAdaptationSR Description:

This algorithm is designed to deactivate unnecessary security services. To achieve this task, the algorithm monitors and classifies the inbound/outbound traffic and maps the protocol and the port numbers to MTIDS policies presented in Table 2.1 to figure-out which set(s) of rules should be deactivated. In this algorithm, MSR SRQt,SRt, and ReqPriod, are used as inputs, as shown in line 1. ReqPriod depicts the deactivation time which is configurable. Hence, tenant can set run-time to deactivate the unneeded security services(Set of Rules). In this algorithm, Δt serve as a timer to determine the period of time for regular checking. For example, let's assume that Δt is configured to 24 hours. This means that the algorithm will do the regular checking each 24 hours. timeCapture,in line 2, will contain the recent time of captured packets. Line 3, is intended to capture the actual the time. After recording the time, the algorithm starts capturing the packet as shown in Line 5. While in Line 7, the update enables the timeCapture map to update the time of the entry that its key has the value of typePk and it will add new element, the element has a key which is the value of typePk, in case this the entry point is not listed. Lines 8, and 11, the algorithm compares the recent time by the time determined for checking. Finally, in lines 12 to 15, the security services will be deactivated if the condition or policy

2.7 Implementation and evaluation results

The main objective is to evaluate the performance of the OIDS by examining its capabilities in terms of CPU consumption.

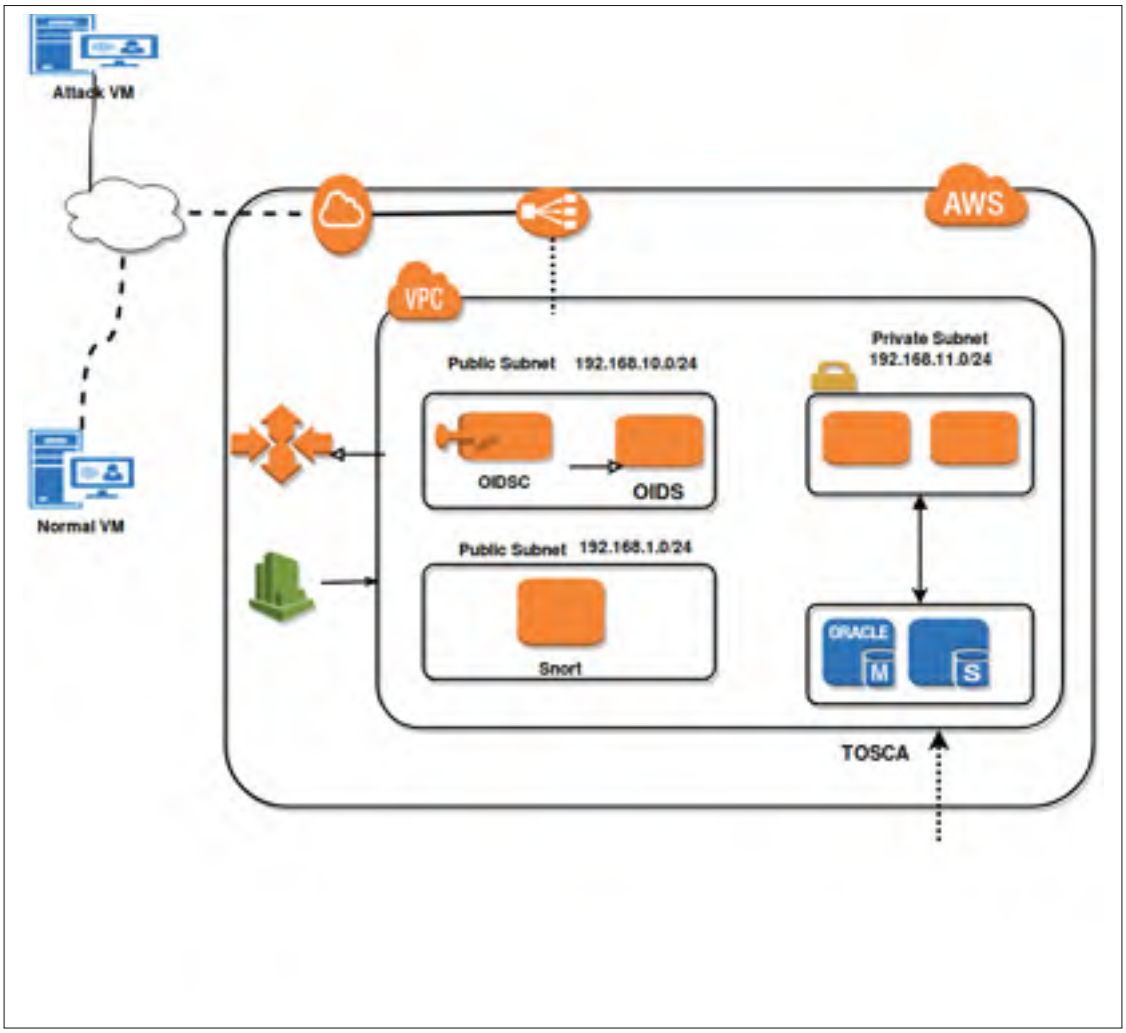


Figure 2.8 Network Architecture

2.7.1 Network Architecture Setup

Our MTIDS is deployed on top of a public cloud. Deploying the instances on top of the AWS cloud goes through several stages including the deployment of Virtual Private Clouds (VPCs), choosing an Amazon Machine Image (AMI), creating security groups, subnets etc. As it is depicted in figure 2.9, the targeted application contains a number of VMs (VMs) namely: a web server and database server deployed in different private subnets. First, we have created a virtual private cloud (VPC) (Amazon, 2016b), which is an isolated virtual network reserved for AWS tenant accounts. Second, on top of the VPC,

we create two subnets (private and public subnets) to launch EC2 instances. A web server (Apache 2.0 and database server (MySQL) are deployed in the private subnet. OIDS is located in the public subnet. The OIDS is a network intrusion detection system that acts as a gateway for the application deployed on the public subnet. Third, a security group acts as a virtual firewall that controls the traffic for at least an instance and determines the permitted network services that can run on each VM. It is also used to enforce the inbound/outbound traffic to pass through the Optimized IDS, which is connected to the instances located in the private subnets. The OIDS can span across different VPCs, regions, and availability zones. Hence, it can provide a collaborative OIDS. Fourth, the internet gateway will enable internet access for the public subnet. Elastic IP is assigned for the OIDS. This allows the instance placed in the public subnet to be accessed from the internet through an internet gateway. We use an open source intrusion detection system called Snort (Snort, 2015) to build the Optimized IDS' Analyzer Engine of the Optimized IDS. Snort is used as a packet sniffer to monitor network traffic in real time. It then carefully inspects each incoming/outgoing packet carefully to detect any serious payload or suspicious anomalies. Microsoft SQL Server (microsoft, 2016) is used to build the warning storage. Snorby is used as a graphical interface to demonstrate different information and statistics about the detected attacks. TOSCA (Oasis, 2017) is used to build the (TSRS) which allows tenants to specify their security requirements and create their topology. The specifications of all VMs as well as the attacker's machine that is used in our trials are both shown in Table2.3.

The main objective of our approach is to reduce the cost and to meet the tenants' needs in terms of security. Indeed, understanding the performance of the IDS resources is very valuable for managing both the IDS capacity and cost. Since we are seeking to meet the variety of the tenant in terms of security requirements, we compared the default snort,where all set of rules are activated, with our OIDS loaded with a set of rule at time. The names of the used sets of rules are illustrated in Table 2.4. Python Code is written for computing the CPU utilization and for illustrating these differences.

Table 2.3 Description of Virtual instances deployed on AWS

Software	**O**S Ubuntu	**T**ype	**M**emory	**S**torage(GB)	**v**CPU
OIDS	16.04 LTS	t2.small	1	32	1
OIDS	16.04 LTS	t2.medium	1	32	2
Snort	16.04 LTS	t2.small	1	32	1
Snort	16.04 LTS	t2.medium	1	32	2
Web Servers	16.04 LTS	t2.micro	1	32	1
DB Servers	16.04 LTS	t2.micro	1	32	1
Kali	16.04 LTS	t2.micro	1	80	1
Traffic-Generator	16.04 LTS	t2.micro	1	80	1
TRSM(ODISC)	16.04 LTS	m3.medium	3.75	80	2

Table 2.4 Rules Sets

Rules Sets Name				
DOS	IMAP	DDOS	ftp	ICMP
Dns	p2p	Misc	Netbois	nntp
Delete	Tftp	Virus	Sql	
Multimedia	Web-cgi	Telnet	Snmp	

Moreover, since the price of instances (VMs) relies on the number of CPU cores in within the instance, two different instances were used to demonstrate how our approach reduces the cost. Table 2.3 demonstrates the features of the two different VMs that are used on our trails. As it can be seen, t2.small instance is designed as a core machine and provides a computational power that is less, compared to the power provided by the dual core machines (t2.medium).

2.7.2 Workload

Workload is employed to evaluate the monitoring performance overhead of the IDS. In our experimentation, three scenarios were conducted for evaluating the MTIDS, more specifically, the OIDS. Overall, we used both normal traffic and combination of normal traffic and malicious traffic to evaluate the performance overhead of the optimized IDS(OIDS) by loading different services (sets of rules)

- First Scenario(Sending Normal traffic): in this scenario, a legitimate traffic was generated. Iperf3 (Iperf, 2016), which is a tool used for measuring the network performance (mainly used for measuring achievable bandwidth on IP networks), was employed for generating a background traffic. Iperf3 is installed on both the server and client, where the server receives packets sent from the client on a fixed port. The traffic generator is used to to evaluate the optimized IDS with various sets of rules individually loaded at time. The gained results are compared with Snort IDS where all rules sets are activated).

- Second Scenario (Sending a combination traffic): In this scenario, the optimized IDS is exposed to a mixed traffic (legitimate and Malicious traffic). The CPU utilization is then monitored and plotted to be compared with the first scenario. A traffic was generated using iperf3. Afterwards, the attacks are randomly generated after an IDS is started. Kali Linux is used as a tool for SYN- flooding the Web server with a malicious traffic. According to (Chapade *et al.*, 2013), servers are exposed to the thread of SYN floods as a result of an attack that sends SYN packets sourced by fake IP address. This leads to the servers to handlinge these packets as real connection requests. Thus, servers are triggered to initiate a half-open connection by sending back a SYN-ACK packet (Acknowledgement), and then wait for a reply from the sender's address. In this case,; since the IP used for the attack is a fake one, the server is likely to never receive a response, which prevents it from responding to the legitimate users' requests.

- Third Scenario: (Generating different types of traffic): In this scenario, we generated different traffic in order to validate the effectiveness of the OIDS that are used a novel mechanism taking into account classifying traffic in which allows offering better security mechanism. After generating the traffic, flow logs are used to identify the traffic type, and based on that, an algorithm activates the appropriate security service. In essence, OIDS analyzes the network packet to determine whether these packets matches the MTDIS policies, discussed in Table 6.1. Based on these policies, the OIDS either activate or deactivate security services.

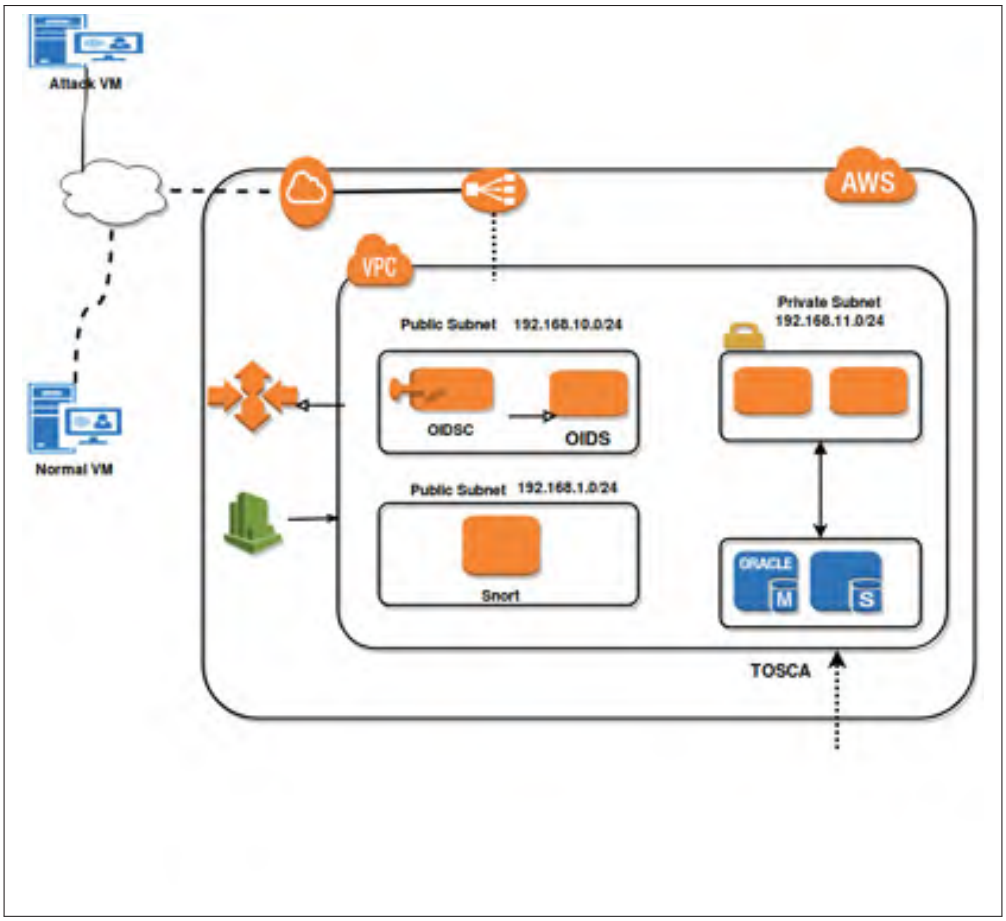
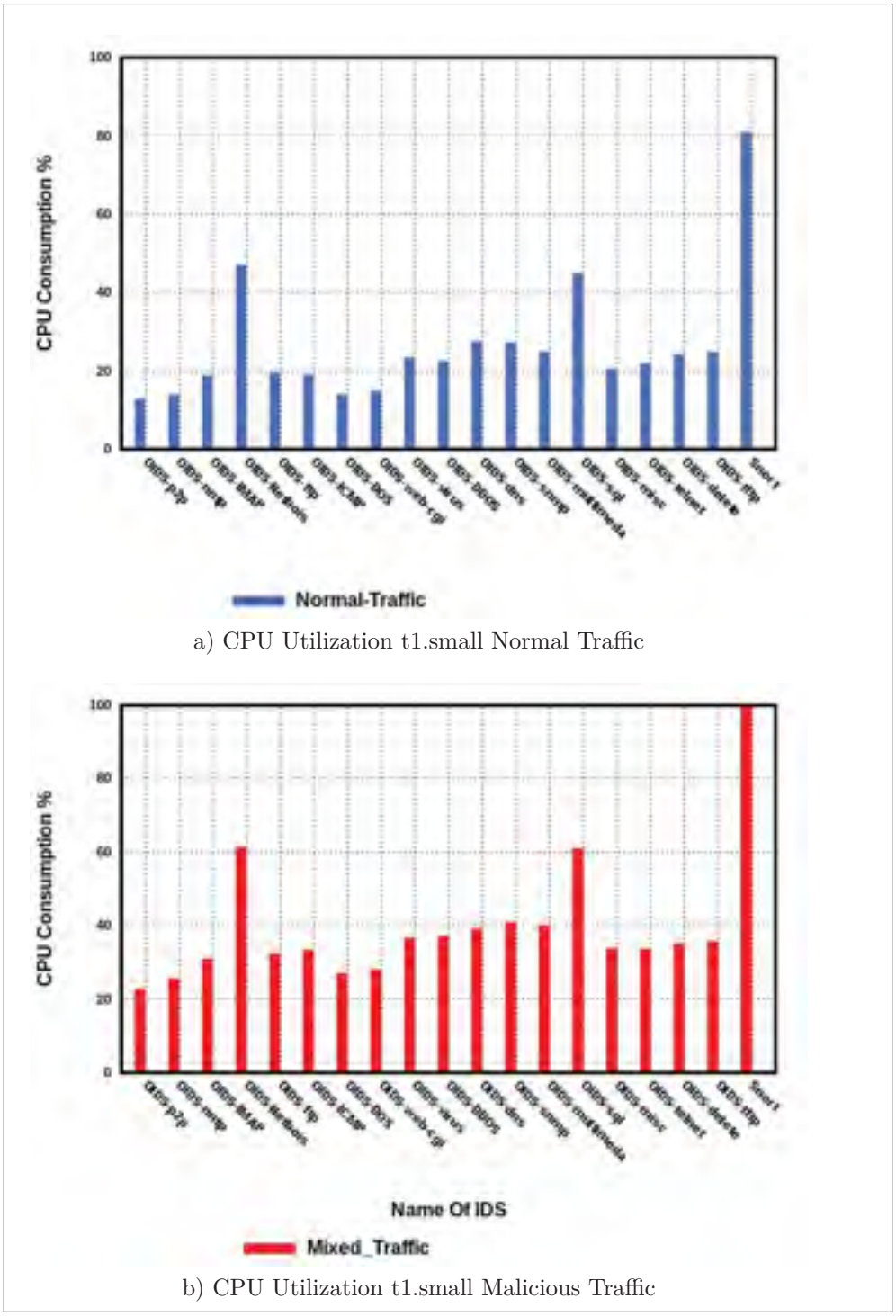


Figure 2.9 Network Architecture

2.7.3 Finding

As previously mentioned, various experiments were conducted to justify our proposed approach. The first experiment is to compare the default Snort with the OIDS loaded with individual Set of rules at time. Through this experiment, we have proven that increasing the number of set of rules does not necessary leads to increasing the efficiency of the security mechanism; rather it increases the resource consumption as shown in Figures 2.10a , 2.10b and 2.11a, 2.11b. Generally, these figures present the result of our experiment with the OIDS and Snort in AWS. They reveal a CPU consumption comparison between the OIDS, which loaded by an individual set of rules and Snort IDS simultaneously, while they are sniffing a normal and malicious traffic.

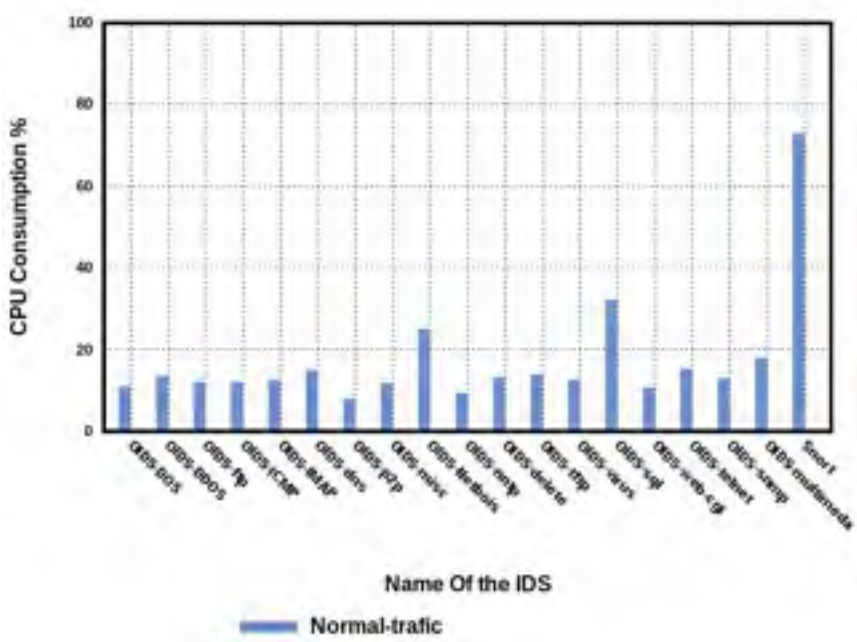


a) CPU Utilization t1.small Normal Traffic

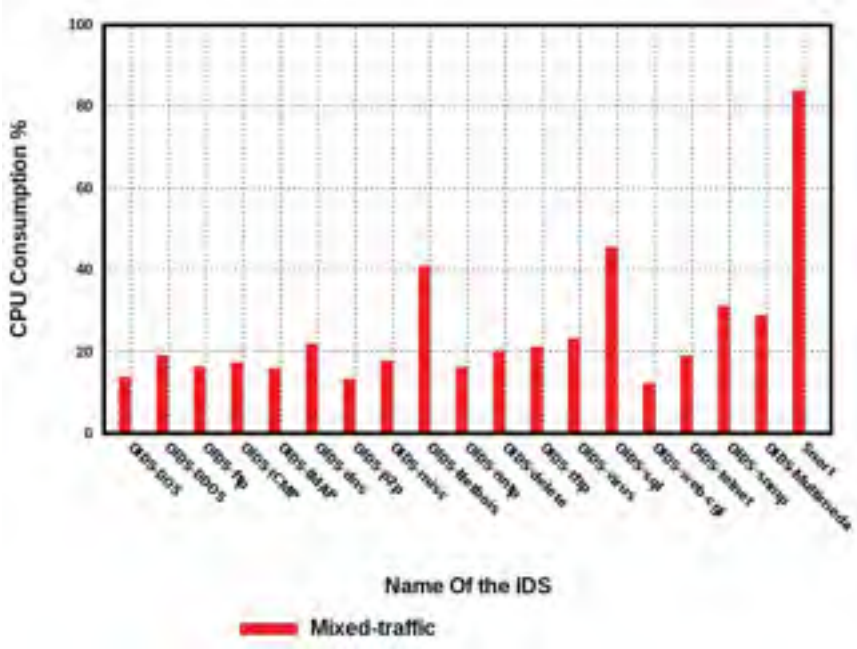
b) CPU Utilization t1.small Malicious Traffic

Figure 2.10 Compares CPU Consumption between Snort vs OIDS

As shown in Figure 2.10a and 2.10b, the p2p set of rules utilizes the CPU the least among all other sets by around 13 % for the normal traffic and around 22 % for the



a) CPU Utilization t2.medium Normal Traffic



b) PU Utilization t2.medium Malicious Traffic

Figure 2.11 Compares CPU Consumption between Snort vs OIDS

mixed traffic(normal and malicious traffic). OIDS with Dos, web-cgi and nntp set of rules

consumes a bit more by around 14 % compared to the p2p for the normal traffic and 29 % when a mixed traffic is generated. The Ftp, ICMP, and IMAP set of rules loaded to the OIDS recorded more CPU consumption by around 19 % for the normal traffic; whereas the CPU shows more increase roughly from 30 %to 33 % in the malicious traffic.

The consumption of CPU for the OIDS loaded with DDOS, dns, telnet,snmp,multimedia, tftp, and virus set of rules computed from 22% to 27% for the normal traffic while it recorded from 35 % to 37 % for the mixed traffic . The CPU consumption increased when the OIDS loaded with Sql, Netbois approximately 46 and 60 for the normal and the mixed traffic respectively. On the other hand, in Snort the CPU consumption is the highest as it reaches slightly over 80% usage for the normal traffic and 100% for the mixed traffic. The consumption of CPU for the OIDS loaded with DDOS, dns, telnet,snmp,multimedia, tftp, and various set of rules computed from 22% to 27% for the normal traffic while it recorded from 35 % to 37 % for the mixed traffic . The CPU consumption increased when the OIDS loaded with Sql, Netbois approximately 46 and 60 for the normal and the mixed traffic respectively. On the other hand, in Snort the CPU consumption is the highest as it slightly reaches over 80% usage for the normal traffic and 100% for the mixed traffic.

The same experimentation was conducted similarly; but with more powerful instances of type t2 medium instances. As shown in Figure 2.11a, 2.11b, Snort IDS exposes the highest CPU consumption rate when normal and mixed traffic was sniffed by snort. Whereas the OIDS loaded with p2p set of rules records the lowest CPU consumption rate. While the OIDS that individually loaded with Dos, and web-cgi concurrently utilizes around 11% and 14 % of the CPU for normal and mixed traffic respectively. The CPU consumption illustrates a bit increase with Ftp, ICMP, and IMAP set of rules by around 13 % for the normal traffic and around 17 % for the mixed traffic. Moreover, OIDS with DDOS, dns, telnet,snmp,multimedia, tftp, and virus set of rules exhibits a CPU consumption ranges from 13% to 15 % for the normal traffic. While in the mixed traffic it goes from 19 % to 25 %. In case of OIDS with solely Sql, Netbois set of rules at time, the recorded CPU utilization ranges from 25 % to 32 % in the normal traffic and from 41 % to 46 % in mixed

traffic. The highest CPU consumption was recorded for Snort IDS with both normal and mixed traffic. Overall, Pre-defined OIDS loaded with several set of rules simultaneously consumes fewer resources compared to Snort IDS. Hence, the smaller number of rules, the less resource consumption will be. To accomplish the third test scenario discussed on subsection 2.7.2, various type of traffic was generated for a 120 seconds. Figure 2.12 illustrates the CPU utilization of the OIDS(green line) compared to Snort(red line-points). In the figure, the x-axis represents the time per second whereas the Y-axis represents the percentage of CPU consumption.

In the beginning of the trial, by default all OIDS security services are turned on. We generated a traffic that is not included in the OIDS policy. After this traffic being examined, OIDS keeps all security services activated. This leads to overloading the CPU as shown in Figure 2.12. Then after 19 second, we changed the behavior of the traffic by generating traffic at port 80 over tcp protocol. As a result, the OIDS turns off all set of rules except the DOS security Service as it is matches the OIDS policy illustrated in Table 2.1. As a result, the CPU consumption decreases to roughly 30 %. The same process of generating traffic is repeated again. The CPU consumption increased and becomes overloaded from second 62 to 90. Afterwards, it decreases to 30% from second 90 to 100. On the other hand, there were no changes for Snort CPU consumption; so it remains overloaded.

2.8 Discussion and Future Work

In this paper, we have introduced the tenants' requirements in terms of a security service and the current security level provided by cloud providers. We showed the advantage of employing our approach as a service as it overcomes the shortcomings of the traditional IDS. In fact, several factors encouraged us to introduce a new model of security. Firstly, there is the network architecture or topology as the cloud tenant can deploy their VMs in different locations. Secondly, the tenant's budget can be limited. Therefore, security services should be provisioned to meet various requirements including tenants' budget.

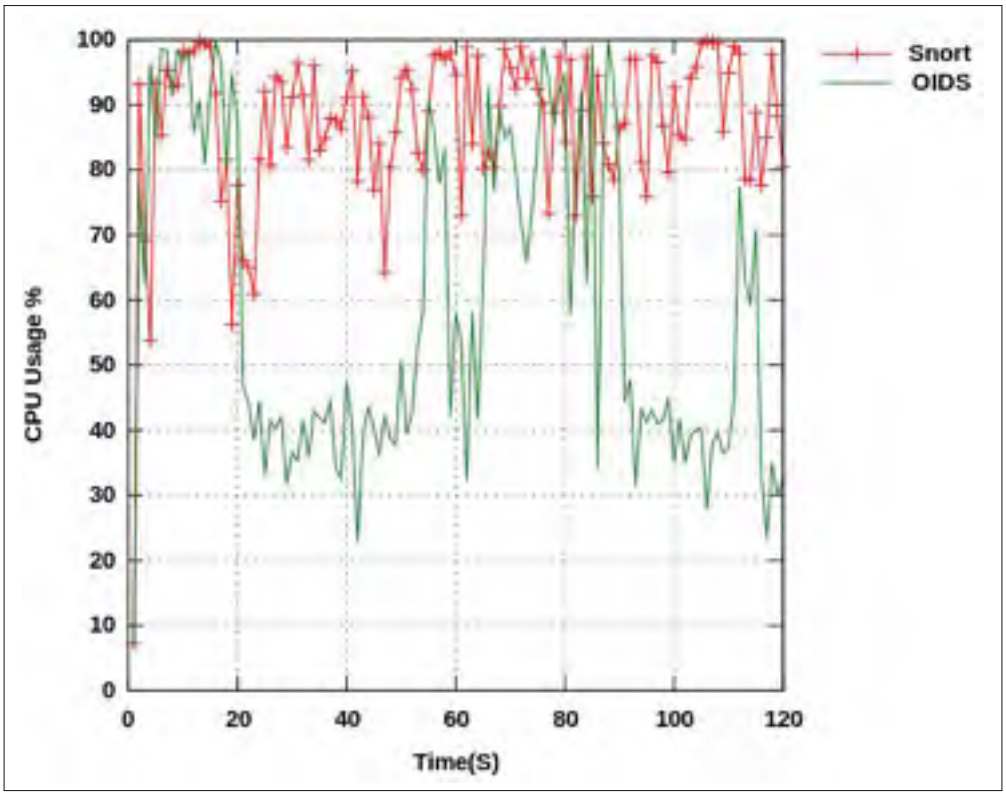


Figure 2.12 CPU Consumption of the OIDS vs Snort

For example, a tenant may need a specific security service. Hence, instead of provisioning a full package of security, providing part of the package for an affordable service fee would attract more tenants to adopt the proposed security framework. The experimental results show the value of the adaptable MTIDS as it proved that it is a cost-efficient framework compared to existing approaches. As the number of cloud tenants increases every passing day, such an approach is immensely needed to meet the tenants' requirements in terms of security service level as well as cost. This approach also allows the security service provider to maximize their gain as well as utilize their resources more efficiently. The obtained result gives us a hint about the CPU consumption which leads to the following benefits:

- Taking appropriate action in terms of the time needed to scale up/down instances.

- The CPU consumption of each sets of rules. Hence, if the OIDS consumes more resources, OIDS administrator can predict the CPU consumption issues in advance. Therefore, an immediate action can be taken to fix the issue.

As it is demonstrated in our experimental results, MTIDS is able to achieve re- source consumption reduction. This benefits the service provider as the number of instances is decreased. The experimental results showed that our approach can use an affordable small VM of t2. small type, which costs around $200 instead of using a more powerful VM of t2. medium type, which costs around $400. As our results also demonstrated, when IDS is deployed and launched with all of its features, the resources are over-utilized instead of functioning efficiently. The increase of resource consumption results in scaling up and deploying more VMs to process the traffic as the IDS will drop a large number of packets due to high CPU consumption incurred by passing by all sets of rules. Consequently, tenants ought to pay more for the increased number of instances. Our proposed solution solves these issues by selectively opting out the unneeded security services. It is able to reduce the resource consumption of the tenant while providing better security services. In our approach, we only concentrated at 20 sets of rules among all IDS rule sets. We will further analyze the remaining sets of rules in our future work. Additionally, we are currently considering different data mining techniques as well as pattern recognition to recognize anomalies and known intrusions. Hence, different machine learning techniques including ,for example Decision Tree(DT), Random Forest(RF), Support Vector Machine(SVM) will be used to accomplish this.

2.9 Conclusion

Adopting cloud computing as a new technology and a paradigm for the new era of computing has recently become widespread and intriguing within enterprises. The number of end-users is growing tremendously every day as users move their personal data to cloud storage services. For some, security remains a major concern without bearing in mind the high cost for security service provided by cloud service providers. In this paper,

we introduced a scalable, elasticity, on-demand pay as you use Multi-tenant IDS as a service framework that is offered for cloud tenants and security service providers. This proposed framework aims at reducing the security service cost for the tenants (end-users) and maximizes the security service providers' profit. It also enables them to efficiently manage their resources. MTIDS is designed to meet the tenants' needs by enabling them to specify their security requirements. Our framework is able to automatically activate and deactivate the sets of rules based on the tenants' needs. We validated the features of MTIDS through extensive trials on AWS. Our results showed that MTIDS effectively reduces the cost and meets the budget of tenants.

CHAPTER 3

ARTICLE 2: MULTI-TENANT ANOMALY INTRUSION DETECTION SYSTEM(MAIDS)

Mohamed Hawedi[1], Abdi Ramy [1] , Chamseddine Talhi [1] , Hanifa Boucheneb [2]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Computer Engineering and Software Engineering, Polytechnique Montréal
2900 Edouard Montpetit Blvd, Montreal, QC H3T 1J4

Article Submitted « Computers & Security (Elsevier) » April 2019.

The internet enables the provision of cloud services, and, as it is continually under attacks, cloud services are being exposed to an increasing number of attacks. New and sophisticated attacks have become endless. Consequently, there is a need for developing more flexible and adaptive security approaches to preventing attacks. Anomaly intrusion-detection systems (IDS) are considered valuable for protecting critical infrastructure against suspicious activities and theft of data. Different approaches have been proposed. They are, however, still unable to provide an appropriate security mechanism. In this paper, we present a new anomaly intrusion detection paradigm — the Multi-tenant Anomaly Intrusion Detection System (MAIDS) — that is capable of detecting intrusive attacks and that can automatically generate new signatures from the attacked system. The proposed approach uses a tenant's environment configuration to automatically generate customized set of rules. This can significantly improve the efficiency of the IDS and can ensure the extraction of valuable information from the monitored network.

3.1 Introduction

Cloud computing is a model for enabling ubiquitous and convenient access to a shared pool of computing resources — networks, servers storage, applications, and other services — with minimal configuration or management efforts Mell & Grance (2011). It facilitates

access to persistent data and information on remote servers. The growth of the internet and the ever increasing demands of the users continue to drive advancements in information technology. This is contributing to making cloud computing a useful and desirable technology to tenants. It has encouraged many organizations worldwide to adopt cloud computing.

Cloud computing offers many benefits. It provides flexibility and scalability of services; it enables organizations to save cost; and it improves operational efficiency, agility, and environmental sustainability Chou (2015).

Infrastructure as a Service (IaaS) refers to the provision of dedicated computing infrastructure in the form of virtual machine (VM) deployment. The idea behind IaaS is to make the servers, networks, as well as the storage, available on demand to the cloud users. Amazon EC2 Amazon (2015) and Microsoft Windows Azure Azure (2018) are examples of the cloud service providers. Platform as a Service (PaaS) aims at providing computing platform-layer resources such as operating systems and software frameworks; an example is Microsoft Windows Azure. Software as a Service (SaaS) targets offering applications on demand on the internet Hawedi *et al.* (2018a). Salesforce salesforce (2017) is an example of SaaS providers.

All of these services — SaaS, PaaS, and IaaS — can be consumed without the need for any specialized knowledge of the various underlying technologies. In addition to providing different services, cloud computing offers different deployment models: private, public, community, and hybrid models exist Mishra *et al.* (2018). However, securing the cloud is still a major challenge. Indeed, as the systems run over the Internet, services provided by cloud providers are prone to different attacks (e.g. Denial of Service (DoS), Distributed Denial of Service (DDoS), and SQL injection) Hawedi *et al.* (2018b). Intrusion detection system (IDS) is used to monitor the inbound/outbound traffic and to analyze them for potential attacks. The IDSs are categorized based on their detection signature or rule and anomaly-based detection systems. In the signature-based technique, the system's

activities are analyzed by examining the events that match the signature that identifies the attack types. This kind of technique uses pre-defined rules against known attacks. In contrast, however, anomaly-based IDSs concentrate on monitoring and analyzing the behavior of the system, identifying the unusual and abnormal behavior. They assume that abnormal activities are different from authorized activities. Hence, these activities can be detected using systems that can distinguish between normal and abnormal activities. By constructing profiles representing the habitual behavior of users, hosts, and the network, anomaly detectors learn themselves Hawedi *et al.* (2018a). Anomaly detection aims at identifying the event that seems to be abnormal with respect to the normal system behavior Modi *et al.* (2013).

To summarize, anomaly detection systems rely on the expected behavior of the system to detect abnormality; and it is signaled any deviation from the expected behavior as anomalous. In anomaly-based approach, data relating to the behavior of legitimate users over a period of time is collected; then, statistical tests are applied on the collected data to determine whether that behavior is legitimate or not. This allows a tenant to detect attacks that have not been found previously. Thus, enabling the detection of an unknown or zero-day attack — the vulnerability that has not been known previously. One of the key elements of these techniques is the ability to efficiently generate rules in such a way that it can lower the false alarm rate for unknown, as well as, for known attacks Modi *et al.* (2013). With this regard, IDS becomes very important in terms of securing the cloud environment. Thus, Intrusion Detection System (IDS) has been proposed as a solution to cope or mitigate the security issues for both cloud tenants and providers.

However, in numerous cases, traditional IDS failed to detect a new attack. The existing IDSs have limitations in terms of extensibility and adaptability as they are not able to detect sequential behavior. Furthermore, the existing IDS rules are not effective for unique networks or organizations because they are designed for generic environments.

Creating custom rule set in an automated manner based on each tenant environment can help in improving the effectiveness of the IDS. Additionally, automation rule creation requires less knowledge and time consumption compared to the manual rule creation where the number of administrators who create rule sets are large. Hence, administrators would face many difficulties regarding valuable information extraction. By gathering the custom rules with the standard rules, the IDS can be customized to the unique networks or organization requirements. This results in increased security of the assets that are considered valuable and can be utilized to prevent data breaches that the existing IDS deployment could not.

In this paper, we propose Multi-tenant Anomaly-based Intrusion Detection System (MAIDS) that is designed to monitor the tenant system activities and classify them as either normal or abnormal. The MAIDS is capable of meeting the tenant's requirements. This is because MAIDS offers the flexibility and the freedom to select the IDS that fits its requirements. The proposed approach enables the generation of custom rules that resulted from the classification done by the anomaly IDS. Indeed, tenants who deploy the same application are more likely to get similar attacks. Hence, they have interests in sharing their findings in real-time. Consequently, the generated rules would be shared between tenants based on their interests. Exchanging or sharing the custom rules among tenants helps in decreasing the influence of the attacks. Moreover, working in a collaborative manner can help tenants to increase the ability to avoid new attacks by adding the detected attacks to the blocking rule table.

The remainder of the paper is organized as follows. We discuss the related work in Section 3.2. In Section 3.3, the cloud tenants' requirements are described. The architecture of MAIDS is described in detail in Section 3.4. In Section 3.5, we explain the setup of the experiments and discuss the results. We conclude the paper and present the future work in Section 3.6.

3.2 Related Work

In the previous section, we highlighted the problems and challenges to detecting attacks. Here, we review some work based on intrusion-detection systems that have been proposed by researchers to detect intrusions in the cloud. Many of the researchers proposed defense mechanisms based on misuse-intrusion detection systems.

Nikolai & Wang (2014) propose a Hypervisor-based Cloud Intrusion Detection System (HCIDS) to detect denial of service attacks within a cloud environment. In their work, they examine system metrics of the cloud instances directly from the Virtual Machine Monitor (VMM) or hypervisor, which hosts the virtual machines to find any potential misuse patterns. And these metrics include, network data transmitted and received , and CPU utilization, etc.

Lo *et al.* (2010) proposed a federation defense based on IDS that is deployed in each cloud computing region. The deployed IDSs work in collaboration by exchanging the alerts (messages) to decrease the influence of the DoS attack. The authors proposed a cooperative agent whose main task is to receive alert messages from the other IDSs deployed in each region. The accuracy of the exchanged alerts are determined by agents through an election: agents vote on the alerts. If the alerts are accepted by an agent, a new blocking rule is added into the block table against this type of packet attacks on the cloud regions by the system. This new blocking rule can provide advance attack detection to other cloud computing regions, except the victim.

Roschke *et al.* (2009) proposed an extensible IDS architecture that consists of several IDS sensors deployed on each VM as well as a remote controller that performs the management task of these sensors. Each deployed sensor is responsible for detecting and reporting any abnormal behavior and to dispatch the triggered events to the event gatherer component, whose main function is to collect the anomalies and store them in the event storage unit or a database for an additional analysis.

Other researchers proposed a hybrid intrusion detection by integrating anomaly and signature IDSs. Tupakula *et al.* (2011) targeted security cloud IaaS by integrating the signature-intrusion-detection approach with the anomaly-intrusion-detection technique. They deployed their hybrid-intrusion-detection systems on top the Virtual Machine Monitor (VMM) to be able to monitor all incoming/out-going packets and to identify the malicious entity that attempts to compromise the virtual machines. Ullah & Mahmoud (2017b) proposed a hybrid model that is built using machine-learning and data-mining techniques for anomaly-based intrusion detection. In their work, they targeted enhancing the capability of detecting network attacks, more specifically, by enhancing the accuracy, increasing the detection rate, through developing a feature-selection model for an anomaly-based intrusion detection. Modi & Patel (2013) merge the traditional signature-based intrusion detection system with the anomaly-based intrusion detection system to enhance the detection accuracy and consequently enhancing the capability of detecting the network attacks. Their approach uses different machine-learning approaches including Bayesian, associative, and decision tree to build the framework. In their work, the incoming/out-going traffic passing through the signature IDS is forwarded to an anomaly IDS for classification. Aljawarneh *et al.* (2018) proposed a hybrid classification-based intrusion detection model and a feature selection to help in detecting attacks over the network. First, the feature selection method is applied to NSL-KDD data set. Later, the n-intrusion detection model based on machine learning approach is built and used to find attacks. Moreover, the captured data is used to improve intrusion detection.

An anomaly IDS based on a feature selection algorithm was proposed by Ambusaidi *et al.* (2016). The proposed algorithm is a mutual-information-based algorithm that analytically chooses the optimal feature for classification.

The proposed feature-selection algorithm is used to build and train an anomaly IDS named Support Vector Machine based IDS (LSSVM-IDS). KDD Cup 99, Kyoto 2006+, and NSL-KDD data sets were used to evaluate LSSVM-IDS. In a similar context, Ullah & Mahmoud (2017a) proposed an anomaly IDS based on a filter-based feature-selection model. The

proposed classification model targets removing irrelevant and redundant features and improving the accuracy rate.

Most of the approaches proposed in the current literature have not considered meeting the tenant's requirements and updating the tenant IDS database signature with new custom rules that meet their requirements. In addition, the discussed approaches have solely concentrated on addressing the performance of the anomaly IDS in terms of the accuracy of the detection rate.

To conclude this section, none of the above discussed approaches is completely satisfactory in terms of providing a better security mechanism that is able to meet the tenant's requirements. Thus, there is still a need to develop a new framework that is able to fill the gap of the work discussed above. In a nutshell, our proposed approach provides several aspects that differentiate it from the work discussed earlier. The following are some of the unique contributions of our approach:

- Our approach is designed to meet the tenants' requirements. The security mechanism — an anomaly IDS — is trained based on the tenant's security requirements. For instance, a tenant who is interested in DOS protection will get a DOS anomaly-IDS module while a tenant that is interested in, for example, SQL injection protection, will get a trained a SQL anomaly-IDS module. A tenant can have an anomaly IDS that has been trained for all attacks too.

- In our approach, we use a new data set that contains network traces of modern attacks for training and evaluating the anomaly IDS. CIDDS Ring *et al.* (2017b) would be used in this approach to train and validate the anomaly IDS.

- We extend the traditional IDS with new rules that play a vital role in increasing the detection of the attacks that cause damage to the tenant's resources. Thus, unlike the previous work, generating custom rules are a very important aspect of our approach.

- We provide an advance protection for the tenants as our approach encourages the tenants to work in collaboration in terms of changing the information regarding the source of an attack.

- Beside generating custom rules, the latency of the framework is also a vital aspect of this approach. We consider the time starting from the classification till the time new rules are dispatched to the tenant.

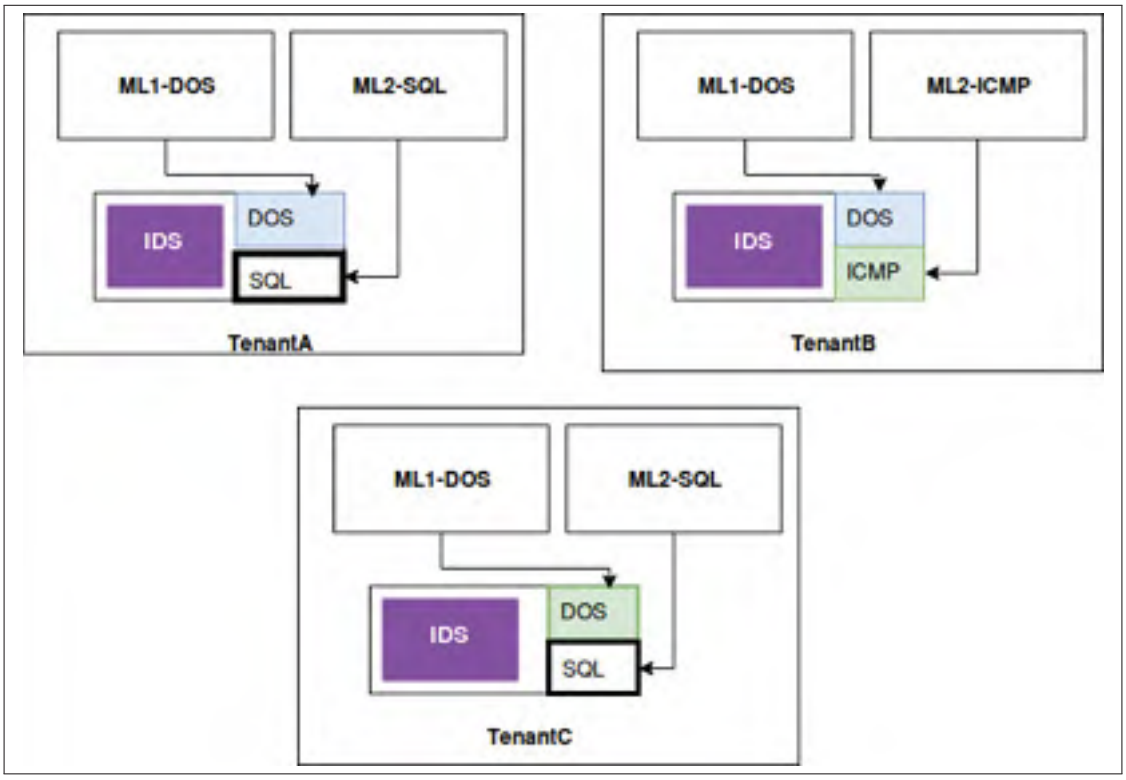


Figure 3.1 Cloud Tenant Security Preferences

3.3 Cloud Tenant Requirements

Tenant is a term commonly used to denote a client that uses a particular service from a cloud computing environment to satisfy an information technology (IT) requirement. The client might be an individual, an organization or a business unit Magar (2012). Multi-

tenant concerns the concept of sharing or using the same set of resources by multiple clients. For instance, different companies use Cumulus to host a number of instances for deploying a number of customer-facing applications . In this case, Cumulus is considered a multi-tenant environment Magar (2012). These companies might have different units or branches (sales, and marketing business units, as an example) where they provide different services; hence, deploying different IaaS topology. Thus, cloud tenants want to have the optimized security arrangement that meet their requirements as they deploy or have different architectures on top of the cloud. The capability of understanding and classifying the tenant's needs enables the provision of a security mechanism with a low cost. Based on the architecture deployment in the cloud, each tenant requires a specific security mechanism. Based on this context, we assume that we have different tenants who need to deploy different security mechanisms to protect their cloud resources. As shown in Figure 3.1, *TenantA* requires DOS, SQL protection while *TenantB* wants to protect his/her cloud resources from DOS and Telent. *TenantC* shares the same security interest with *TenantA*, as DOS and SQL injection are their main concerns.

Based on this assumption, there is a need to provide a security mechanism that takes into consideration the tenants' preferences in terms of the security that meets their deployment architectural perspectives. Our approach allows cloud tenants to make substantial gains in terms of the reduction of cost, security, and update times. Our approach provides a real-time anomaly IDS that is designed to monitor the activities of the system and then classify them as either normal or abnormal (anomalous). Moreover, the proposed approach offers a trained anomaly IDS to tenants based on their demands. In this context, as illustrated in the Figure 3.1, *TenantA* and *TenantC* will get a trained DOS, and SQL anomaly IDSs. While, *TenantC* will receive a trained DOS, and Telnet anomaly IDSs. Our approach offers the flexibility to the cloud tenants to choose one or more anomaly IDSs that meet their needs. Indeed, our approach uses different algorithms including Decision Tree (DT), Random Forest (RF), and Neural Network (NN) algorithms to enable offering the various anomaly IDSs.

Our approach offers the tenant the possibility of automatic creation of custom rules based on tenants' environment which can be added to a set of generic rules or signatures. These rules are sent quickly to tenants based on their security preferences or subscriptions. Our approach enables the communication between tenants based on their security interests. For instance, in Figure 3.1, *TenantA*, and *TenantC* shared the same security interest: they are both looking for DoS and SQL protection. In this case, each time tenant activates/deactivates a security service (DoS), a subscription message will be sent to *Rule Handler* indicating that these tenants are activating/deactivating DoS set of rules. Based on this, the tenants will be able to receive new rules generated based on the classification.

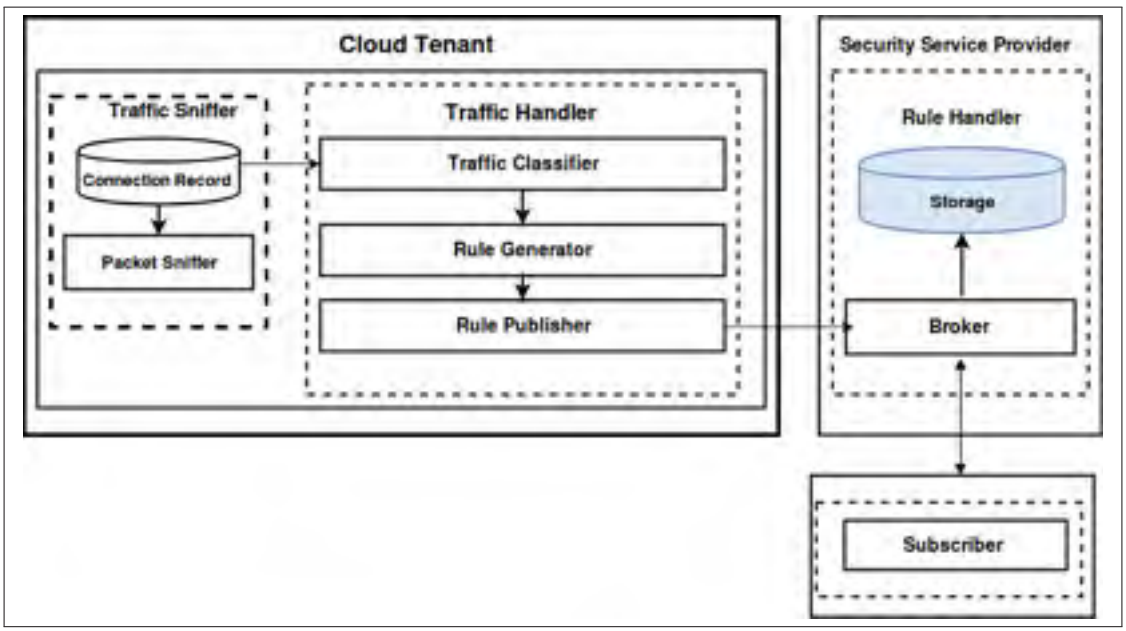


Figure 3.2 MAIDS Architecture

3.4 MAIDS Architecture

The existing security solutions are far from satisfactory to stop the rapid growth in the number of cyber-attacks. Meeting the tenants' requirements in terms of security and

budget helps in providing better security Hawedi *et al.* (2018a). Thus, in our previous work, we proposed a multi-tenant intrusion detection system, MTIDS, as a service that targets the security of the public cloud, where it is designed to provide an appropriate and optimized security that considers the tenants' needs in terms of the security service requirements and budget. MTIDS provides an optimized signature IDS that is designed to adapt to changes in the tenant's environment. It follows certain important policies to activate/deactivate the IDS signatures. The mechanism of MTIDS is briefly explained in the following:

(i) A tenant select its security service interest using the *Tenant Security Manager* component. For instance, some tenants may be interested in Denial-of-service attack (DOS) protection while others are looking for SQL injection attack protection.

(ii) MTIDS analyzes the traffic looking for any change that occurs on the traffic.

(iii) If a change in traffic is detected, MTIDS activates/deactivates the security service based on the change.

MTIDS is a rule-based IDS. It is unable to detect new types of attacks or even any known attacks with a small change in their signatures. Thus, there is a need to provide a security approach that is able to detect an unknown attack. This approach is an extension to the previous approach (MTIDS) as it offers various anomaly IDSs that considered the tenants' requirements.

This section presents the architecture of MAIDS. The main idea is to build a comprehensive security mechanism that is able to protect a company's data and critical infrastructure against unknown suspicious activities, loss and manipulation by unauthorized parties. As depicted in figure 3.2, MAIDS consists of four main components: *Traffic Sniffer*, *Traffic Handler*, *Rule Handler*, and *Subscriber*.

1. **Traffic Sniffer:** The main task of *Traffic Sniffer* is network monitoring; that is, to capture packets without losses and to provide accurate time-stamps. Thus, software based on hardware packet capturing systems are required to ensure that all traffic is captured and stored. The packet sniffing or capturing is an important procedure to many network applications. In this step, the network traffic is sniffed and then dumped straight into a storage to be analyzed *Traffic Classifier*.

2. **Traffic Handel:** This component contains three important elements namely: *Traffic Classifier*, *Rule Handler*, and *Rule Publisher*.

    a. **Traffic Classifier (TC)** : *Traffic Classifier* is the main component that categorises network packets into either normal or abnormal. It is an anomaly IDS that uses various machine-learning algorithms that enable tenants to meet their requirements in terms of choosing the anomaly IDS that fits their cases. In our approach, three supervised machine-learning algorithms including Random Forest (RF), Decision Tree (DT), and Neural Network (NN) were used individually to build different classification models. These selected techniques (DT, RF, and NN) were chosen as they give the best average performance Caruana & Niculescu-Mizil (2006). Decision Tree (DT) is a type of supervised learning algorithm that is commonly used in data mining, more specifically, in classification problems as predictive model that can be used for representing classifiers. RF is a supervised machine learning classification algorithm that is used to overcome problems related to classification and regression.

    b. **Rules Generator (RG):** As known, cloud providers offer different services to different tenants. Thus, using the general rules, as most existing IDS systems do, created for a generic environment to secure the tenant resources is not very effective. This necessitates the creation of custom rules in an automated manner based on each tenant's environment.

    In our approach, RG is designed to generate customs rules based on the traffic classification obtained for the classifier component discussed on Subsection a..

The main objective is to generate up-to-date new rules to overcome the limitations of the existing IDS systems.

Indeed, the IDS use signature or rule matching on the traffic of the network. The structure of the IDS rules such as *Snort IDS*, for example, is split into two parts: *rule header*, which describes the packets' attributes and *rule option*, which alerts a message about the matched packets. The rule header contains different parts including, the action (e.g., alert, drop, etc), protocol (IP, UDP, TCP, ICMP), and other relevant data that define the network packets.

Example 3 Hawedi *et al.* (2018a) reveals more details regarding the IDS rules where the rule header and the rule options are discussed in details:

Example 3. *alert tcp EXTERNAL_NET 80 −> HOME_NET 8080 (msg:"ICMP test"; sid:1000001; rev:1; classtype:icmp-event;)*

This rule examines any traffic directed toward the network and will generate an alert whenever the IDS detects traffic headed inbound from outside to the network over the port (source) 80. Snort rule contains two parts: the rule header and the rule option. The syntax of the rule is explained below SNORT (2018).

1. **Rule Header:** In the rule header (`alert tcp EXTERNAL_NET 80 -> HOME_NET 8080`), `alert` represents the rule action to be taken by the IDS. The IDS will generate an alert if the condition is met; the string `EXTERNAL_NET` represents the source IP and `80` is the port number. All the external traffic from the source IP over port number 80 will be considered by the IDS. The arrow (`->`) indicates the direction from the source to the destination. The `HOME_NET 8080` refers to the destination IP (`HOME_NET`) and its port number (`8080`).
2. **Rule Option:** In this example, the rule option is represented as (`msg:ICMP test; sid:1; rev:1; classtype:icmp-event;`), where the message (`msg:`) is (`ICMP test`), will be included with the generated alert. The string `sid:1;` represents the rule number or ID, which should be unique. The string `rev:1` gives the revision number. This option facilitates the maintenance of the rule. Lastly,

the `classtype:(icmp-event)`, in this example, is used to classify the rule as an 'icmp-event'. This classification helps in the organization of the rules.

c. **Rules Publisher (RP):** In multi-tenant intrusion detection system environments, the requirements of the tenants on their security services are totally different and are distributed. Moreover, the collaboration or communication among the tenants in terms of sharing common events (rules) is fundamental. Our approach provides a significant advantage in terms of instant notifications of events for these different and distributed security systems. It is considered as a powerful model for disseminating the events from the publishers (which is the data-event producer, which cloud be tenant or security-service providers) to subscribers (which is data-event tenants). Indeed, MAIDS provides an event-driven feature that is represented in the *Rule Publisher*, *Broker*, and *Subscriber* components. This feature provides increased performance, reliability, and scalability for the security services. In our approach, RP can be either tenants or security service providers that generated rules based on the classification model. The RP components receive all new rules that are generated by the RG component. The task of RP is limited to publishing or generating the stream of newly generated rule events.

3. **Rule Handler (RH):** The RH component can be centralized or federated and its responsibility is to deliver the rules sent by the publisher to the right tenant. It comprises of two component named *Broker* and *Storage*.

   a. **Broker:** *Broker* is used to route the messages of the generated rules from the tenant publisher component to the tenant subscriber. Figure 3.3 gives the important information required by the broker to route the generated rules between the tenants. First, *TenantA*, and *TenantC* share the same interest in terms of deploying DOS and SQL anomaly detection. Hence, both tenants will send a subscription notification to the broker as well as generate new rules after classifying the traffic. *TenantB* deployed DOS, ICMP anomaly IDS.

Consequently, it will send a notification message to the broker with a DOS-and-ICMP subscription. Based on this information, the broker builds a policy table that has the name of the tenants (*SubscriberNa*) and the security preferences (*Topic*) to facilitate the transfer of the rules to the right tenant.

Table 3.1 Broker Policy

SubscriberNa	**Security Preferences (Topic)**	
TenantA	DOS	SQL
TenatB	DOS	ICMP
TenantC	DOS	SQL

In this assumption, first, *TenantA* has activated DoS set of rule publisher, and it subscribes to DOS. Furthermore, *TenantA* will publish any newly generated rules as well as receive any DOS rules published by other tenants. *TenantB* is interested in protecting its environment from SQL; while *TenantD*'s main task is to publish events related to SQL rules. *TenantC* is interested in receiving all the rules generated by RG . It is clear that *Broker* can be centralized or federated: it has all of the chains of events including the topic (set rule names) and the subscriber (tenant name) that are used within the event flow.

b. **Storage** : The storage acts as a repository that allows the broker to store all the information of a subscriber (tenant), publisher (a tenant or a security provider), including their rule-set types.

4. **Subscriber**: A subscriber is deployed at each tenant's environment and listens to the event streams of the generated rules and then process these events by storing them based on their topic's name, and eventually generates notifications when the stream of rule events are received.

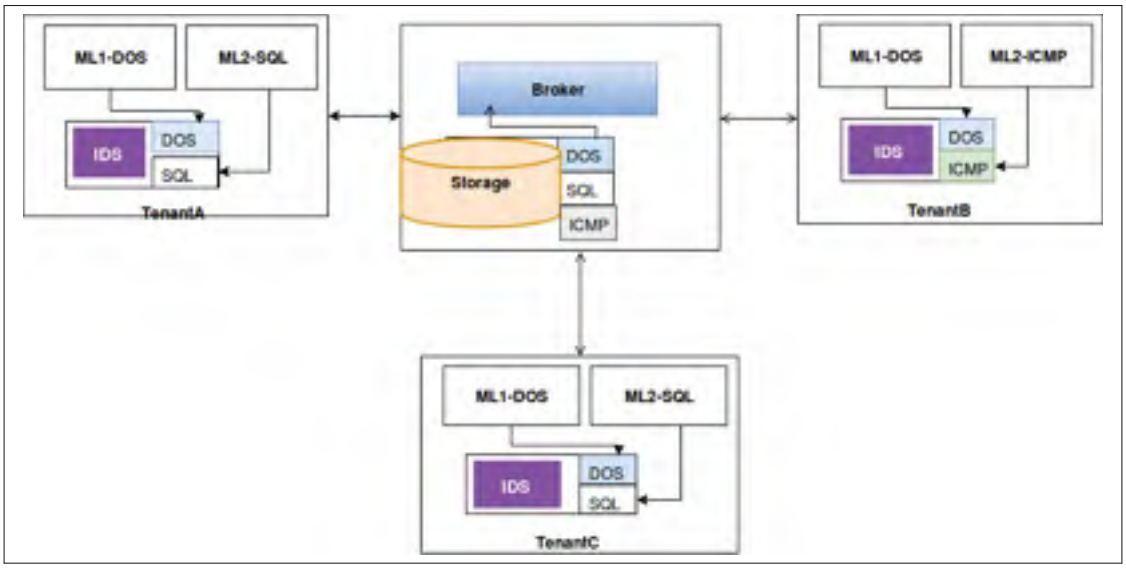


Figure 3.3 Broker Mechanism

3.4.1 The MAIDS Mechanism

In the previous sections, we mentioned that our approach is designed to enable tenants to prevent unknown attacks by generating custom rules and by quickly updating the tenants — who share the same interest — with the newly generated rules. Here, we provide a comprehensive illustration of MAIDS's process by describing how the components of MAIDS interact to provide a powerful security approach.

As shown in Figure 3.4, the incoming traffic is captured by the traffic sniffer (see message (1)). This traffic is handled as follows: (2) sends the captured traffic to the classifier model; (3) tenant selects one of the classifier models that is responsible for determining the status of the traffic, which can either be normal or abnormal; (4) new rules will be generated based on the result of the classification model; (5) the newly generated rules are saved in a storage; (6) the new rules are published by the rule publisher and the publisher dispatches these rules to the broker that has the lists of the information concerning the publisher and the subscriber; (7) rules are received by *Broker*; (8) *Tenant_Subscriber*

sends its security interests to *Broker*; (9) *Broker* sends or delivers the generated rules to tenants based on their security preferences.

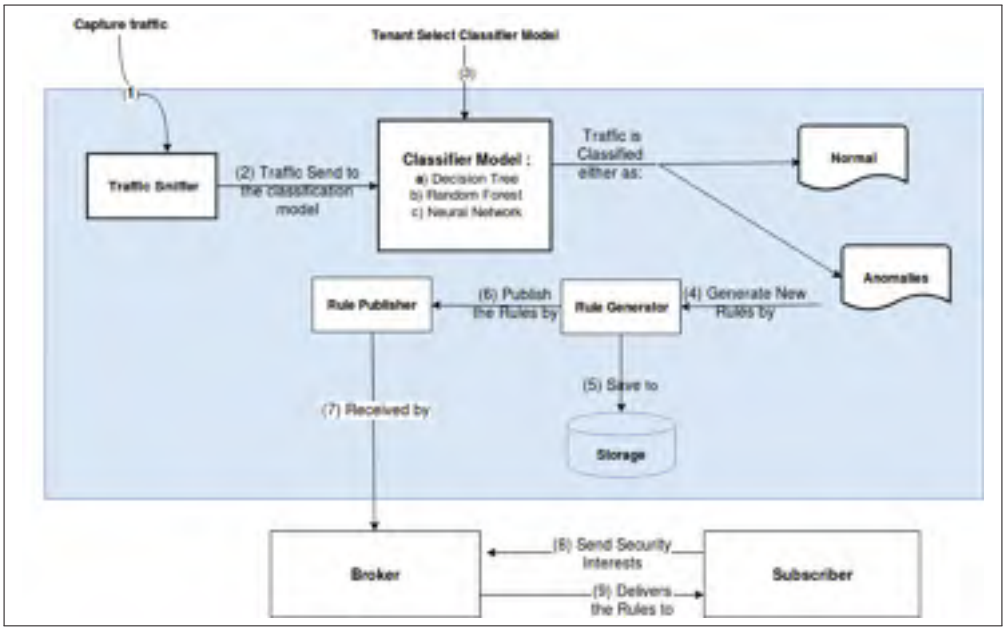


Figure 3.4 The MAIDS Mechanism

Table 3.2 Description of the Deployed AWS instances

Software	**T**ype	**M**emory	**S**torage(GB)	**v**CPU
TS (Bro)	t2.small	1	32	1
Traffic Classifier (TC)	t2.small	1	32	1
Rule Generator (RG)	t2.small	1	32	1
Rule Publisher (RP)	t2.small	1	32	1
Broker	t2.small	1	32	1
Subscriber	t2.micro	1	32	1
Storage	t2.micro	1	64	1

3.5 Implementation and Evaluation Results

Security service updates play a vital role in protecting the tenants' resources from latest threats. Generally, harmful attacks attempt to take advantage of the vulnerabilities in a security service. Signature-based intrusion detection systems have been used by different entities to overcome this issue. However, its weaknesses in terms of the inability to detect unknown or abnormal attacks justifies the need for the anomaly-based IDSs. An anomaly IDS improves the effectiveness of the system as its able to distinguish between a normal and abnormal activity, thereby increasing the ability of generating new rules. These generated rules need to be sent to tenants who share the same interest as some of them do not have an anomaly-based detection or might use a different machine-learning algorithm. Delaying the update of the security service keeps the door open for attackers to penetrate the system. Our approach provides some very significant features in terms of providing various an anomaly IDS based on the needs of the tenant. Furthermore, it provides a fast, automated, and up-to-date system to keep the security service safe and stable. Our proposed approach considers the time taken to update the security service because this is very critical in order to keep the security service safe and stable, and thus be trusted.

3.5.1 Anomaly IDSs Training and Validation

As previously mentioned, our approach uses a recent data set, which includes, the network traces of modern attacks, to train and validate the classification models used to build the anomaly IDSs.

CIDDS Ring *et al.* (2016) data sets were used to train our classification models and to evaluate the proposed anomaly intrusion detection system. CIDDS (Coburg Intrusion Detection Data Sets) is a labelled flow-based data set Ring *et al.* (2017b). The data set was created for evaluation purposes, specifically, for evaluating the anomaly-based network intrusion detection systems. To create this data set, they simulated a small business environment, including several clients and typical servers, like an e-mail server

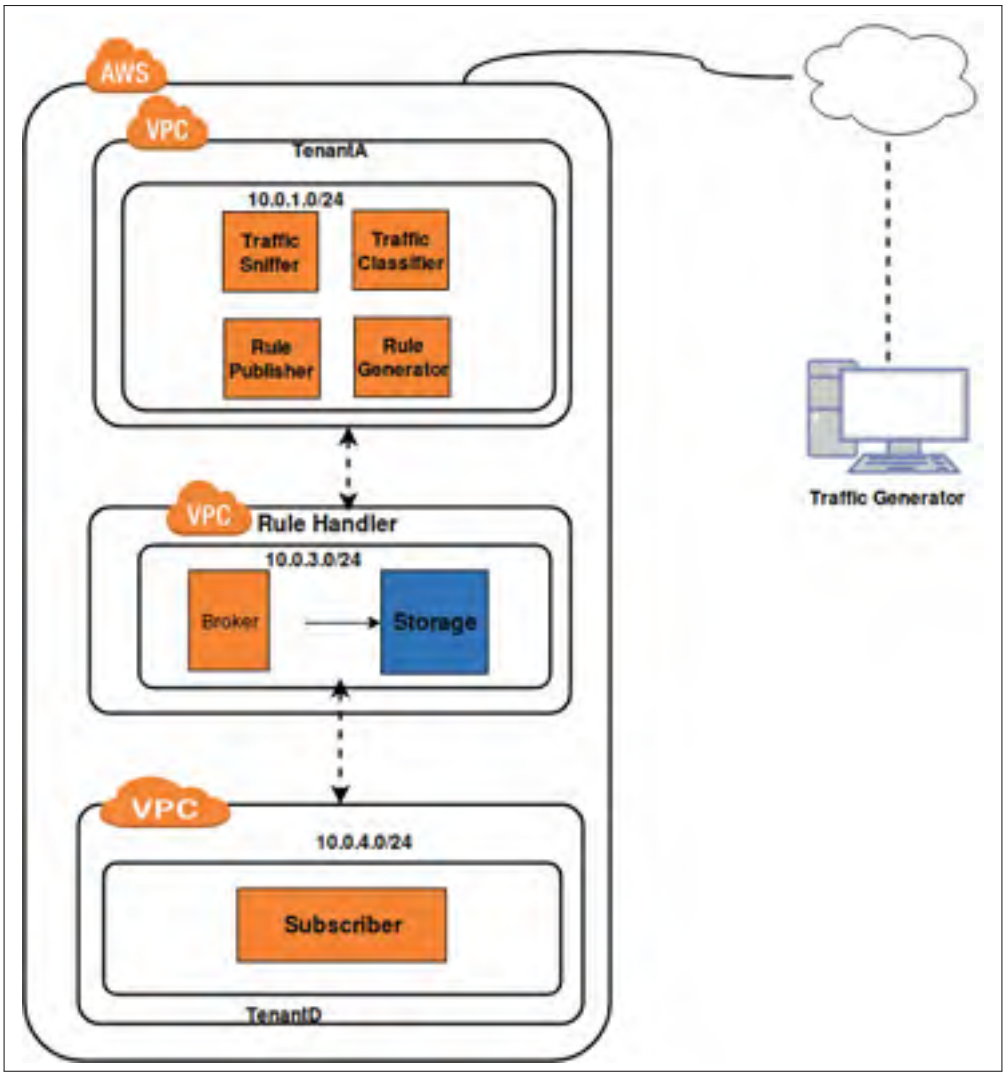


Figure 3.5 Network Architecture Set-up Scenario-1

or a web server by using Open-stack. Moreover, some Python scripts were written to simulate normal user behaviour at the clients. The CIDDS contains 14 attributes as shown in Table 3.3. The labeled attribute number 14, namely *Class*, demonstrates the status of the given instances whether a normal connection instance or an attack. We have ignored two features including the attack ID and description since they are mainly used to explain the information about the attacks. The data sets were split into training and test data sets.

Table 3.3 (CIDDS Set Attributes) Ring *et al.* (2018)

No	Attribute Name	Attribute Description
1	Src IP	IP address of the source node
2	Src Port	Port of the source node
3	Dest IP	IP address of the destination node
4	Dest Port	Port of the destination node
5	Proto	Transport protocol (e.g. ICMP, TCP, or UDP)
6	Date first seen	Start time flow is first seen
7	Duration	Flow duration
8	Bytes	Transmitted bytes
9	Packets	Transmitted packets
10	Flags	TCP Flags
11	AttackDescription	Additional information about an attack
12	AttackType	Type of an attack (portScan, dos.)
13	AttackID	Unique attack ID : (All flows which belong to the same attack carry the same attack id)
14	Class	Category or label of the instance

3.5.2 Network Setup Scenarios

Our approach can be deployed on top of a private or a public cloud. However, in this work, Amazon Web Service Amazon (2015), which is a public cloud, is used to deploy the proposed framework. As shown in Figure 3.6 and Figure 3.5, our MAIDS framework provides some flexibility in terms of offering the ability to choose the classification model that meets the tenant's requirements as well as the ability to deploy in the private and public cloud. The proposed approach can be easily adapted to multi-tenant environments where the public cloud tenants can have or own an account or multiple accounts. In our experimentation, we implemented our approach on top of AWS cloud. We created several Virtual Private Clouds (VPCs) where each VPC represents a cloud tenant resource.

Two different scenarios were considered to evaluate our proposed approach; more specifically, to illustrate the efficiency of the proposed approach. Figure 3.6 and Figure 3.5 present these scenarios:

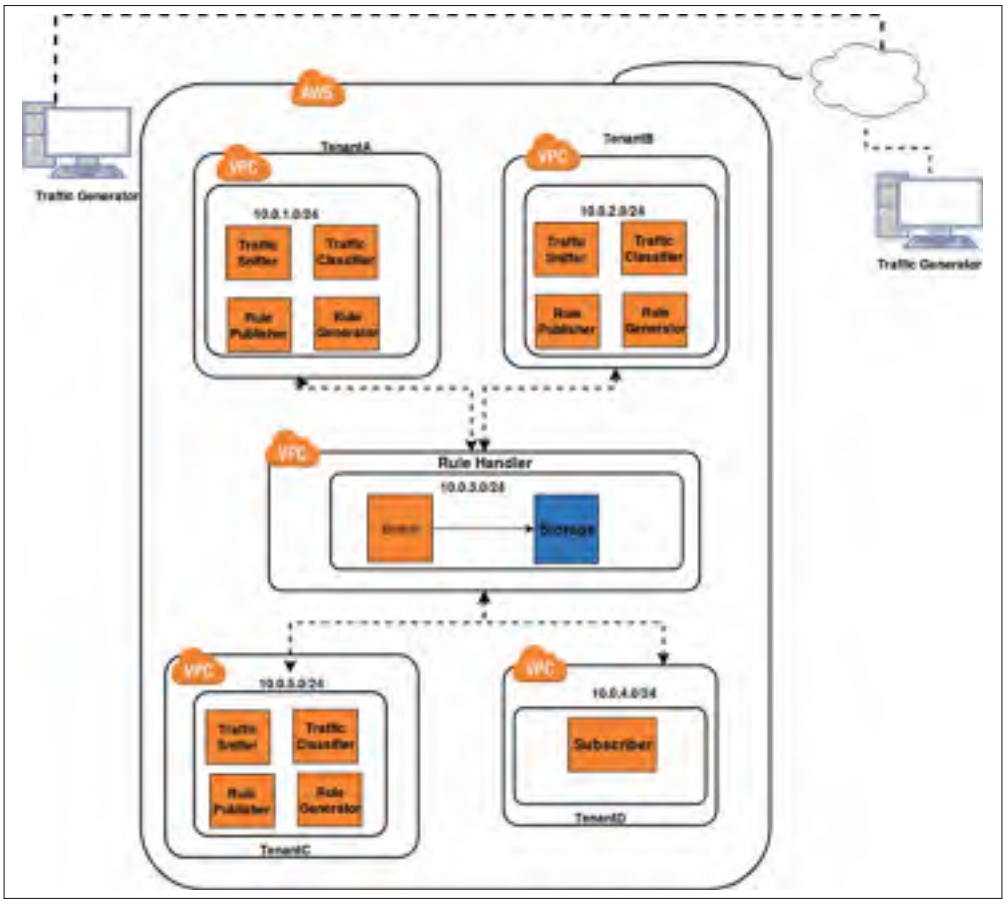


Figure 3.6 Network Architecture Setup Scenario-2

- **1 Tenant (Subscriber) and 1 Tenant (Publisher):** In the first scenario, we consider two tenants: *TenantA* is a publisher that deployed the component of *Traffic Handler* and *TenantD* is the subscriber that share the same interest as *TenantA* in terms of the type of the security service. Thus, *TenantD* will receive any new rules generated by *TenantA*.

  As shown in Figure 3.5, a tenant subscriber (*TenantD*) who has interest in protecting their cloud resource from DOS attack mainly will send a DOS subscription request to RH that is deployed in the subnet with range **10.0.4.0/24**. The subscriber component is implemented using Python. Moreover, a tenant publisher (*TenantA*) is a tenant that receives a traffic from different sources; deploys (*Traffic Sniffer* and *Traffic Handler*

that includes *Traffic Classifier*, *Rule Generator* and *Rule Publisher*) to classify the traffic and to generate new rules if abnormal packets are detected.

As previously mentioned, three different classification models including Random Forest(RF), Decision Tree (DT) and Neural Network (NN) were used to build *Traffic Classifier*. This gives some flexibility to the tenants in terms of choosing the classification model that meets their demands. The classification models were trained using CIDDS data set Ring *et al.* (2017b). Then, *Traffic Classifier* updates its signature IDS as well as other tenants' IDSs by dispatching the new rules — which were generated by the RG component based on the classification model — to *Traffic Handler* (TH). *Rule Handler* is deployed in the subnet **10.0.3.0/24**.

Traffic classification is a critical part or task performed by the MAIDS framework. Hence, to evaluate the trained classification model, both the normal and malicious traffic are generated using Iperf3 Iperf (2016) and Kali Linux Kali (2018) tools respectively. The malicious traffic includes port-scan (nmap) and dos (smurf).

The traffic is generated three times in three periods: 2, 5, and 10 minutes. In each period, Bro IDS is used to sniff the traffic and log it into a storage to be classified by *Traffic Classifier*. In the second and the third times, the percentage of the malicious traffic is increased to ensure that more malicious traffic are generated; thus, triggering the generation of more new rules.

The Bro IDS or the TS deployed in *TenantA* subnet is used to sniff and log the traffic into a buffer (storage). Then, the captured or sniffed traffic is classified by the classifier component (TH) to determine whether the traffic is normal or malicious. We implemented the *Traffic Handler* components, including *Traffic Classifier*, *Rule Generator* and *Rule Publisher* using Python. Thereafter, we run the scripts of the subscriber and the publisher on their respective instance based on the scenario described earlier. The malicious traffic is stored in a file and then sent to *Rule Generator* to generate new rules. These rules are dispatched by *Rule Publisher*. The rules are received by *Rule Handler* (RH) and then sent to *Broker*, which we implemented using

Python. The RH component delivers the rules to their destination (*TenantD*) based on its subscription policy mentioned in 3.; in this case, DoS. All the tenants used the same instance specification. Table 3.2 illustrates the specifications of the AWS instances used in our experiments.

- **Second Scenario: A Tenant (Subscriber) and 3 Tenants (Publisher):** In multi-tenant environments, multiple tenants attempt to dispatch data to *Broker* who is responsible for sending these data to the subscriber tenants. Hence, there is a need to investigate the capability of the framework in terms of the latency when the number of publishers is increased. As it is illustrated in the Figure 3.6, we assumed that we have four tenants (*TenantA*, *TenantB*, *TenantC*, and *TenantD*). These tenants deployed different Virtual Private Cloud **(VPC)**. On top of each VPC, public subnets were created to enable the deployment of MAIDS instances. (*TenantA* creates a subnet with range **(10.0.1.0/24)**, *TenantB* **(10.0.2.0/24)**, *TenantC* **(10.0.5.0/24)**. In these subnets, *Traffic Sniffer* and *Traffic handler* were deployed on top of each tenant's subnet. While *Rule handler* (RH) were deployed on the subnet with the range **(10.0.3.0/24)**. *TenantD* which is a subscriber creates a subnet with the range **(10.0.4.0/24)**.

  We conducted this experiment to demonstrate the efficiency of *Rule Handler* in terms of its capability to handle multi-tenant environment. First, *TenantA*, *TenantB*, and *TenantC* are assigned the same copy of the trained models built for classifying the incoming/out-going traffic. Second, all the tenants, that is, *TenantA*, *TenantB*, and *TenantC*, receive the traffic at the same time and for the same period mentioned in the previous experiment. Moreover, all of the tenants used Bro IDS Zeek (2019) as TS to sniff the traffic sent to TH. The captured or sniffed traffic is classified by the classifiers to determine whether it is normal or abnormal. The malicious traffic is stored in a file and then sent to RG to generate new rules. The new rules are dispatched by *Rule Publisher* and then are received by *Rule Handler* (RH), which then delivers them to *TenantD*.

3.5.3 Finding

As aforementioned, updating the time plays a vital role in keeping the tenant resources safe and stable. Thus, in contrast to other approaches where they concentrate mainly on calculating the performance of an anomaly IDS in terms of accuracy, our proposed approach takes into account the time needed to update the tenant IDS. Hence, we consider the classification time, the time for rule generation, the time of publishing the rules, as well as, the number of the tenants.

To measure the latency (L) as illustrated in Equation 4.3, let T_{class} denotes the classification time; T_{trans} denotes the time of transferring a classified file to RG; we denote the rule generation time with T_{gen}, and T_{publ} denotes the time taken to dispatch a rule till it is received by a tenant.

$$L = T_{class} + T_{trans} + T_{gen} + T_{publ} \tag{3.1}$$

A typical use-case scenario for the multi-tenant public cloud, where many tenants are publishing data to the middle-ware manger (broker) and then the broker delivers the data to its destination. For the purpose of our experiment, we measured the latency and the CPU consumption of the broker where there is a tenant, and where there are multiple tenants, specifically, where two, three, and five tenants are simultaneously publishing rules. We expect the latency and CPU consumed by the broker to be low for the case when there is a-tenant subscriber and a-tenant publisher, and high when there are four-tenant publishers and one tenant-subscriber. The following illustrates the scenarios used in our trials in details.

- Our first experiment was to compare the latency in terms of the time of the classification of the three models including the time needed to generate and deliver new rules to the subscriber tenant. In this experiment, we considered two tenants, *TenantA* acts as a publisher and *TenantC* as a subscriber. We illustrate this in Figure 3.5.

- In our second experiment, we increased the number of the publishers to identify how our approach behaves when the amount of data is increased.

- In the third experiment, we measured the CPU consumption of the broker in cases where one-, two-, three-, and four-tenant publishers are constantly publishing new rules.

- In the fourth experiment, our focus was on *Broker* performance (measuring the latency) by forcing *Broker* to send a thousand rules to different number of tenants. We measured the latency in the case where a publisher sends 1000 rules to different number of tenants: five, ten, fifteen, and twenty tenants. This experiment are different from the previous experiments as the time of rule classification, and the time of rule generation were not considered.

Figure 3.7 and Figure 3.8 present the result of our experiments. They show the latency of our proposed approach as described by Equation 4.3. In general, *Traffic Classifier* based on Decision Tree (DT) shows the lowest latency the while the *Traffic Classifier* based on Random Forest (RF) presents the highest latency. Hence, the classification time and the time needed to generate and deliver the new rules to their destinations are different.

Because the traffic was generated in the three periods: *PeriodA* lasts for 2 minutes; *PeriodB* lasts for five minutes; and finally, for *PeriodC*, the time doubled that of *PeriodB*; it lasts for ten minutes. Table 3.4 presents the generated traffic time specification including the size of the captured traffic needed to be classified. These files get classified using the Traffic classifier models and then the classified data is sent to RG for garnering new rules.

Table 3.4 Traffic Time

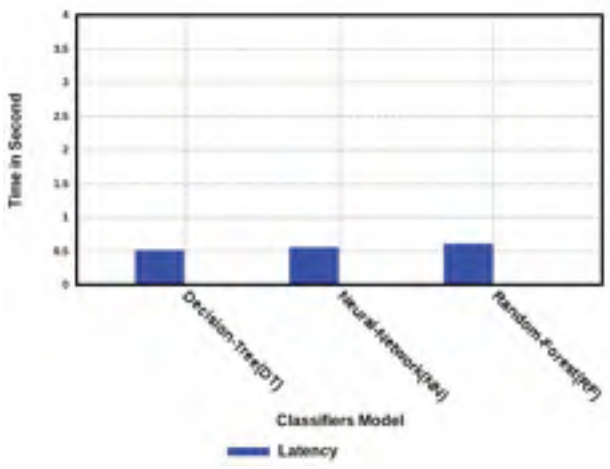
Period Name	**Time(Ms)**	**File Size(MB)**
PeriodA	2	50
PeriodB	7	150
PeriodC	10	250

Figures 3.7a, Figure 3.7b, and Figure 3.7c present the latency of our approach based on the first scenarios where one tenant (publisher) is considered. Figure 3.7a demonstrates the latency based on the *PeriodA* file where 10 rules were generated. It can bee seen that DT provides better latency by allowing the tenant to receive the new rules within 0.51690545s, while the latency of NN increases to 0.56090545s. Moreover, RF takes more time — about 0.610990545s. The latency increased when the traffic was increased. 3.7b shows the latency where the size of the file being classified is increased as the amount of the traffic is increased. It can be seen, however, that, as the latency is increased, the size of the file needed to be classified is increased. This makes the classification model takes more time for classifying the captured packets. Hence, the number of the new rules goes to 100. DT gives better latency by classifying, generating, and dispatching, new rules within 1.6157883s, while the latency of NN goes to 1.7485718s. Moreover, RF takes more time — it takes 1.8957883s. Figure 3.7c shows gives a higher latency for DT of around 3.26457098s while RF recorded around 3.73457098s. The NN registered 3.4857098s.

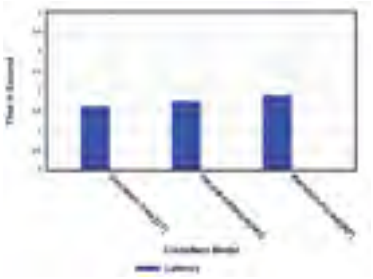
Figures 3.8a, 3.8b 3.8c compares the latency when there are three tenants simultaneously classifying traffic using different classifier models, generating new rules and publishing the new rules to the subscribers tenants.

As shown in the Figure 3.8a, the latency recorded was low when DT is used for the classification while RF recorded the highest. The latency recorded was 0.52440255s when DT was used while was recorded 0.56840255s with NN. Rf recorded a slight increase to roughly 0.61840255s We increased the size of the file to 150Mb. This file was classified using the three models and then 100 rules were generated after the classification as shown in Figure 3.8b. DT still registered the smallest latency of around 1.6187058s. However, the latency when RF is used was high at approximately 1.8987058s. Moreover. the latency was around 1.7507058s when NN was used. The size of classified file increased to 250Mb in order to generate more malicious packets and therefore generate more rules. Figure 3.8c compares the latency of three tenants simultaneously using different classification models and sending the new rules. It can be seen that the tenant latency while using DT

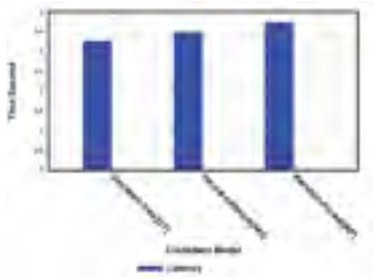
was low at around 3.46457098s. The tenants needed a few seconds of around 3.82980098s to complete the entire process while they need more time of 4.07980098s to finish the whole process when RF is used.



a) Latency of 10 Rules Latency(One publisher and one subscriber



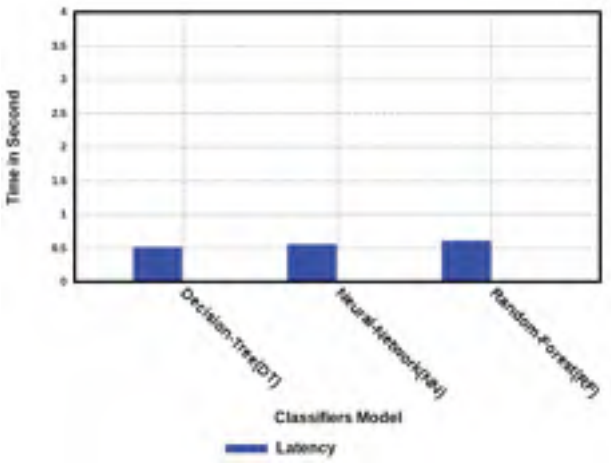
b) 100 Rules Latency(One publisher and one subscriber)



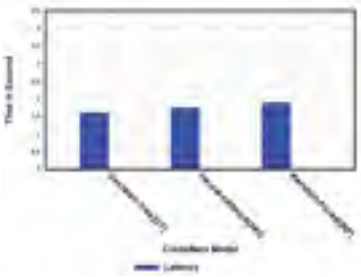
c) 150 Rules Latency (One publisher and one subscriber)

Figure 3.7 Comparison of latency 1 subscriber and 1 publishers

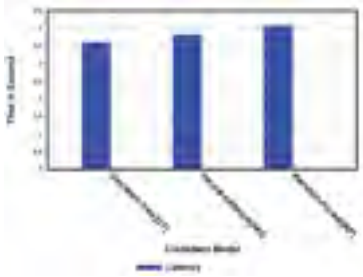
Figure 3.9 compares the CPU consumption of *Broker* when there are different tenants simultaneously dispatching new generated rules to the broker for the period of 140s. The CPU consumption of *Broker* is the lowest at around 26% in case of 1-tenant publisher and 1-tenant subscriber while the highest CPU consumption of roughly 40% was recorded



a) [Latency of 10 Rules Latency(Three publishers and one subscriber



b) 100 Rules Latency(Three publishers and one subscriber)



c) 150 Rules Latency (Three publisher tenants and a subscriber)

Figure 3.8 Comparison of latency 1 subscriber and 3 publishers

when the number of the publishers reached 4. The CPU consumption was 28% and 30% in the cases with two- and three-tenant publishers respectively.

Figure 3.10 shows the latency where a thousand rule were sent by a publisher to the different number of the tenants. Here, the measured latency are limited to measuring the time starting from the time of sending the rules by the publisher to the time when the rules are received by the tenants. As it can be seen, 0.32523s is considered a good performance as it takes 0.162615s, 0.32523s to deliver the rules to five and ten tenants respectively. Furthermore, Broker still functions very well even when the number of the

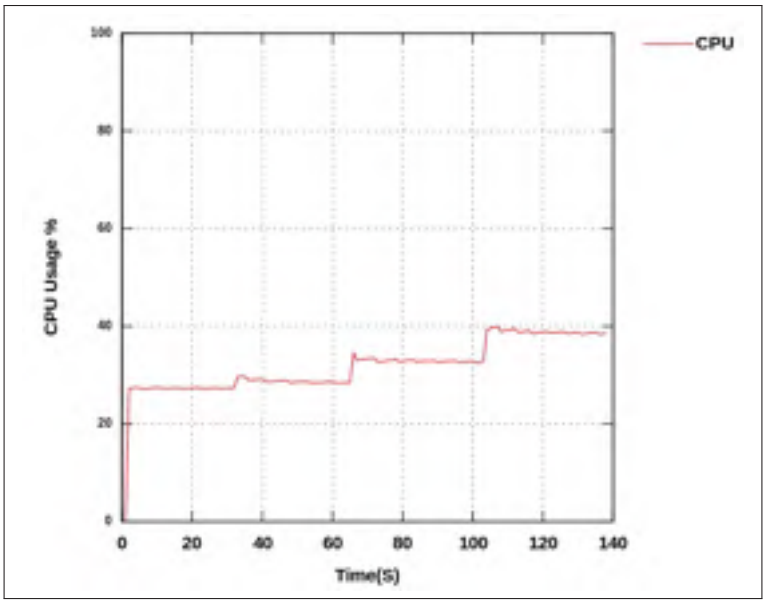


Figure 3.9 Broker CPU Consumption

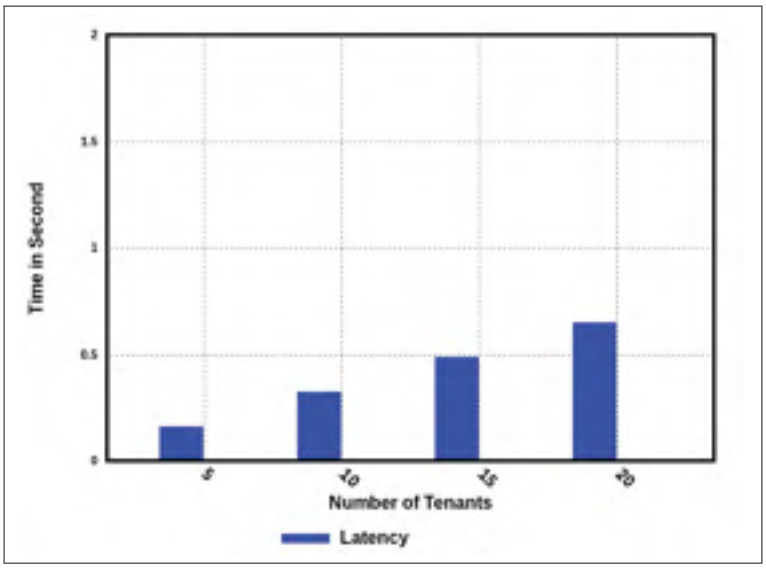


Figure 3.10 Broker Latency Performance

tenants are increased. We recorded 0.487845s in the case of fifteen tenants and 0.65046s in the case of twenty tenants. Our experimental results show that our proposed approach can provide a very high security mechanism as it enables tenants to detect some intrusions

that can not be detected by the signature detection. Moreover, it demonstrates that the approach has low latency and CPU consumption.

3.5.4 Discussion

As illustrated in the experiments, our framework is able to provide new custom rules that can be added to the standard rule baseline by enabling the existing IDS deployed at the tenant environment to be tailored to the requirements of a unique cloud tenant environment or to the unique business a deployment needs. Our proposed approach enables tenants to choose the classification model that meets their requirements as it offers three classification models: DT, NN, and RF. As is it shown in Subsection 3.5.3, finding different factors, including the size of captured file, the classification time, the time needed to send the classified file to RG, the rule generation time, and the process of dispatching the rule to its tenants based on some set of rules. Topic subscription plays vital roles in providing better latency. Our results show that the latency of the framework, which takes into consideration the time for the traffic classification and the time for the generation of new rules including the time for publishing till the rules are received by the subscribers (tenants), is extremely valuable in both scenarios. For the latency in the first scenario (a-tenant subscriber and a-tenant publisher), tenants were able to receive the new custom rules in a few seconds.

Another important aspect of our experiments is the performance of the broker in terms of its capability of handling the rules published by the tenant with low instance features including CPU and other important performance characteristics of the hardware. Overall, the CPU consumption was low for the case with one subscriber and one publisher, but increased with multiple publishers and a subscriber. The results show that small instance with 1-core CPU makes *Broker* able to handle large streams of rules.

3.6 Conclusion and Future Work

Security service updates play a vital role in protecting the tenants' resources from latest threats. Indeed, harmful attacks generally attempt to take advantage of the vulnerabilities in the security services. Thus, a delay in updating the security services keeps the door open for the attackers to penetrate the system. Our approach provides very crucial features: it provides fast, automated, and up-to-date system for updating the security services to keep them safe and stable. The proposed approach is tailored to meeting the requirements of the tenants implementing the generation of custom rules that can be added to the existing IDS standard rules. In the future, we will consider different network set-ups such as distributing the framework components into the cloud regions or accounts. In addition, we will consider the effect of increasing the number of the subscribers and publishers to identify the maximum number of the tenants that the broker can handle gracefully in terms of publishing the rules.

CHAPTER 4

ARTICLE 3: COLLABORATIVE REAL-TIME INTRUSION DETECTION SYSTEM (CRIDS)

Mohamed Hawedi[1], Abdi Ramy [1] , Chamseddine Talhi [1] , Hanifa Boucheneb [2]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
[2] Department of Computer Engineering and Software Engineering, Polytechnique Montréal
2900 Edouard Montpetit Blvd, Montreal, QC H3T 1J4

Article Submitted « The Journal of Supercomputing » August, 3, 2019.

A cloud environment is composed of a set of computing resources that provide services on-demand and as pay-as-you-go to cloud tenants. The services are offered over the Internet, which facilitates easy access by the users as well as by the intruders to the data stored in the cloud. Detecting intrusion attempts, however, has become very critical; it has also become more difficult. Hence, the traditional anomaly intrusion-detection systems for detecting cloud intrusions have been proposed. But they are largely ineffective as they fail to detect anomalies behavior and new attacks because the attack characteristics and patterns continue to change. Therefore, the traditional anomaly IDS that is trained with a data set can offer high accuracy rate at a point in time. It may lose its accuracy at a later time because of changes in the characteristics and the patterns of attacks. Thus, we propose a new Collaborative Real-time Anomaly Intrusion Detection System (CRIDS) that trains the classification model in real-time to provide improved anomaly detection in the cloud.

4.1 Introduction

Cloud computing has been developing rapidly recently due to its distinctive characteristics, which include broadband-network access, provision of on-demand services, rapid elasticity, resource pooling and measured service. Indeed, cloud uses the internet to provide its

services. This enables the cloud users to perform their daily jobs or tasks regardless where they are (Hawedi *et al.*, 2018a). Cloud computing provides various services that are elastically offered to users over the internet. This enables users to comfortably perform their daily jobs regardless of their locations. Consequently, more and more users deploy and develop high computation-intensive applications and lightweight applications in the cloud. The security of the cloud resources encompasses the CIA triad(confidentiality, availability, and integrity) of the data in the cloud. Hence, ensuring these aspects of the cloud security has become very challenging. An attacker can exploit the cloud computing vulnerabilities to gain unauthorized access and perform malicious activities that affect the CIA triad of the cloud resources and the services. Because of this, the security of cloud computing has gradually become the main obstacle to further development. Therefore, the security of critical information systems in the cloud should be considered very important and thus be provided to the tenants (Xing *et al.*, 2018). Tenant resources must be protected from attacks, unauthorized disclosure of information and the modification or destruction of data.

Indeed, different requirements play vital role in terms of selecting the best security that meets the tenants' needs. Among different requirements, detecting known and abnormal activities and recognizing normal traffic are an important aspect. Cloud IaaS, for instance, are exposed to attacks including DoS, port scanning, password guessing, or buffer overflow by both internal and external intruders. This motivates the need to provide a new security mechanism that is able to provide better security and integrity of the information systems.

Among different security mechanisms, intrusion detection system (IDS) is a very important mechanism; it applies several rules to distinguish between legitimate and illegitimate events (Elhag *et al.*, 2019). It plays an important role in network protection by assisting the security service provider in detecting malicious behaviors. IDSs are classified into two main sets: network-based IDSs target detecting attacks in the network traffic, while host-based IDSs target detecting attacks at the host itself.

In terms of techniques, IDSs are classified into various groups: *anomaly detection* is used to detect unknown attacks and *signature detection* (misuse) is able to detect known attacks. Indeed, a signature-based IDS detects the attacks by looking for some specific patterns. Because this technique (signature detection) relies on simple-rule systems and string comparison, it is very prone to spoofing: the flow classifier can be mislead by intruders that inject appropriate flags or tags into its own traffic (Tsilimantos *et al.*, 2018).

The traditional anomaly intrusion-detection system was used by different entities including researchers, academicians, and practitioners as a solution to overcoming spoofing. Nevertheless, the traditional anomaly intrusion-detection systems cannot detect new unknown attacks as the patterns of attacks are changing rapidly. Thus, there is a need to develop a unique and novel anomaly IDS that is able to keep track of changing attack patterns.

Collaboration is a way of allowing communication between entities — tenants who have the same applications, cloud providers, cloud security providers, cloud providers and tenants, or cloud security providers and tenants. Hence, collaboration is considered as a very important and imperative aspect in building trust and improving the security as it enables exchange of knowledge regarding security between entities. This reduces the impact of security threats and enhances the capability of meeting the security requirements.

This paper focuses on providing an anomaly detection as a security service designed to enable the monitoring and protection of the tenants' virtual environments. The main contribution of this work is to provide a Collaborative Real-time Anomaly Intrusion Detection Systems (CRIDS) — that takes the tenant's security requirements into consideration. The proposed approach provides a portable, and declarative security framework that is able to share more system resources. This helps to achieve cost reduction. The framework also minimizes the time and effort related to the framework management. In addition, to recognize unseen attacks, the framework generates new models and updates the tenants with the new generated security mechanism (classification model) by working in a collaborative manner. Among the various machine learning techniques, Decision Tree

(DT), Support Vector Machine (SVM), and Random Forest (RF) have been used to build our anomaly IDSs as they have better ability to recognize normal network traffic and to detect unknown intrusion attacks.

In short, we will pursue the following objectives:

- Overcoming the lack and the limitation of the traditional anomaly IDSs by providing a new anomaly-based IDS that is capable of adapting to the changes that occur in the network-intrusion characteristics and patterns. We do this by developing and introducing an up-to-date anomaly IDS that continuously trains the IDS model or classifier to detect new attacks.
- Providing a light-weight, portable, reproducible and declarative framework that is able to share more system resources. We take into consideration, cost reduction and minimizing the time and effort related to framework management by offering automated management features.
- Minimizing the additional resources and maximizing resource sharing between the framework components by achieving further resource efficiency; the sharing is a very important aspect and it is imperative to build trust as well as to improve the security as it enables the exchange of knowledge regarding security. This results in reducing the impact of the security threats and enhances the capability of the provider to fulfill the security requirements.
- Enable cloud entities such as tenants to get advance and collaborative security mechanism framework. With the collaboration features, entities are enabled to exchange knowledge regarding attack; hence, build trust.

The remainder of the paper is organized as follows. We introduce our approach in Sect. 4.1. We discuss the related work in Sect. 4.2. In Sect. 4.3, we describe the cloud tenants' requirements. The architecture of CRIDS is described in detail in Sect. 4.4. In Sec 4.5,

we describe the evaluation of our system and discuss the results of our experiments. We conclude the paper and present the future work in Sects. 4.6 and 4.7.

4.2 Related Work

There are numerous work proposed on anomaly-based IDS in cloud that we have been studied; we discuss them in detail as follows:

4.2.1 Anomaly Based Detection Approaches

Various anomaly detection approaches for detecting new attacks were suggested in the literature.

The authors, (Velmurugan & Thirukumaran, 2012) proposed an anomaly IDS to overcome the limitations of the traditional IDS. In their work, they proposed two main approaches: performance and information approaches. The performance approach is based on an artificial neural network algorithm. It aims at detecting any attempts to exploit the vulnerabilities of the system and also, to contribute to the automatic discovery of new attacks. The performance approach receives a profile that includes user behaviors that are captured by the event auditor node. The performance approach classifies the received data; any deviation observed is detected as an anomaly. The information approach has a database that consists of pre-configured rules or information regarding specific vulnerabilities and attacks. Thus, the information approach enables a high discovery rate of known attacks.

An anomaly detection system was proposed by (Pandeeswari & Kumar, 2016). The work targets the hypervisor (virtual machine monitor) layer and aptly named *hypervisor detector*. Its main idea is to detect abnormal behaviors on virtual network by analyzing the network events on VMs. In addition, this work is designed to improve the accuracy of intrusion detection systems by using a hybrid algorithm which is a mix of artificial neural network (ANN) and fuzzy C-Means clustering (FCM) algorithms. The proposed approach

follows three stages: In the first stage, the large data-set is split into training subsets using a fuzzy clustering technique. For the experiment, DARPA's KDD cup data-set 1999 was used. In the second stage, different ANNs are trained using the trained sets obtained after splitting in the first stage. Finally, fuzzy aggregation module is used for re-learning and combining the generated ANN modules into a single ANN module to eliminate the errors of different ANNs. In the third phase, fuzzy aggregation module is introduced to aggregate the ANN module's results into a unique module through which it can eliminate the detected errors.

Similar to this work is an anomaly detection system framework proposed by (Ganeshkumar & Pandeeswari, 2016). It is designed to detect abnormal activities in the cloud with high detection accuracy and minimum false negative rate. The proposed anomaly hypervisor detector model is built based on adaptive neuro-fuzzy inference system (ANFIS), which is an integration of fuzzy systems with adaptation and learning proficiency of neural network. It was developed to monitor the activities of the virtual machines. In this work, anomalous activities belonging to a host and network can be detected without deploying the model in the virtual machines.

4.2.2 A hybrid IDS Approaches (Anomaly-Based & Signatures-Based Detection)

A hybrid approach that combine signature-based and anomaly-based IDSs have been proposed to overcome the detection issues.

Modi, Chirage and Patel, Dhiren (Modi & Patel, 2018) presented a new IDS framework whose aim is to detect abnormal events in cloud virtual networks.

The Hybrid-NIDS aims at examining and detecting intrusion in the network traffic and in the cloud environment. To enable the monitoring of traffic inbound from external networks to the VMs or the internal VMs traffic, Hybrid-NIDS sensors are deployed on each cloud host machine and region. Such deployment facilitates the capability of monitoring multiple

VMs that are deployed on the same host simultaneously, and guarantees the protection of the host machine and VMs from penetration.

(Tupakula *et al.*, 2011) targeted security cloud IaaS by integrating the signature-based intrusion-detection approach with the anomaly-based intrusion-detection technique. They deployed their hybrid intrusion-detection systems on top of the virtual machine monitor (VMM) to be able to monitor all the incoming and out-going packets, and to identify the malicious entity that attempts to compromise the virtual machines.

(Modi & Patel, 2013) merge the traditional signature-based
intrusion-detection system with the anomaly-based intrusion-detection system to improve the detection accuracy and consequently enhance the capability of detecting the network attacks. Their approach uses different machine-learning approaches including Bayesian, associative, and decision tree to build the framework. In their work, the incoming or out-going traffic passing through the signature IDS is forwarded to an anomaly IDS for classification.

(Aljawarneh *et al.*, 2018) proposed a hybrid classification-based intrusion detection model that uses a feature selection to help in detecting attacks over the network. In this work, the feature selection method is first applied to *NSL-KDD* data set. Later, the *n-intrusion* detection model based on machine-learning approach is built and used to find attacks. At the end, the captured data is used to improve intrusion detection.

4.2.3 Collaborative IDS Approaches

Some work in the literature present intrusion-detection system that are deployed and work in a cooperative manner in which IDS signatures are exchanged between cloud tenants or providers for detecting suspicious activities.

(Alruwaili & Gulliver, 2014) proposed a collaborative IDS framework that aims to protect the cloud infrastructure layer against known and unknown threats. Further, they target

the protection of providers and its tenants from the loss of services that are mainly caused by known and unknown attacks. In their work, both the signature-based and anomaly-based techniques are used for enabling known and unknown threats detection.

A distributed architecture aims at providing intrusion detection in cloud computing was presented by Ficco et al.
(Ficco *et al.*, 2013). Their approach targets providing protection for the cloud providers. They developed a framework, which monitors the cloud resources and identifies the resources of the cloud providers that have been compromised by other providers. This framework is designed to detect scalable distributed attacks.

A collaborative IDS framework that targets cloud layers including the infrastructure, the platform, and the application levels was introduced by
(Gul & Hussain, 2011). This work targets coordinated attacks. It is done by allowing cloud providers to implement distributed IDSs to detect any intrusion attempts and enable cloud tenants to monitor their applications.

Taghavi Zargar et al. (Zargar *et al.*, 2011) introduce a distributed, collaborative, and data-driven intrusion detection and prevention framework (DCDIDP) for cloud computing environments that targets coping with an attack using intrusion detection and prevention systems (IDPSs). The proposed framework allows cloud service providers to collaborate in a distributive way by which it can reply to attacks and offer universal IDPSs. The collaborative IDS deployed by the cloud providers shares global databases that they use to detect sophisticated cooperative intrusion.

However, all the discussed approaches have not taken into consideration the aspect of meeting the requirements of the tenants and providing the tenants who share same interest with rapid advance attack detection. To conclude this section, the approaches discussed earlier are not entirely satisfactory with regard to the provision of a security mechanism that is capable of satisfying the tenant's requirements. The following are some of the unique contributions of our approach that differentiate it from the work discussed earlier:

- The proposed real-time anomaly IDS is designed to meet the requirements of the tenants. The security mechanism — a real-time anomaly IDS — is first trained based on the tenant's security requirements. For instance, a tenant who is interested in DOS protection will get a DOS anomaly-IDS model while a tenant that is interested in, say, SQL injection protection, will get a trained a SQL anomaly-IDS model. A tenant can have an anomaly IDS that has been trained for all attacks too.

- We extend the traditional anomaly IDS with continuous update mechanism in which it plays a vital role in increasing the rate of the attack detection (detecting new unknown attacks) that can cause damage to the tenant's resources. In this regard, unlike the previous work, generating custom real-time anomaly IDSs is a very fundamental feature of our approach.

- Our proposed approach provides an advance protection for the tenants with privacy guarantee as our approach encourages the tenants to work in collaboration with regard to new generated classification models.

- Since the latency of the framework is also a vital aspect of this approach, the time starting from building the new anomaly IDS classification model till the time new model/models are dispatched to the tenant is also considered by our approach.

4.3 Cloud Tenant Requirements

Tenant security requirements tend to be different as they have different IaaS, topology, and deployed applications. Hence, there is the need to provide a security mechanism that is able to meet their security requirements Hawedi *et al.* (2018b). In our previous work, we assume that we have different tenants who need to deploy different security mechanisms to protect their cloud resources. As shown in Figure 4.1 , *TenantA* requires DOS and SQL protection while *TenantB* wants to protect his/her cloud resources from DOS and Telnet. *TenantC* shares the same security interest as *TenantA*, since DOS and SQL injection are their main concerns.

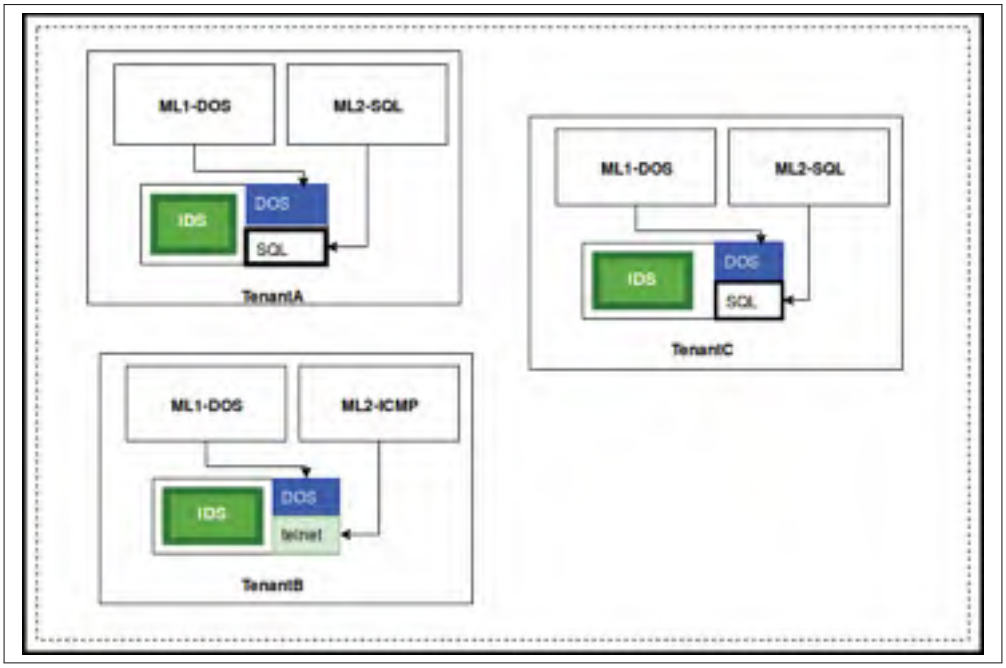


Figure 4.1 Tenant Security Requirement

Based on this assumption, there is a need to develop a security mechanism that takes into account the tenants' preferences. The implemented security should meet the tenants' deployment architectural perspectives. With this context, our proposed approach allows cloud tenants to obtain significant gains regarding cost reduction, security, and update times.

In our previous work, we provided an anomaly IDS that is designed to monitor the activities of the system and then classify them as either normal or abnormal (anomalous). Moreover, the proposed approach offers a trained anomaly IDS to tenants based on their demands. In this context, as illustrated in Figure 4.1, *TenantA* and *TenantC* will get a trained DOS, and SQL anomaly IDSs. While *TenantC* will receive a trained DOS and Telnet anomaly IDSs. Our approach offers certain flexibility that allows cloud tenants to choose one or more anomaly IDSs that meet their needs.

Based on the classification of attacks, we enable the tenant to automatically create custom rules tailored to their environment, which can be added to a set of generic rules or

signatures. These rules are sent quickly to tenants based on their security preferences or subscriptions as the proposed approach enables communication between tenants based on their security interests. For instance, in Figure 4.1, *TenantA* and *TenantC* shared the same security interest: they are both looking for DoS and SQL protection. In this case, each time a tenant activates or deactivates a security service (DoS), a subscription message is sent to indicate that these tenants are activating or deactivating DoS set of rules. Based on this, the tenants will be able to receive the new rules generated based on the classification.

However, some concerns have risen with the regard to the ability of the Anomaly IDS to classify and detect new threats.

Therefore, in this work, we propose a real-time anomaly IDS, which continuously updates the classification model based on the newly captured threats. Our approach offers the tenant the possibility to automatically generate new classification models and shares these models based on the environment. Theses models are sent to other tenants based on their security requirements, interests or subscriptions. For instance, in Figure 4.1, *TenantA* and *TenantC* shared the same security interest: they are both looking for DoS and SQL protection. In this case, each time a tenant activates or deactivates a security service (DoS), a subscription message will be sent to *Model Manager* indicating that these tenants are activating or deactivating DoS security services. Consequently, the tenants will be able to receive the new classification model.

4.4 Proposed Approach

The proposed Collaborative Real-time Intrusion Detection System (CRIDS) framework is designed to enable a cloud tenant to get a security mechanism that is able to provide high accuracy rates and low false positive rates. The proposed framework is a light-weight, portable and scalable framework that is able to share system resources. This framework not only provides known/unknown attack detection, but also updates the anomaly IDS

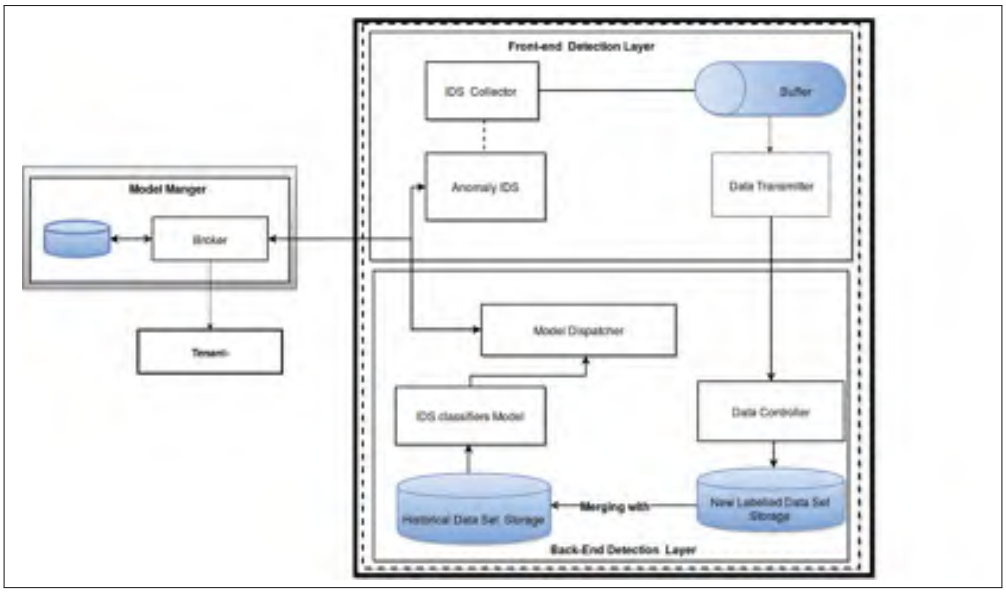


Figure 4.2 The Architecture of CRIDS

by continuously training the IDS model or the classifier to be able to detect new threats. Our proposed approach encompasses of two main layers namely: the front-end detection layer and back-end detection layer. The layers are shown in Fig 4.2.

1. ***Front-End Detection Layer*:**

   This layer is composed of *IDS Collector* (IC), *Buffer*, *Data Transmitter*, and *Anomaly IDS* (AI).

   a. IDS Collector (IC): IC aims at collecting summaries of the incidents and keeping track of data of attacks. It collects the traffic passing through the network and then stores them in a buffer.

   b. Buffer is a temporary storage used to store packets captured by the IC component. The captured packet flow has some attributes including start time and end time of the flow, number of packets, the IP addresses and port number of the source, the IP addresses and port number of the destination, transport protocol, size, TCP flags, and relative timestamp. In our approach, we adjust the buffer size to 4000 packets to enable rapid training data-set generation. Each time the buffer

reaches 4000 packets, a new data session is generated and then transferred by *Data transmitter* to *Data Controller*.

c. Data Transmitter(DT): dispatches the temporary data (network activities) stored by Buffer to Data Controller for clustering and labeling.

d. Anomaly IDS (AI): the main task of AI is to detect anomalies. The AI is a real-time anomaly IDS that is continuously trained so that it can detect new anomalies.

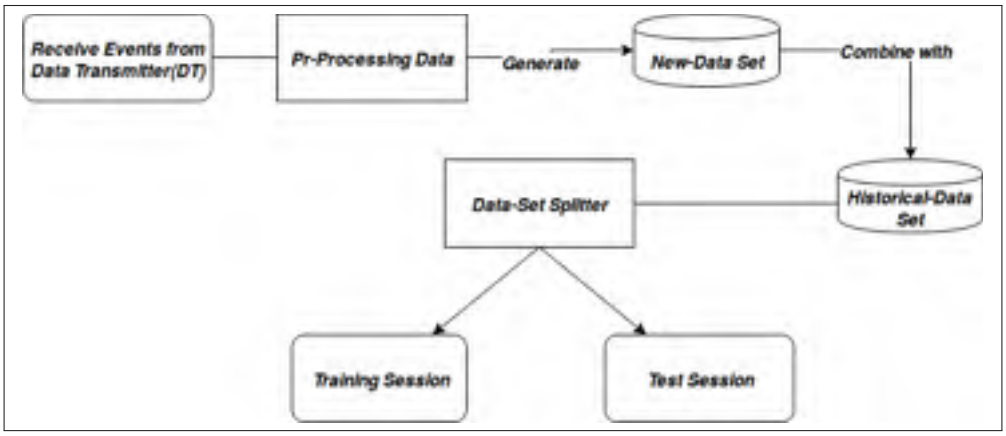


Figure 4.3 DC Process in collaboration with Data Transmitter

2. ***Back-end Detection Layer*:**

This layer is composed from Data Controller, New Data Set Storage, Historical Data Set Storage, and IDS Classifiers, and Model Dispatcher.

a. Data Controller (DC): The DC is the core component of the back-end detection Layer and is responsible for generating train and test data-set sessions. It performs its functions by interacting with the DT component.

Figure 4.3 illustrates the process that DC follows to generate a new data-set session. **(1)** DC receives the new events (traffic) captured by the IC component. **(2)** Later, IC pre-processes the received events; it analyzes and classifies the captured traffic to determine whether or not it is normal. Based on this, it labels

each connection on the data-set (as either normal or attack) and generate a new data-set.

(3) The new data-set is combined with the historical data-set to build a comprehensive training and testing data-sets; hence, building an accurate model.

b. IDS Classifiers Model (ICM): its main task is to classify the network traffic into either normal or abnormal. Various supervised machine-learning algorithms can be used to build ICM. Hence, the model would be trained and validated based on the labeled data-set generated by the DC component. ICM is designed to build and generate up-to-date classifier model that overcomes the limitations of the existing anomaly IDS in terms of detecting new threats.

c. Model Dispatcher (MD): The security requirements of the tenants tend to be different and distributed in multi-tenant environments. Thus, sharing information regarding the security are very vital. Our proposed approach provides a fundamental advantage in terms of instant notifications of events for these different and distributed security systems. It is considered a powerful model for disseminating the events from the publishers (i.e., the data-event model producer, which could be tenants or security-service providers) to the subscribers (i.e., data-event tenants who share the same interest as the publisher). The MD component is responsible for dispatching the generated model (ICM) to *Model Manger*.

The Security Service Providers **(SSP)** perform an important task in meeting the need of the tenants who share the same interest. Thus, the *Model Manger* (MM) component plays an important role in achieving this goal. The MM component can be centralized or federated; its responsibility is to deliver the model sent by MD to the right tenant. It consists of two components named *Broker* and *Storage*.

1. Broker: *Broker* is used to route the new generated model from the MD component to the tenant subscriber. As shown in Figure 4.4, *TenantA* and *TenantC* have the same

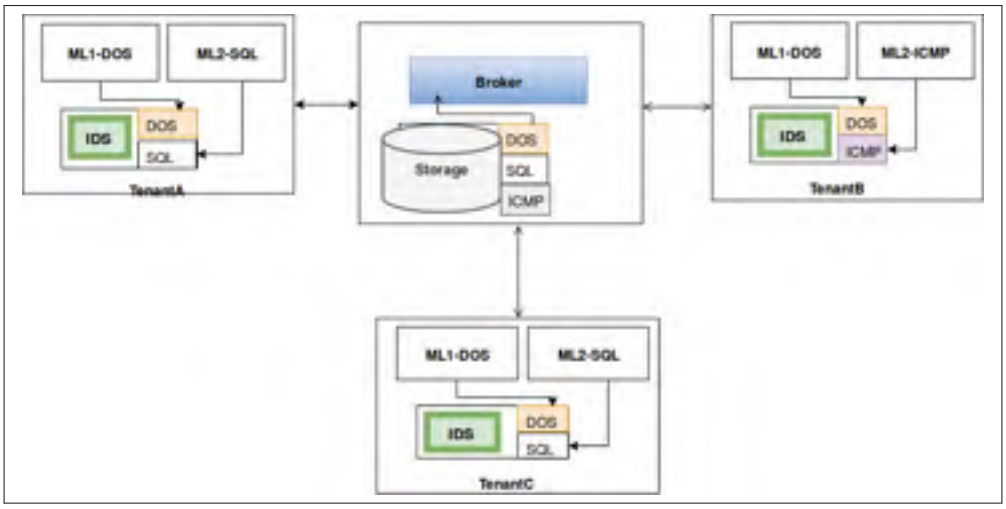


Figure 4.4 Classifiers Model Broker Mechanism

security requirements: they are both seeking the DOS and SQL anomaly detection while *TenantB* deploys *DOS* and *ICMP* anomaly IDS.

A subscription notification includes security requirement *(DOS, SQL)* and a new generated model (ML-DOS, ML-SQL) will be sent to *Broker* by *TenantA* and *TenantC*. On the other hand, *TenantB* is seeking *(DOS, ICMP)* protection; hence, he/she deploys *(DOS, ICMP)* anomaly IDS. As a result, a notification message (with *a DOS-and-ICMP subscription*) will be sent to the broker. Accordingly, a policy table that includes the name of the tenants (*SubscriberNa*) and the security preferences (*Topic(Security interests*) will be created to facilitate the transfer of the classification models to the right tenant.

Table 4.1 Broker Policy to achieve sharing interests between tenants

SubscriberNa	**Security Classification Model Preferences (Topic)**	
TenantA	DOS	SQL
TenatB	DOS	ICMP
TenantC	DOS	SQL

In this assumption, first, *TenantA* has activated DOS set of rule publisher, and it subscribes to DOS. Furthermore, *TenantA* will publish any new generated classification model as well as receive any DOS model published by other tenants. *TenantB* is interested in protecting its environment from SQL; while *TenantD*'s main task is to publish models related to SQL. *TenantC* is interested in receiving all the classification models generated by ICM. It is clear that *Broker* can be centralized or federated: it has all of the chains of events including the topic (classification model names) and the subscriber (tenant name) that are used within the event flow.

2. **Storage** : allows the broker to store all the information regarding a subscriber (tenant), publisher (a tenant or a security provider), including their classification model types.

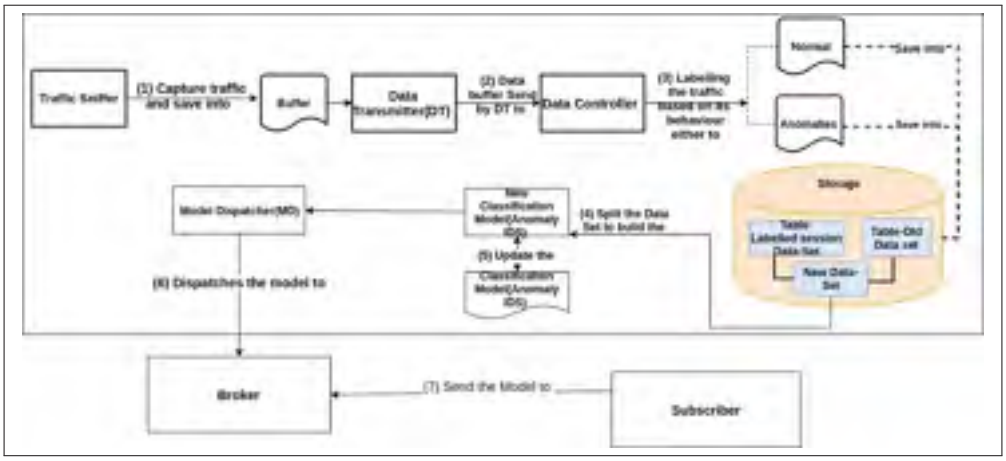


Figure 4.5 The mechanism of the CRIDS

4.4.1 CRIDS Mechanism

In the previous sections, we mentioned that our approach is designed to enable tenants to prevent unknown attacks by generating custom rules and by quickly updating the tenants — who share the same interest — with the new generated rules. Here, we provide

a detailed illustration of CRIDS's process by describing how the components of CRIDS interact to provide a powerful security approach.

As shown in Figure 4.5, the incoming traffic is captured by the traffic sniffer and stored in a buffer (see message (1)). (2) The data buffer is sent to *Data Controller (DC)* by *Data Transmitter (TD)*. (3) DC uses determines if the network data traffic is normal or abnormal. Consequently, a new data set is generated and then combined with the historical data set. With this scheme, the new data now has more knowledge about old and new threats; thus contributing to the generation of a more accurate model. (4) The new data-set is split to a training and testing data-sets to build the classification model. (5) Update or replace the old classification model (anomaly IDS) with the new model. (6) The *Dispatcher* sends the new model to *Broker*, who then sends the model to the subscriber (tenant).

4.5 System Evaluation

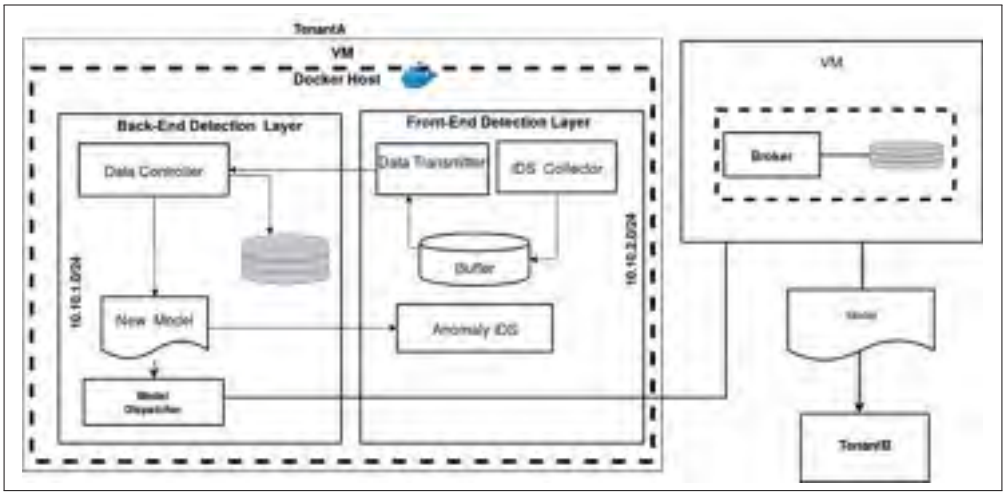


Figure 4.6 Network Architecture Set up

The main goal of our proposed work is to propose a real-time anomaly IDS that is continuously trained with recent malicious data to improve the detection accuracy. In this section, we explain our approach to evaluating the proposed work.

4.5.1 Environment setup

This work uses the micro-service technology. It allow us to develop a light-weight, portable framework that is able to share system resources. It also enables us to the minimize the framework management time and effort through automated management features.

We use Docker Container Docker (2019) — a lightweight virtualization technology — to implement our framework. The Container allows us to partition the physical machine resources and to provide isolated instances for resource management.

Figure 4.6 shows the deployment of the CRIDS component on top of Docker running on Ubuntu 16.04.4 LTS Linux operating system. We created two subnets for the deployment of the framework. The components of the front-end detection layer were deployed on a subnet range (10.10.2.0/24) while the (10.10.1.0/24) subnet range was used for the back-end detection layer. Bro IDS images were used to create the *IDS collector (IC)* container to collect or capture incoming and out-going traffic passing through the network; MySQL Docker image was used to build the *Buffer* that was used to store the data captured by IC as discussed earlier. We created a container for the *Data Transmitter (DT)*, which is responsible for transferring captured data to DC.

Although our approach can be implemented with different MLs, in this work, we used the following well-known machine-learning algorithm (ML) classifiers to build the AI model.

1. **Decision Tree (DT)**: In DT, data is represented in the form of a hierarchical tree that consists of a group of nodes that describes a problem with various solutions. In this approach, a complex problem is divided into simpler problems; then same strategy is applied recursively to the sub-problems. The solutions of sub-problems are

combined in the form of a tree to yield the solution of the complex problem Gama *et al.* (2003).

2. **Support Vector Machines (SVM)**: finds the best hyper-plane separating the data points of different classes Tsilimantos *et al.* (2018).

3. **Random Forest (RF):** RF is used to overcome the problems related to classification and regression. It builds multiple decision trees to get better predictive results. It also creates a forest with a number of decision trees to get the best solution.

Table 4.2 Data sets Traffic Type

Iteration	**Type of traffic**	**Number of Packet**	**Normal**	**Attacks**
It1-DT1	Normal Traffic, smurf Attack	4000	70 %	30 %
It2-DT2	Normal Traffic, Nmap Attack, Smurf Attack	8000	47 %	53 %
It3-DT3	Normal Traffic, Nmap attacks, Smurf Attacks , Pod Attacks	12000	32 %	68%

4.5.2 Data-Set Creation and Labeling

To create our data sets, we use a buffer methodology that enables our proposed framework to maintain newly added data for continuous training of the network stream. Each time the buffer reaches 4000 packets, we save the content of the buffer into the database and clear the buffer. With this context, we simulated different traffic containing normal and malicious packets.

Consequently, we obtained different data-sets that enable us to evaluate the performance and the feasibility of our real-time anomaly IDS.

The normal and malicious traffic were simulated using different tools — *Iperf* Iperf (2019) and *Kali Linux* Kali (2019) respectively. To improve the quality of CRIDS, different experimental scenarios based on the traffic types were conducted to study the performance of CRIDS. We obtained different data-sets from these scenarios. Table 4.2 illustrates the types of the data-sets created and the type of the traffic that corresponds to each data-set as well as the number of the packets and their proportional percentage.

The first scenario, *Data-Set (Ds1)*: **Ds1** contains normal traffic and malicious traffic (distributed denial of service) that were generated to overwhelm the system. Basically, distributed denial of service (DDoS) flooding can cause devastating attack on the victim's computer. Therefore, we simulated Smurf (DDos) attack. The 'Smurf' attack is a kind of denial-of-service attack where the attacker sends large number of Internet Control Message Protocol (ICMP) packets to the victim computer Kumar *et al.* (2006).

The second scenario, *Data-Set (Ds2)*: This set is a combination of the data collected in *Ds1* that includes 'normal' traffic as well as 'Smurf' with a new simulated 'nmap' attack. For *Ds2*, we first simulated 'nmap' attack: a port scanning attack that is used to detect vulnerabilities. When the buffer reaches buffer size, the collected data was added to those collected for *Ds1*.

The third scenario, *Data-Set (Ds3)*: This set is composed of 'normal' traffic as well as 'Smurf' and 'nmap' attacks that were gathered in *Ds2* and new simulated 'pod' attacks. For *Ds3*, we first simulated a ping-of-death ('pod') attack, which is a kind of denial of service (DoS) attack where the attacker sends large IP packet to the victims. We later added the collected data to *Ds2*.

4.5.2.1 Labeling Data Set Traffic

Labeling is the second step after collecting the data sets. In the literature, a variety of algorithms have been proposed for creating and labeling data-set traffic.

Zheng *et al.* (2011) proposed an anomaly intrusion detection called IDCPSO (Clustering and PSO). The IDCPSO model aims at: (1) modeling the normal behavior of a user by creating clusters from unlabeled training data-sets; (2) labeling a cluster as either 'normal' or 'abnormal'.

The application on intrusion detection using K-means cluster algorithm was introduced Jianliang *et al.* (2009). In this work, K-means algorithm was used to cluster and analyze the data by looking for patterns in a set of unlabeled data. Lin *et al.* (2015) proposes an intrusion detection system based on combining cluster centers and nearest neighbors (CANN). The CANN uses K-means clustering algorithm to extract cluster centers of each pre-defined category; then the nearest neighbor of each data sample in the same cluster is identified.

Pamukov *et al.* (2018) proposed Negative Selection Algorithm (NSA) for creating training data-set by relying on the normal behavior of the network traffic only. NSA is capable of performing unsupervised learning; It identifies any deviations as anomalies. According to Igawa & Ohashi (2009), NSA algorithm involves a number of detectors whose main task is to detect the abnormal data. Generally, NSA has two stages: generation sage and detection stage. In the generation stage, the candidate detector is randomly produced and then tested to know if it is able to recognize the self-samples. The detector is removed if it recognizes any sample or is added to the detector set in case it is unable to recognize self-samples at all. In the detection stage, the performance of the detector set is validated to see if it is able to recognize the input data. The recognized data are considered as abnormal while the unrecognized data are considered as normal Igawa & Ohashi (2009).

The foregoing approaches can be used to label our data-sets.However, in this evaluation, we are not experimenting labeling algorithms since we are generating normal as well abnormal traffic. The generated packets are dumped using Bro IDS and then additional pre processing steps are performed. Later, each data set is split into a training set (68%)

and a testing set (32%). The classifying process, which uses supervised machine learning, is divided into two phase: training and validation.

4.5.3 Performance Metrics

The goal of this work is to provide a new collaborative real-time anomaly intrusion-detection system that constantly trains its classification model to detect anomalies and to overcome issues related to the change in the attack patterns. Furthermore, the proposed approach enable tenants who share the same interest to work in collaboration by exchanging their generated anomaly IDS models. This subsection describes the metrics used to evaluate the performance of the framework. The evaluation is presented in two steps:

- *Step1*: Evaluating the performance of the classifiers in terms of accuracy, etc., for each experimental scenario discussed earlier. Our aim is to evaluate the effectiveness of the classifier in predicting the class label of the instances.

  The following formulas define these metrics Weng & Poon (2008):

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4.1}$$

$$Accuracy = TN + TP/(TN + TP + FN + FP) \tag{4.2}$$

  Step2: Measuring the latency — the time needed to update the tenant with the new generated anomaly IDS model. This is because updating the tenants security system plays a vital role in keeping the tenant resources safe and stable. Based on this, we derived Equation 4.3 to measure the latency.

  Let $T_{Building}$ denotes the time taken to build the anomaly IDS, including training and validation times; while T_{Publ} denotes the time taken to dispatch a rule until it has been received by a tenant.

$$Latency = T_{Building} + T_{Dispatch} \tag{4.3}$$

4.5.4 Experimental Results

4.5.4.1 Experiments1 : Evaluate the performance of CRIDS and then compare the performance with that of an anomaly IDS

To achieve this, data-sets were gathered as discussed in the Subsection 4.5.2.

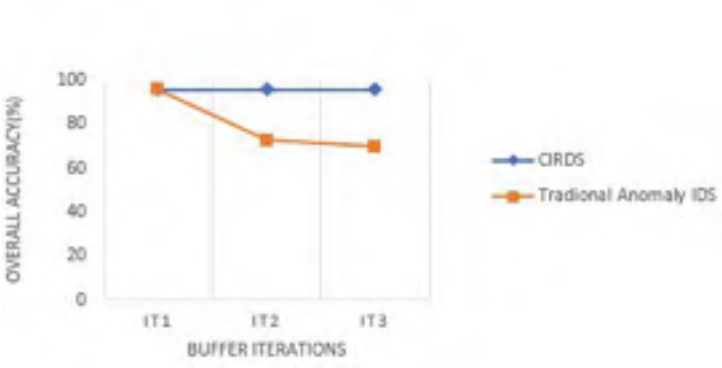
In our first experiment, our aim is to understand the need for updating the training set. This is to enable us to address the concern of the importance of updating the repository of malicious and normal data.

Here, we are more specifically aimed at evaluating the performance of the CRIDS. To archive this goal, we created different data sets using the approach discussed in Subsection 4.5.2. These were used to build different anomaly-based IDSs.

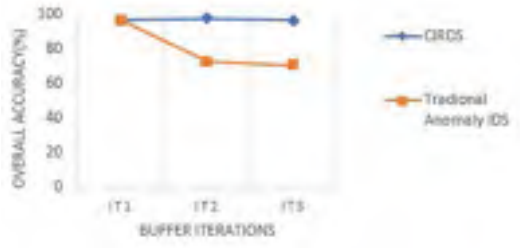
Figures 4.7a,4.7b, and 4.7c show the overall accuracy of CRIDS compared to the traditional anomaly IDS. As shown in the figure, CRIDS-based SVM, DT, and RF show notable superiority in terms of the attack detection rate compared to the traditional anomaly IDS. Furthermore, CRIDS-DT gives a better result compared to the others CRIDS types.

As shown in Fig 4.7a. CRIDS-SVM performs well over time as it keeps learning and the accuracy stays at roughly 96% over buffer iterations. While the traditional anomaly IDS-based SVM proves unreliable; it also shows a lack of the ability to discover the new threats as the accuracy goes down to around 73% for the second iteration and 70% for the third iteration session.

Additionally, CRIDS-based decision tree (CRIDS-DT), as shown in Fig 4.7b, posts clear superiority in terms of accuracy standing at over 97% in Iteration 1 (It1), Iteration 2 (It2), and Iteration 3 (It3) compared to the traditional anomaly IDS-based Decision Tree,



a) Comparison of the accuracy of traditional IDS vs CRIDS-based support vector machine (SVM)



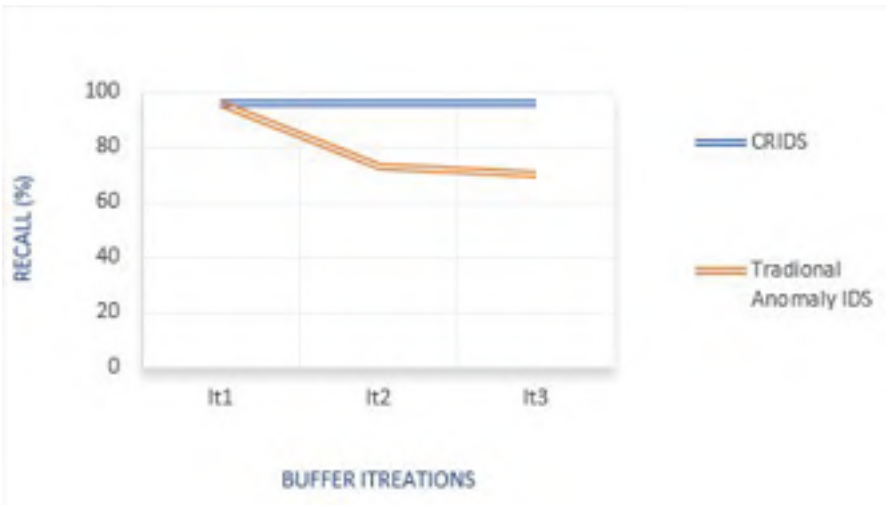
b) Comparison of the accuracy of traditional IDS vs CRIDS-based decision tree (DT)

c) Comparison of the accuracy of traditional IDS vs CRIDS-based random forest (RF)

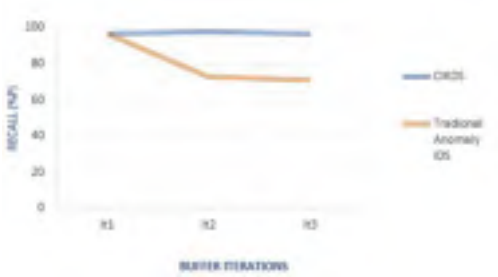
Figure 4.7 Comparison of *Accuracy* between CCRIDS based (SVM, DT, and RF) and traditional anomaly IDS

where the performance deteriorates to about 71% in Iteration 3. Furthermore, traditional Random Forest posts low accuracy over time at around 72% for IT2, IT3 as shown in Fig 4.7c. The CRIDS-based Random Forest (CRIDS-RF) proves to be more reliable in detecting attacks; its posts accuracy of over 98% for all the iterations; the accuracy of the traditional anomaly IDS-based RF recorded were 74% and 74% for the It2, and IT3 respectively

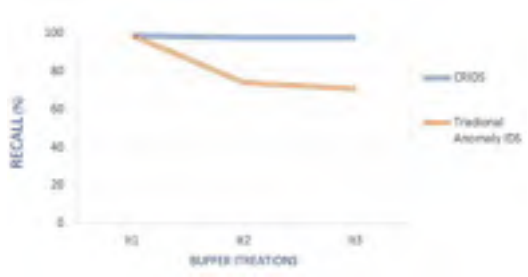
It is very important for a ML algorithm to return relevant results most of the times. Thus, in this approach Recall is measured. As shown in Figures 4.8a, Fig 4.8b, and Fig 4.8c ,



a) Comparison of the Recall of traditional IDS vs CRIDS-based support vector machine (SVM)



b) Comparison of the Recall of traditional IDS vs CRIDS-based decision tree (DT)

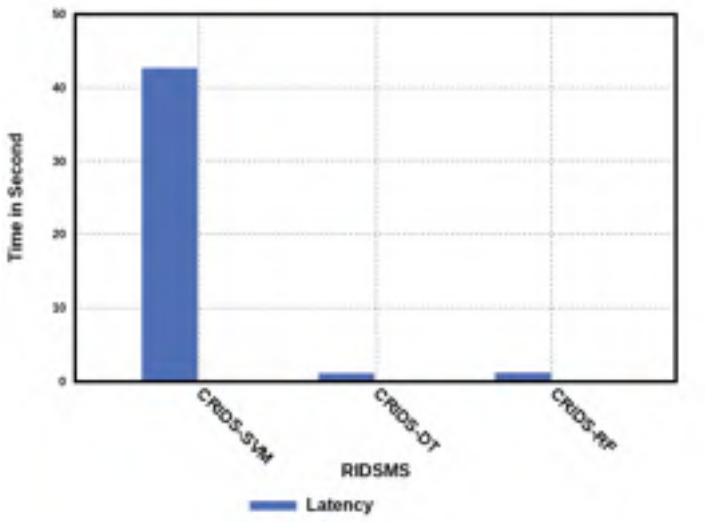


c) Comparison of the Recall of traditional IDS vs CRIDS-based random forest (RF)

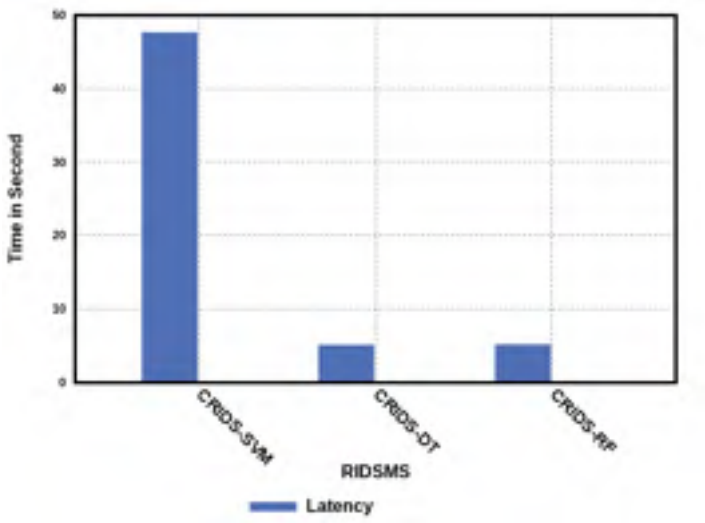
Figure 4.8 Comparison of *Recall* between CCRIDS based (SVM, DT, and RF) and traditional anomaly IDS

the CRIDS-based SVM, DT, and RF show clear superiority compared to the traditional anomaly IDS. Furthermore, CRIDS-DT performs better than other CRIDS.

It is shown in Fig 4.8a, Fig 4.8b, and Fig 4.8c, the classifiers for the CRIDS and the traditional anomaly IDS show a similar trend and perform better when the data-set in Iteration 1 is used. The performance of the traditional IDS dropped significantly at It2, and IT3. In contrast, however, the CRIDS performance is stable at around 97% for CRIDS-SVM, 98% for CRIDS-DT, while it records 98% for CRIDS-RF. While the performance of the traditional anomaly IDS based SVM was 73% for the first iteration, and 70% for the



a) CRIDS Latency comparison(1 publishers and 1 subscriber)



b) CRIDS Latency comparison(1 publishers and five subscriber)

Figure 4.9 Comparison of Latency of CRIDS Models

second iteration (It2). Also, the performance of the traditional anomaly IDS-based DT reduces in It2, and It3 to around 74% and 71%. Furthermore, the traditional anomaly IDS-based RF registered a decrease in It2, and It3 to around 73% and 71%. This can be explained by the fact that the CRIDS models are continuously trained so that they are able to detect new unknown threats.

4.5.4.2 Experiments2: Measuring the Latency

In our second experiment, we addressed the latency of CRIDS in updating the tenants who share same interest in terms of the security mechanism.

Figure 4.9a, and 4.9b presents the latency of our proposed approach as described by Equation 4.3. Generally, the *CRIDS* based on DT posts the lowest latency the while the *CRIDS* based on SVM presents the lowest latency. Notably, CRIDS demonstrated some reliability in terms of updating the tenants with new generated models.

Figure 4.9a presents the latency of our approach based on the first scenario where one tenant (publisher) and one (subscriber) are considered. It can be seen that CRIDS-DT provides better latency by enabling tenants to receive the model within 1.11s while the latency of CRIDS-RF records a slight time increase to approximately 1.18s in delivering the mode . The CRIDS-SVM takes more time in delivering the model where the latency recorded is around 42.7s. Number of tenants can impact the latency.

Figure 4.9b shows the latency of our approach based on the scenarios where one tenant (publisher) and five tenants (subscribers) are considered. It can seen that CRIDS-DT provides better latency by enabling tenants to receive the model within roughly 5s, while the latency of CRIDS-RF is increased to about 5.18s. The CRIDS-SVM takes more time – about 47s.

4.6 Discussion

In this work, we introduced a novel machine real-time anomaly IDS framework to classify network traffic in real time. The core of our approach is providing an anomaly IDS that classifies network traffic, and provides advance attack detection by exchanging the anomaly IDSs' models. The proposed framework offers better privacy as it mainly shares the new generated ML models with other tenants. Furthermore, it enables the tenants to select the best CRIDS model that fits their individual needs. Based on the results

obtained, we can conclude that our CRIDS is superior to the traditional IDS; our CRIDS enables the traditional IDSs to keep learning and updating their models as it incrementally adds new buffers (captured traffic) to the existing model. Hence, it is able to detect new threats. Furthermore, our purposed approach shows better latency in updating the tenants who shares the same interest with a new generated model. This provides an advance attack detection as the received model can be combined with the existing model. Whereas comparing different scenarios, the CRIDS loaded with DT is better at detecting new attacks. Moreover, it has better latency regrading dispatching the model to different tenants.

4.7 Conclusion and Future work

Cloud computing vulnerabilities can exploited by an attacker to perform unauthorized activities that affect the cloud resources' integrity, availability, and confidentiality. Thus, maintaining the confidentiality, integrity, and availability of the cloud computing resources is a fundamental requirement that needs to be considered. Traditional anomaly IDSs have been used to detect malicious attacks. However, the continued evolution of attacks hinders the capability of traditional anomaly intrusion-detection system to detect new attacks. Thus, we introduced a new real-time intrusion-detection system based on micro-services monitors and classifies network traffic in real time to detect anomalies. The proposed approach offers some flexibility in that tenants can choose the anomaly IDS that fits their individual needs. The proposed approach ensures privacy and advance attack detection as it enables the cloud tenants who share the same interest to exchange the new generated anomaly IDS model. For future work, we will complete the full architecture by integrating the generated models with the existing model and then evaluate the performance of new integrated model.

CONCLUSION AND RECOMMENDATIONS

Despite the remarkable advantages of cloud computing, security remains a major challenge due to the distributed nature of the cloud computing environments as well as its cloud resources are provided on the internet, which unfortunately, makes them an attractive target for abuse; and hence, an area of vulnerabilities. With this regard, cloud tenants need security mechanisms that are capable of meeting and fulfilling their requirements and needs.

Thus, addressing cloud security problems have been the essential aims of this thesis. More specifically, these security barriers and their implications have been studied and addressed using various models.

At different stages in this research, many questions have arisen and numerous sub-goals had to be identified in order to accomplish the main goal. Hence, we began by investigating the tenants' requirements in terms of security that can be impacted by different factors such as the type of the deployed application, topology, etc. After studying the main existing solutions and their associated security approaches, we were able to identify the limitations of the security mechanisms offered to the tenants by the cloud providers. Based on this investigation, we started to build our solution.

In the first article (Chapter 2), we have proposed a novel IDS framework — Multi-tenant Intrusion Detection System (MTIDS) as a service that targets the security of the public cloud —. In particular, the MTIDS delivers appropriate and optimized security taking into consideration the tenants' needs in terms of their security service requirements and budget.

In the second article (Chapter 3), we proposed a new anomaly intrusion detection paradigm — the Multi-tenant Anomaly Intrusion Detection System (MAIDS) — that is capable of

detecting intrusive attacks and that can automatically generate new signatures from the attacked system. The proposed approach is able to automatically generate customized set of rules. This can significantly improve the efficiency of the IDS and can ensure the extraction of valuable information from the monitored network.

Finally, in the third article (Chapter 4), we highlighted the need for real-time intrusion detection System in detecting abnormal behavior and new attacks that its characteristics and patterns continue to change. Further, we highlighted the need for providing an advance intrusion detection that enables tenants who share the same security interests to collaborate by exchanging new generated anomaly IDSs. Thus, we propose a new Collaborative Real-time Anomaly Intrusion Detection System (CRIDS) that trains the classification model in real-time to provide improved anomaly detection in the cloud and to enable cloud tenants to collaborate in terms of exchanging newly generated classification models.

In a nutshell, the proposed solutions were conducted after a deep investigation regarding the existing security offered to the tenants. The results have demonstrated the efficiency of each of the proposed approaches.

Future Research Directions

In this thesis, our proposed solution targets providing a security mechanism that is capable of meeting the cloud tenants' security requirements. The proposed security enables cloud tenants to have an advance-attack detection by enabling them to exchange relevant information regarding the security defense. This security information includes new generated rules and classification models that are gathered from each tenant to be shared with other tenants on-demand by which they provide advance attack detection.

Nevertheless, we still believe there are many considerable and interesting suggestions that can be tracked in the future. We highlight some of them below:

- Testing the proposed framework based on diverse collaborative forms: between tenants belonging to different regions; between various service/IaaS providers. This will enable gathering of large security information.

- Optimizing the broker's capabilities: it can be through evaluating the relevance of the new rules and classification models before publishing and/or broadcasting them. This will enable reducing the overload of the broker. Hence, increasing its availability. Furthermore, overcoming the redundancy issues resulted from sending the same information to the tenants.

- Performing scalability: testing by deploying the solution on a large-scale test bed where the number of tenants applications is very large. This will allow us to test the broker, which is responsible for performing different tasks with regards to managing the events shared between the tenants. This investigation will enable to measure the capabilities of the Manger and Broker in terms of the number of tenants' requests that they can manage.

BIBLIOGRAPHY

Adil, M. & Ijaz, I. (2015). IDS in Cloud Computing to Secure Virtual Environment. *International Journal of Enhanced Research in Science Technology & Engineering*, 4(3), 199–207.

Alharkan, T. & Martin, P. (2012). Idsaas: Intrusion detection system as a service in public clouds. *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pp. 686–687.

Aljawarneh, S., Aldwairi, M. & Yassein, M. B. (2018). Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, 25, 152–160.

Almorsy, M., Grundy, J. & Müller, I. (2016). An analysis of the cloud computing security problem. *arXiv preprint arXiv:1609.01107*.

Almorsy, M., Grundy, J. & Ibrahim, A. S. (2011). Collaboration-based cloud computing security management framework. *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 364–371.

Alruwaili, F. F. & Gulliver, A. (2014). CCIPS: A cooperative intrusion detection and prevention framework for cloud services. *International Journal of Latest Trends in Computing*, 4(4).

Amazon. (2015, January, 30). Amazon Web Services (AWS). Consulted at https://aws.amazon.com.

Amazon. (2016a, January, 1). Amazon EC2 [Format]. Consulted at https://aws.amazon.com/ec2/?nc2=h_m1.

Amazon. (2016b, May, 1). Amazon Virtual Private Cloud (VPC) [Format]. Consulted at https://aws.amazon.com/vpc.

Ambusaidi, M. A., He, X., Nanda, P. & Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE transactions on computers*, 65(10), 2986–2998.

Arba. (2018, January, 30). Aruba Cloud. Consulted at https://www.arubacloud.com.

Ashktorab, V., Taghizadeh, S. R. et al. (2012). Security threats and countermeasures in cloud computing. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 1(2), 234–245.

Azure. (2018). Microsoft Azure: Cloud Computing Platform & Services. Consulted at https://azure.microsoft.com.

Bouzida, Y. & Cuppens, F. (2006). Neural networks vs. decision trees for intrusion detection. *IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM)*, pp. 81–88.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.

Caruana, R. & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning*, pp. 161–168.

Chapade, S., Pandey, K. & Bhade, D. (2013). Securing cloud servers against flooding based DDoS attacks. *Communication Systems and Network Technologies (CSNT), 2013 International Conference on*, pp. 524–528.

Chou, D. C. (2015). Cloud computing: A value creation model. *Computer Standards & Interfaces*, 38, 72–77.

Debar, H., Curry, D. A. & Feinstein, B. S. (2007). The intrusion detection message exchange format (IDMEF).

Demchenko, Y., Turkmen, F., Slawik, M. & de Laat, C. (2017). Defining Intercloud Security Framework and Architecture Components for Multi-Cloud Data Intensive Applications. *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*, pp. 945–952.

Docker. (2019). Enterprise Container Platform for High-Velocity Innovation. Consulted at https://www.docker.com/.

Elhag, S., Fernández, A., Altalhi, A., Alshomrani, S. & Herrera, F. (2019). A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. *Soft Computing*, 23(4), 1321–1336.

Ferry, N., Brataas, G., Rossini, A., Chauvel, F. & Solberg, A. (2014). Towards Bridging the Gap Between Scalability and Elasticity. *CLOSER*, pp. 746–751.

Ficco, M., Venticinque, S. & Di Martino, B. (2012). Mosaic-based intrusion detection framework for cloud computing. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 628-644.

Ficco, M., Tasquier, L. & Aversa, R. (2013). Intrusion detection in cloud computing. *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*, pp. 276–283.

FSabahi & AMovaghar. (2008). Intrusion detection: A survey. *Proc. - The 3rd Int. Conf. Systems and Networks Communications, ICSNC 2008-Includes I-CENTRIC 2008: Int. Conf. Advances in Human-Oriented and Personalized Mechanisms, Technologies, and Services*, (January 2008), 23–26.

Gama, J., Rocha, R. & Medas, P. (2003). Accurate decision trees for mining high-speed data streams. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 523–528.

Ganeshkumar, P. & Pandeeswari, N. (2016). Adaptive neuro-fuzzy-based anomaly detection system in cloud. *International Journal of Fuzzy Systems*, 18(3), 367–378.

Gul, I. & Hussain, M. (2011). Distributed cloud intrusion detection model. *International Journal of Advanced Science and Technology*, 34(38), 135.

Gupta, S. & Kumar, P. (2017). Profile and back off based distributed NIDS in cloud. *Wireless Personal Communications*, 94(4), 2879–2900.

Hawedi, M., Talhi, C. & Boucheneb, H. (2018a). Multi-tenant intrusion detection system for public cloud (MTIDS). *The Journal of Supercomputing*, 74(10), 5199–5230.

Hawedi, M., Talhi, C. & Boucheneb, H. (2018b). Security as a Service for Public Cloud Tenants (SaaS). *Procedia computer science*, 130, 1025–1030.

iana. (2016). Service Name and Transport Protocol Port Number Registry. Consulted at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

Ibrahim, A. S., Hamlyn-Harris, J., Grundy, J. & Almorsy, M. (2011). Cloudsec: a security monitoring appliance for virtual machines in the iaas cloud model. *2011 5th International Conference on Network and System Security*, pp. 113–120.

Igawa, K. & Ohashi, H. (2009). A negative selection algorithm for classification and reduction of the noise effect. *Applied Soft Computing*, 9(1), 431–438.

Iperf. (2016, October, 2). iPerf-The ultimate speed test tool for TCP, UDP and SCTP [Format]. Consulted at https://iperf.fr/iperf-download.php/.

Iperf. (2019). iPerf-The ultimate speed test tool for TCP, UDP and SCTP. Consulted at https://iperf.fr/iperf-download.php/.

Jamil, D. & Zaki, H. (2011). Security issues in cloud computing and countermeasures. *International Journal of Engineering Science and Technology (IJEST)*, 3(4), 2672–2676.

Jianliang, M., Haikun, S. & Ling, B. (2009). The application on intrusion detection based on k-means cluster algorithm. *2009 International Forum on Information Technology and Applications*, 1, 150–152.

Kali. (2018). Our Most Advanced Penetration Testing Distribution, Ever. Consulted at https://www.kali.org/.

Kali. (2019). Our Most Advanced Penetration Testing Distribution, Ever. Consulted at https://www.kali.org/.

Katsaros, G., Menzel, M., Lenk, A., Rake-Revelant, J., Skipp, R. & Eberhardt, J. (2014). Cloud Service Orchestration with TOSCA, Chef and Openstack. *Proceedings of the IEEE International Conference on Cloud Engineering (IC2E)*, pp. 34.

Keegan, N., Ji, S.-Y., Chaudhary, A., Concolato, C., Yu, B. & Jeong, D. H. (2016). A survey of cloud-based network intrusion detection analysis. *Human-centric Computing and Information Sciences*, 6(1), 19.

Kevric, J., Jukic, S. & Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, 28(1), 1051–1058.

Kulkarni, G., Gambhir, J., Patil, T. & Dongare, A. (2012). A security aspects in cloud computing. *2012 IEEE International Conference on Computer Science and Automation Engineering*, pp. 547–550.

Kumar, S., Azad, M., Gomez, O. & Valdez, R. (2006). Can microsoft's Service Pack2 (SP2) security software prevent SMURF attacks? *Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (AICT-ICIW'06)*, pp. 89–89.

Lin, W.-C., Ke, S.-W. & Tsai, C.-F. (2015). CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. *Knowledge-based systems*, 78, 13–21.

Lo, C.-C., Huang, C.-C. & Ku, J. (2010). A cooperative intrusion detection system framework for cloud computing networks. *Parallel processing workshops (ICPPW), 2010 39th international conference on*, pp. 280–284.

Magar, A. (2012). Data Protection in Multi-Tenant Cloud Environments.

Man, N. D. & Huh, E.-N. (2012). A collaborative intrusion detection system framework for cloud computing. *Proceedings of the International Conference on IT Convergence and Security 2011*, pp. 91–109.

Mathew, S. (2014). Overview of Amazon Web Services. (November).

Mell, P. & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, 145.

Microsoft. (2015, July). Microsoft adopts first international cloud privacy standard. Consulted at https://blogs.microsoft.com/on-the-issues/2015/02/16/microsoft-adopts-first-international-cloud-privacy-standard/.

microsoft. (2016, January, 30). Microsoft SQL Server [Format]. Consulted at https: //www.microsoft.com.

microsoft. (2017). Official Home Page. Consulted at https://www.microsoft.2.

Mishra, N., Sharma, T. K., Sharma, V. & Vimal, V. (2018). Secure Framework for Data Security in Cloud Computing. In *Soft Computing: Theories and Applications* (pp. 61–71). Springer.

Modi, C. & Patel, D. (2018). A feasible approach to intrusion detection in virtual network layer of Cloud computing. *Sādhanā*, 43(7), 114.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A. & Rajarajan, M. (2013). A survey of intrusion detection techniques in cloud. *Journal of network and computer applications*, 36(1), 42–57.

Modi, C. N. & Acha, K. (2017). Virtualization layer security challenges and intrusion detection/prevention systems in cloud computing: a comprehensive review. *the Journal of Supercomputing*, 73(3), 1192–1234.

Modi, C. N. & Patel, D. (2013). A novel hybrid-network intrusion detection system (H-NIDS) in cloud computing. *Computational Intelligence in Cyber Security (CICS), 2013 IEEE Symposium on*, pp. 23–30.

Nikolai, J. & Wang, Y. (2014). Hypervisor-based cloud intrusion detection system. *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pp. 989–993.

Oasis. (2017). OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) TC. Consulted at https://www.oasis-open.org/committees/tosca/faq.php.

Oktay, U. & Sahingoz, O. K. (2013). Proxy network intrusion detection system for cloud computing. *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on*, pp. 98–104.

Onwubiko, C. (2010). Security issues to cloud computing. In *Cloud Computing* (pp. 271–288). Springer.

Openstack. Open source software for creating private and public clouds. Consulted at https://www.openstack.org/.

Openstack. (2015, Mar, 1). Open source software for creating private and public clouds [Format]. Consulted at "ww.openstack.org".

Osanaiye, O., Choo, K.-K. R. & Dlodlo, M. (2016). Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67, 147–165.

Pamukov, M. E., Poulkov, V. K. & Shterev, V. A. (2018). Negative Selection and Neural Network Based Algorithm for Intrusion Detection in IoT. *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pp. 1–5.

Pandeeswari, N. & Kumar, G. (2016). Anomaly detection system in cloud environment using fuzzy clustering based ANN. *Mobile Networks and Applications*, 21(3), 494–505.

Park, H., Lee, E.-J., Park, D.-H., Eun, J.-S. & Kim, S.-H. (2016). PaaS offering for the big data analysis of each individual APC. *Information and Communication Technology Convergence (ICTC), 2016 International Conference on*, pp. 30–32.

Patel, A., Taghavi, M., Bakhtiyari, K. & JúNior, J. C. (2013). An intrusion detection and prevention system in cloud computing: A systematic review. *Journal of network and computer applications*, 36(1), 25–41.

Patel, S. K. & Sonker, A. (2016). Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort. *International Journal of Future Generation Communication and Networking*, 9(6), 339–350.

Pearson, S. (2009). Taking account of privacy when designing cloud computing services. *2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pp. 44–52.

Petcu, D., Di Martino, B., Venticinque, S., Rak, M., Máhr, T., Lopez, G. E., Brito, F., Cossu, R., Stopar, M., Šperka, S. & Stankovski, V. (2013). Experiences in building a mOSAIC of clouds. *Journal of Cloud Computing*, 2(1), 1–22.

Rackspac. (2018, January, 30). Rackspac [Format]. Consulted at https://www.rackspace.com.

Ren, K., Wang, C. & Wang, Q. (2012). Security Challenges for the Public Cloud. *IEEE Internet Computing*, 16(1), 69-73. doi: 10.1109/MIC.2012.14.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. & Hotho, A. (2016). The CIDDS Concept. Consulted at https://www.hs-coburg.de/forschung-kooperation/forschungsprojekte-oeffentlich/ingenieurwissenschaften/cidds-coburg-intrusion-detection-data-sets.html.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. & Hotho, A. (2017a). Creation of Flow-Based Data Sets for Intrusion Detection. *Journal of Information Warfare*, 16, 40–53.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. & Hotho, A. (2017b). Flow-based benchmark data sets for intrusion detection. In *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS)* (pp. 361–369). ACPI.

Ring, M., Wunderlich, S., Grüdl, D., Landes, D. & Hotho, A. (2018). Technical Report CIDDS-001 data set. 2018-6-30, Consulted at https://www.hs-coburg.de/forschung-kooperation/forschungsprojekte-oeffentlich/ingenieurwissenschaften/cidds-coburg-intrusion-detection-data-sets.html.

Rosado, D. G. (2012). *Security Engineering for Cloud Computing: Approaches and Tools: Approaches and Tools*. IGI Global.

Roschke, S., Cheng, F. & Meinel, C. (2009). An extensible and virtualization-compatible IDS management architecture. *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, 2, 130–134.

salesforce. (2017, Mar, 19). The Customer Success Platform To Grow Your Business [Format]. Consulted at https://www.salesforce.com.

Scarfone, K. & Mell, P. (2007). Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007), 94.

Shawish, A. & Salama, M. (2014). Cloud computing:paradigms and technologies. *Inter-cooperative collective intelligence:Techniques and applications*, pp. 39–67.

Snort. (2015). Snort-Network Intrusion Detection & Prevention System. Consulted at https://www.snort.org/.

Snort. (2017, May, 1). Official Snort Ruleset covering the most emerging threats [Format]. Consulted at https://www.snort.org/products.

SNORT. (2018). SNORT @R Users Manual. Consulted at https://www.snort.org/documents.

Tan, Z., Nagar, U. T., He, X., Nanda, P., Liu, R. P., Wang, S. & Hu, J. (2014). Enhancing big data security with collaborative intrusion detection. *IEEE Cloud Computing*, 1(3), 27–33.

Tsilimantos, D., Karagkioules, T. & Valentin, S. (2018). Classifying flows and buffer state for YouTube's HTTP adaptive streaming service in mobile networks. *Proceedings of the 9th ACM Multimedia Systems Conference*, pp. 138–149.

Tupakula, U., Varadharajan, V. & Akku, N. (2011). Intrusion detection techniques for infrastructure as a service cloud. *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pp. 744–751.

Ullah, I. & Mahmoud, Q. H. (2017a). A filter-based feature selection model for anomaly-based intrusion detection systems. *Big Data (Big Data), 2017 IEEE International Conference on*, pp. 2151–2159.

Ullah, I. & Mahmoud, Q. H. (2017b). A hybrid model for anomaly-based intrusion detection in SCADA networks. *Big Data (Big Data), 2017 IEEE International Conference on*, pp. 2160–2167.

Varadharajan, V. & Tupakula, U. (2014). Security as a service model for cloud environment. *Network and Service Management, IEEE Transactions on*, 11(1), 60–75.

Varia, J. (2010). Architecting for the Cloud : Best Practices. *Amazon Web Service*, 1, 1–23.

Velmurugan, N. & Thirukumaran, S. (2012). Effective Analysis of Cloud Based Intrusion Detection System. *Int J Comput Appl Inform Technol*.

VMWARE. (2018, Sep). VMware – Official Site. Consulted at https://www.vmware.com/.

Wang, Z. & Zhu, Y. (2017). A centralized HIDS framework for private cloud. *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2017 18th IEEE/ACIS International Conference on*, pp. 115–120.

Weng, C. G. & Poon, J. (2008). A new evaluation measure for imbalanced datasets. *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pp. 27–32.

Wikipedia. (2018a). Function mathematics. Consulted at https://en.wikipedia.org/wiki/Function_mathematics".

Wikipedia. (2018b). Power set. Consulted at https://en.wikipedia.org/wiki/Power_set.

XEN. (2018). VS16 Video Spotlight with Xen Project's Lars Kurth. Consulted at https://www.xenproject.org.

Xing, J., Zhou, H., Shen, J., Zhu, K., Wangt, Y., Wu, C. & Ruan, W. (2018). AsIDPS: Auto-Scaling Intrusion Detection and Prevention System for Cloud. *2018 25th International Conference on Telecommunications (ICT)*, pp. 207–212.

Yadav, S. (2013). Comparative study on open source software for cloud computing platform: Eucalyptus, openstack and opennebula. *International Journal Of Engineering And Science*, 3(10), 51–54.

Yan, Y., Xu, B. & Gu, Z. (2008). Automatic service composition using and/or graph. *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pp. 335–338.

Zaman, S. (2009). A collaborative architecture for distributed intrusion detection system based on lightweight modules.

Zargar, S. T., Takabi, H. & Joshi, J. B. (2011). DCDIDP: A distributed, collaborative, and data-driven intrusion detection and prevention framework for cloud computing environments. *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on*, pp. 332–341.

Zarrabi, A. & Zarrabi, A. (2012). Internet intrusion detection system service in a cloud.

Zeek. (2019). The Zeek Network Security Monitor. Consulted at https://www.zeek.org/.

Zhang, J., Zhang, J., Ding, H., Wan, J., Ren, Y. & Wang, J. (2013). Designing and applying an education iaas system based on openstack. *Appl. Math*, 7(1L), 155–160.

Zhang, Q., Cheng, L. & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1), 7–18.

Zheng, H., Hou, M. & Wang, Y. (2011). An efficient hybrid clustering-PSO algorithm for anomaly intrusion detection. *Journal of Software*, 6(12), 2350–2360.