DEFENSE AGAINST ATTACKS IN SELF SERVICE CLOUD USING REINFORCEMENT LEARNING

By

SAI SRAVAN GUDIPATI

Bachelor of Technology in Computer Science

and Engineering

Gitam University

Visakhapatnam, Andhra Pradesh

2013

Submitted to the Faculty of the Graduate College of the
Oklahoma State University in partial fulfillment of
the requirements for the Degree of
MASTER OF SCIENCE
July, 2016

DEFENSE AGAINST ATTACKS IN SELF SERVICE

CLOUD USING REINFORCEMENT LEARNING

Thesis Approved:

Dr. Johnson Thomas

Thesis Adviser

Dr. David Cline

Dr. Ronak Etemadpour

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Johnson Thomas for his excellent guidance, patience, and providing me with an excellent atmosphere for doing research. His guidance helped me to successfully complete my research.

Besides my advisor, I would like to thank rest of the thesis committee: Dr. David Cline, Dr. Ronak Etemadpour for their encouragement and insightful comments.

Last but not the least; I would like to thank my family for supporting me throughout my life.

Name: SAI SRAVAN GUDIPATI

Date of Degree: July, 2016

Title of Study: DEFENSE AGAINST ATTACKS IN SELF SERVICE CLOUD USING REINFORCEMENT LEARNING

Major Field: COMPUTER SCIENCE

Abstract:

Cloud computing offers various services which are analogous to traditional data centers. The on demand supply of resources make this model of utility computing as the platform for many web based services. However, security is always a major concern. This thesis proposes a new architecture called Self-service cloud computing with virtual shield (VS) to secure the entire cloud environment. When a malicious attack is predicated, the Virtual shield (VS) dynamically changes the configurations of the client virtual machines (VM) using a reinforcement learning mechanism to achieve the required security. The system may be dynamically modified in response to changes in system configuration, state, and/or workload. The reward values generated during the learning process determines the reconfiguration of the client. Simulation results show that the dynamic reconfiguration of virtual machines when anticipated to confront an attack, diminishes the likelihood of an attack and secures the cloud virtual machines.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

1.1 OVERVIEW

Cloud computing is a novel architecture in the field of information technology. Cloud computing offers various services which are analogous to traditional data centers. Software as a service (SAAS), Infrastructure as a service (IAAS), Application as a service (AAAS), Platform as a service (PAAS) promotes cloud computing to various organizations [7]. Cloud computing provides location independent services to the user. Resource allocation, data management, load balancing is under the control of cloud service providers. However, security is always a major concern in cyber cloud technology. The principles, methodologies, and tools for secure cloud computing are yet to be developed. Various cloud security systems such as advanced cloud systems (ASP) through secure virtualization [8], cloud protector through cloud trace back mechanism [10], hierarchical attribute encryption [11] have been proposed to enhance security in the cloud environment. However, these mechanisms degrade the performance of the system and counter only known attacks.

A novel architecture called self-service cloud computing [1] has been introduced to resolve the security faced by guest virtual machines. This system is not concerned about the security of the host operating system since most of the privileges exists within the guest Meta domain created. Moreover, there is no standard way of measuring the cloud system with respect to security. This paper introduces an extension to self-service cloud computing [1] to dynamically configure the privileges between the host operating system and guest virtual machines in SSC (Self Service Cloud Computing).This protects the host operating system from the guest virtual machines.

1.2 SELF SERVICE CLOUD WITH VIRTUAL SHIELD

Self-service cloud computing is a service oriented architecture which mitigated security and privacy issues related to client virtual machines. Inflexible control, which requires cloud providers to define security measurements like VMware introspection, migration and check pointing are handed over to the client's Meta domain in SSC. The Hypervisor and hardware are assumed to be a TRUSTED COMPUTING BASE since they are provided by trusted organizations in this architecture. In self-service cloud computing the host operating system has no privileges to view the guests virtual CPU, memory or the configuration parameters of the Meta domain. The operating system acts to initiate the boot up process and hold the privileges to shut down the virtual machines. Though this architecture provides a MUTUALLY TRUSTED SERVICE DOMAINS (MTSDs) which is a regulatory compliance between cloud providers and users it is not sufficient to handle the misuse of the cloud infrastructure by the guest operating systems.

This research focuses on shifting the privileges between the host operating system and guest virtual machines. A Virtual Shield (VS) is introduced to act according to the information

provided by the MTSDs. The Virtual shield is a virtual machine designed to dynamically configure the guest virtual machines with the help of a reinforcement learning algorithm.

1.3 REINFORCEMENT LEARNING

In the proposed architecture the virtual shield learns from the environment using reinforcement learning. This learning in turn facilitates the configuration of the host and guest virtual machines dynamically. The shifting of privileges reduces the probability of an attack in self-service cloud computing.

1.4 PROBLEM STATEMENT

Guest operating systems misuse the cloud infrastructure for malicious activities. At the moment, there is no way to identify the attacks because most of the privileges exist within the Meta domain of the guest. This increases the chances of an attack and results in various attacks on the host virtual machine and hypervisor.

1.5 RESEARCH OBJECTIVE

The objective of this work is to reduce the probability of an attack even before it actually happens in self-service cloud computing by introducing a virtual shield in the system. The virtual shield has the capability to learn from the environment and dynamically configure the host and guest operating systems. This learning is based on the reinforcement learning methodology.

1.6 OUTLINE

The rest of the thesis is organized as follows: Chapter 2 provides the review of literature, Chapter 3 presents the proposed work, Chapter 4 covers simulations and Chapter 5 with the results. Finally, Chapter 6 concludes our paper and provides some insights into future work.

CHAPTER II

LITERATURE REVIEW

Cloud computing refers to computing over the internet where dynamically scaled shared resources (mostly virtual) are provided as a service by using virtualization platforms. The cloud architecture is based on virtualization of the resources. Below is the basic cloud architecture [7].

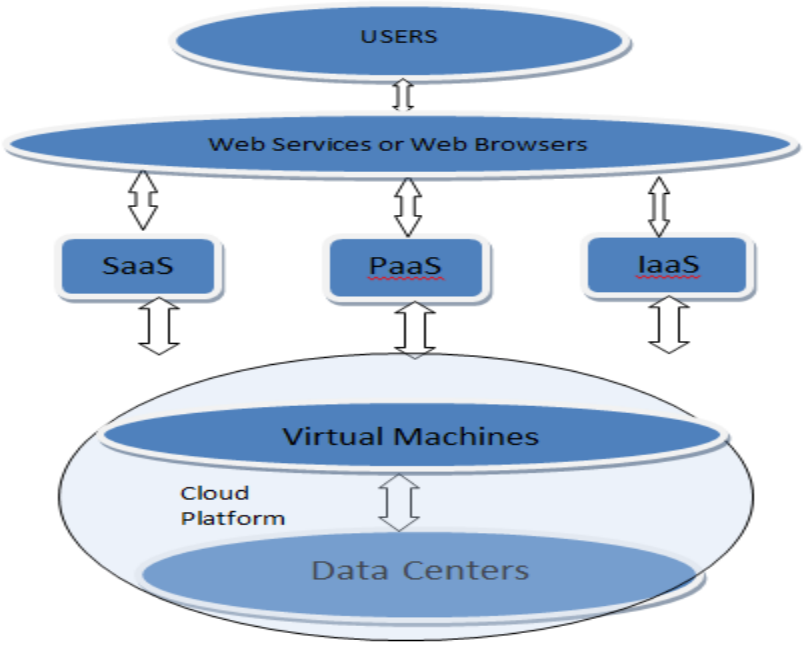

Figure 2.1: Basic Cloud Frameworks

Cloud computing utilizes a service oriented architecture to utilize the services of the cloud. There are different types of virtualizations that are used in cloud computing such as Storage, Network and server virtualization [8] which yields a different set of security concerns for each type of virtualization technique used.

2.1 SECURITY ISSUES IN CLOUD COMPUTING

The client or user is unaware of which physical system the process is actually running on and where the data is stored. If a malicious user is from same Physical system, he can get the data from the physical system. This is because VM's (Virtual Machines) map the data on storage provided logically but all the data resides physically on single storage. Data centers are located across the globe. A user should be able to define where his data and process should reside because each country has different security policies.

Different data centers have different security policies and different VM's run on different zones of security leading to loss of policies and increased security concerns. Because all these VM's from different security zones communicate with each other on a Virtual Network, a weak link will pose a severe threat to the whole application [9].

Since the cloud uses virtualization, it needs to keep up to date with the latest patches for all the virtual machines which is very difficult to manage. Security configuration management is a serious problem and administrators have to keep track of each VM and all the security policies related to data and localization. The Cloud uses different service models such as SaaS, Paas, Daas, Iaas, and NaaS [7], which introduces different levels of security systems for each kind of service models.

Virtualization introduces many more problems into the cloud [8]. Using virtualization introduces many new OS types over which the applications are run. These new OS's are a security concern. Different virtual OS's have different security mechanisms. If one of the new OS is attacked, then the attacker will try to get access to the underlying physical host. This means it will affect the security of all other virtual machines running on this physical host.

The VM's communicate with each other over the network which opens avenues for the guest to guest attack where one virtual machine tries to attack other virtual machines. Moreover, it is difficult to keep track of the VM's. In this scenario two VM's communicate with each other over a network. VM1 can get information regarding VM2 by sending queries while communicating. VM1 might be an intruder or a malicious user. Since it's difficult to keep track of VM's it is difficult to determine who the malicious user is and what information has been compromised.

All the services (Saas, Paas, IaaS, Security as a Service, DaaS, NaaS) [7] are offered using web services or Web browsers. Hence VM security alone is not enough. Using these web services users will get access to the VM's on which these services run. we therefore need to secure the way users communicate with these VMs. Cloud provider need to trust when a user uses an application developed on the cloud. A different user in another VM from the cloud can communicate with the application running on the Cloud. Hence, we need to have trust between the users in the cloud when they communicate with each other.

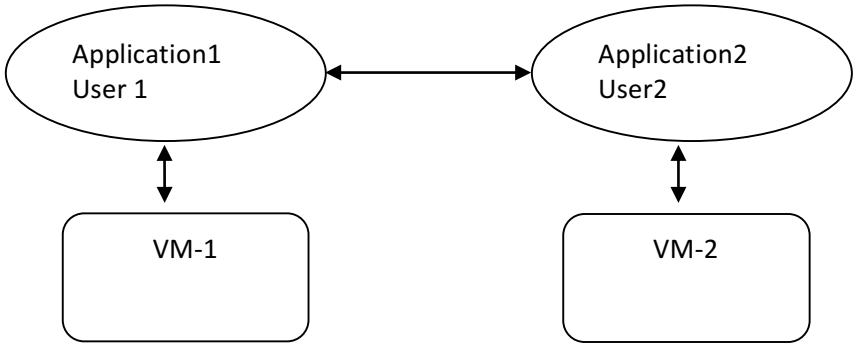

Figure 2.2 Trusts between Users

The dynamic and elastic nature of cloud introduces new threats [7]. When new resources are added to the existing cloud they must be compatible with existing security policies before use. This introduces dynamic security assignment before use.

The proposed work is related to self-service cloud computing [1]. The self-service cloud computing architecture is modified to enhance security and reduce the chances of attack on the system. To enhance the security of the cloud system, reinforcement learning methods can be used. Furthermore, the virtual shield can be configured with different security metrics to defend against various attacks on the host virtual machine.

2.2 SELF SERVICE CLOUD COMPUTING

Self-service cloud computing is a computing model that resolves two shortcomings in the traditional cloud architecture. Virtualization is the key to any cloud architecture [7]. Virtual machine monitors are used in many cloud architectures to administer and execute client virtual machines. These virtual machine monitors comprises of a Hypervisor, Hardware and a host virtual machine called dom0. The hypervisor and hardware are assumed to be a trusted computing base whereas the dom0 is considered to be the source of different attacks. Since most of the

privileges lies within dom0 there is a high risk of utilizing this administrative domain for malicious activities.

The two major problems in traditional cloud computing are

- Security and privacy of the client virtual machine

  The state of the client virtual machine can be inspected by dom0. It holds the privileges to inspect the contents of the client VMs and their configurations. The client virtual machines security and privacy can be compromised due to various attacks by the host virtual machine's (Dom0). Misconfiguration and malicious system administrators can be the source of attacks.

- Inflexible control over the client VMs

  Virtualization facilitates different services to the client. It has the potential to enable services like, migration, check pointing and VM introspection [1]. However, the deployment of these services in the present cloud architecture is under the control of cloud infrastructure providers. The client virtual machines have no control over the adoption of these services. Upon the request of the Client, the virtual machines are configured with these services. However, these services won't fit for all the clients. A few client VMs may use encryption to securely transfer data packets, but the service that checks the malicious content using signatures may not be able to use the encryption mechanism. The client virtual machines may need different security mechanisms for different kinds of attacks. Thus the present cloud architecture has inflexible control over the client VMs.

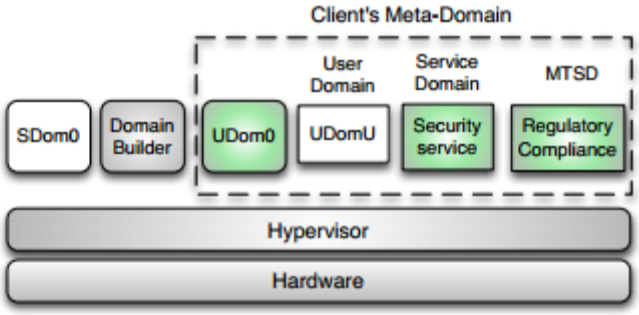

Figure 2.3 Self Service Cloud Computing

Self-service cloud computing addresses these two shortcomings by assigning more privileges to client virtual machines. The protocol is designed to protect the client virtual machines from malicious system administrators and to provide control of the services required by the client. The SSC (Self-service cloud computing) divides the entire system into two TCBs (Trusted COMPUTING BASE). The system consists of the system level TCB, with the hardware, the SSC hypervisor, the domain builder and a client-level TCB, with the Udom0 and service domains.

UDom0 is the client side per user administrative domain that can monitor and control the set of VMs of a particular client. This virtual machine attempts to start a VM in SSC. It also has the privileges to perform system services on the client virtual machines.

UDomUs are the actual client side virtual machines with the guest operating systems.

SDs (Service Domains) can be configured with required security services in the system.

MTSDs (Mutually trusted service domains) act as a regulatory compliance between cloud providers and clients. This holds the policies and mechanisms that the provider will use to control

the clients VMs. The information provided by the MTSDs is the key source for the virtual shield in our prototype model.

All these comprise to form the client side Meta domain.

DomB (Domain Builder) is a virtual machine provided by the cloud provider to build the guest virtual machines upon the request from client.

SDom0 (System side administrative domain) administers the client virtual machines. It takes care of starting and stopping of the client VMs.

2.3 ATTACKS IN THE CLOUD

2.3.1 DENIAL OF SERVICE (DOS) ATTACK

A Denial of Service attack [16] is an attempt to obtain excessive computation resources from the cloud and make them unavailable to its intended users. When the cloud computing operating system recognizes the high workload on specific servers, it will provide more computational resources to virtual machines and service instances to adapt to the extra workload; this can be due to a Denial of Service attack causing performance degradation of the system. The Self-service cloud can be vulnerable to a Denial of Service attack, which can be damaging and might result in complete shut down or degradation of a client virtual machine.

A malicious client might try to compromise the availability and integrity of cloud computational resources. A Denial of Service is usually caused by cloud resource usage exceeding the threshold value or exceeding the threshold rate of change (the threshold rate of change is an estimate of uptrend and downtrend during peak or non-peak periods). A Denial of service attack can be harmful in a cloud environment as one virtual machine can be used as a source of denial of

service attack to another virtual machine in the same infrastructure, causing maximum workload to the co-resident virtual machines [17].

Denial of Service attacks misuse the network bandwidth capacity and deteriorate the quality of service by creating congestions at the network level. But with improvements in network bandwidth capacity, the focus of Denial of Service attack have moved from network level to application level. Denial of Service attack uses legitimate application-layer requests to overwhelm server resources causing application Denial of Service attack.

Network based defense models have attempted to identify these attacks by controlling traffic volume or separating traffic patterns at the intermediate routers. But, these defense models protect at the network level, which the application Denial of Service attack can bypass. It also suffers from a high false-positive error rate because sometimes the unseen normal behavior are often predicted to be an attack. Since every traffic is reviewed against the normal behavior model, this expands time complexity and introduces extra service delays for non-malicious clients. Furthermore, in a dynamic environment incorrect prediction of an attack can reduce efficiency of the overall system.

Testing Virtual servers for Denial of Service attack:

The application Denial of Service attack always aims at disrupting application service rather than depleting network resources.

- A Denial of Service attack saturates the server buffer with a flood of malicious requests. Malicious requests will negatively affect the victim server machines; consequently, their average response time (ART) will be higher than that of normal cases. Therefore, ART can work as an indicator of an application Denial of Service attack. Therefore, we calculate the estimated response time (ERT) of the virtual server by inspecting the resource usage. ERT is monitored to detect initial malicious activities during testing.

2.3.2 SIDE CHANNEL ATTACK

The client operates on a virtualized cloud environment sharing its hardware with one or more virtual machines, co-resident on the same physical server. On the basis of a service level agreement with the cloud provider, the client presumes that their virtual machines have exclusive rights over the physical server. Although clients have special administrative powers and privileges to maintain their own virtual machines, they have no control or visibility on how the hypervisor does its functions (the hypervisor, also called a virtual machine manager, is a program that allows multiple virtual machine to share a single physical hardware of the cloud provider). The hypervisor controls the cloud provider's processor and resources, allocating what is needed to each virtual machine while making sure that they cannot disrupt each other [19]. Clients know only about resources that have been allocated to them.

A malicious client virtual machine may try to exploit its co-residency to extract sensitive data from co-resident virtual machines without their knowing. Victims are clients running confidential services in the cloud. We assume that, like any client, a malicious user can run and control many instances in the cloud, simply by requesting cloud resource instances from the cloud provider. Further, it is possible that an attacker's instances might run on the same physical server as target victims. The attacker utilizes the shared physical server to exploit the victim's confidential information.

Testing Virtual servers for Side channel attack:

We assume that the attacker (malicious client) predicts the availability zone and instance type of the potential target victims.

Availability zone: The cloud is hosted in multiple locations worldwide, which are composed of regions and Availability Zones. Each region is a separate geographic area and has multiple, isolated locations known as Availability Zones. Each Availability Zones in a region are connected

through low-latency links. When we launch an instance, we can select a region that puts our instances closer to specific target customers. (EC2 for example is divided into 3 availability zones i.e. zone1, zone2, and zone3)

Instance type: The cloud provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and gives flexibility to choose the appropriate mix of resources for our applications. (EC2 for example is divided into 5 instance types i.e. m1.small, c1.medium, m1.large, m1.xlarge and c1.xlarge)

An attacker (malicious client) can flood using probe instances in two ways. An attacker generates an attacker instance, which is like a target instance in terms of resource requirements and checks whether it is co-resident with the target. The two ways are:

- Over some period of time, the attacker repeatedly runs probe instances in the target availability zone and of the target instance type.
- We assume that an attacker can also launch probe instances soon after the launch of a target victim instance. The attacker then engages in instance flooding: running as many instances in parallel as possible in the target availability zone and of the target instance type.

Each probe instance checks whether it is co-resident with the targets by comparing its instance UDom0 IP with target instance UDom0 IP.

- A malicious client can determine its UDom0 IP from the first hop of its instance on any route. The malicious client uses its UDom0 IP to compare it with target UDom0 IP to confirm co-residency.
- UDom0 IP of target instances is determined by performing a TCP SYN trace route and inspecting the last hop. (In TCP SYN, trace route malicious clients send IP packets with a short life, and wait for ICMP (Internet Control Message Protocol) packets to report the

death of these packets. An IP packet has a field called "TTL" (as "Time To Live") which is decremented at each hop; when it reaches 0, the packet dies, and the router on which this happens is supposed to send back a "Time Exceeded" ICMP message. That ICMP message contains the IP address of the said router, thus revealing it. TCP SYN trace route can generate more number of ICMP and UDP packets in the network [18].

Therefore, during testing, instance count of the clients can work as an initial indicator and then by monitoring bandwidth usage (number of ICMP and UDP packet generated) of the suspected clients we can detect the probability of the side channel attack.

2.3 REINFORCEMENT LEARNING

Reinforcement learning is learning what to do and how to map situations to actions to maximize the numerical reward. Reinforcement learning is defined not by characterizing learning methods, but by characterizing the learning problem [6].

2.3.1 ELEMENTS OF REINFORCEMENT LEARNING

Policy, a reward function, a value function, and a model of the environment are the different elements of reinforcement learning.

A policy defines the learning agent's way of behaving at a given time. It's a mapping from perceived states of the environment to actions to be taken when in those states. A reward function defines the goal in a reinforcement learning problem. It maps each perceived state (or state – action pair) of that state. A reinforcement learning agent's sole objective is to maximize the total reward it receives in the long run. A value function specifies what is good in the long run. The value of the state is the total amount of reward an agent can expect to accumulate over the future

starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account, the states that are likely to follow and the rewards available in those states. For example, a state might might always yield a low immediate reward but still have a high value because it is regularly followed by other states that yield high rewards. A model predicts the resultant next state and reward for a given state and action.

CHAPTER III

PROPOSED WORK

3.1 INTRODUCTION

The goal of this work is to design an architecture, that reduces the probability of an attack in self-service cloud computing. In addition to the components involved in SSC, the new architecture will have a *virtual shield* (VS) that exists between the host virtual machine and meta domain as shown in fig 3.1.The SSC protocol will be modified to facilitate the interaction between MTSDs and the virtual shield.

3.2 SYSTEM BUILDING

The basic assumption is that the applications and operating system are secure. Although the cloud service provider is trusted, the cloud administrator is not. This required for the provider to supply a Trusted Computing Base (TCB) running a Virtual Machine Monitor (VMM) and the physical hardware to be equipped with an IOMMU (Input Output Memory Management Unit) and a Trusted Platform Module(TPM) chip. Cloud system administrators are entrusted with system tasks and maintaining the cloud infrastructure. Hence, they have access to the administrative domain (dom0) which is a privileged Virtual Machine (VM) that is used to control and monitor client VMs. and the privileges that it entails. Cloud system administrators are adversarial (or could make mistakes), and by extension, that the administrative domain is untrusted.

Administrators have the means to misuse dom0's privileges to snoop or even alter client data. This threat has been addressed in the SSC model.

The Trusted Computing Base (TCB) of the cloud infrastructure is split in two parts, a system level TCB, which consists of the hypervisor, domB, BIOS, the boot loader and virtual shield (VS) which is controlled by the cloud provider, and a client level TCB, which consists of the client's Udom0, SDs, and MTSDs. Reconfiguration in this context includes the standard definition, that is, reconfiguring resources such as allocated processors, memory, disks, network adapters and the user interface. Reconfiguration in this work also means adding or easing restriction to the kinds of privileged instructions a virtual machine can execute. In this work, the VS will determine a course of action if a malicious VM is detected. The VS may recommend the VM be removed from the cloud or it may recommend a reconfiguration of the VMs.

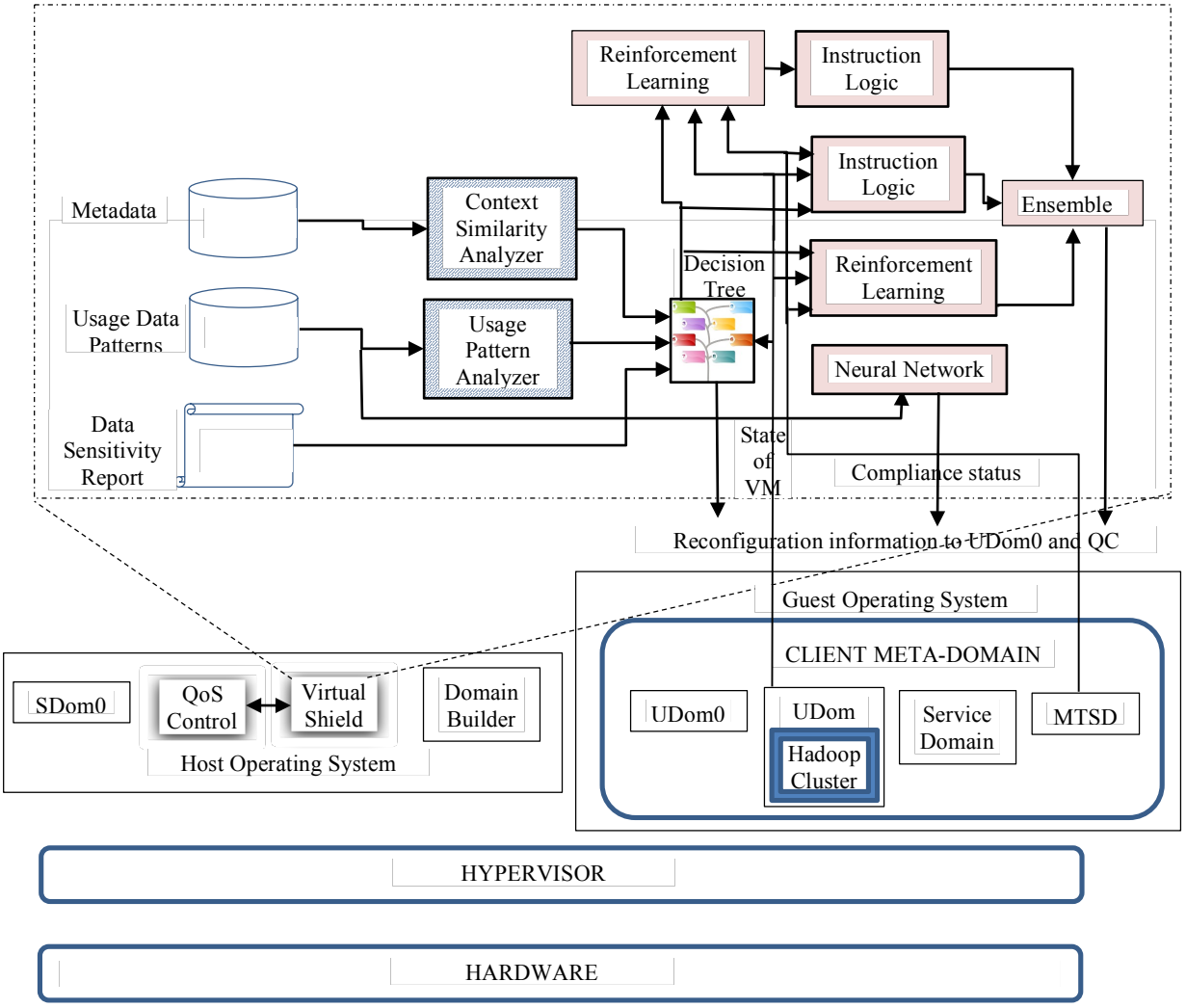

Figure 3.1 Self Service Cloud with Virtual Shield

Figure 3.1 shows the proposed cloud architecture. The main difference between the proposed architecture and SSC [40] is the addition of two new units, the Virtual Shield (**VS**) and the QoS Control (**QC**). **MTSD**s (Mutually-trusted service domains) execute privileged services that check regulatory compliance in a manner that is mutually agreed upon between the cloud provider and the client.

3.3 COMPONENTS

3.3.1. SDom0

SDom0 is the system side administrative domain. This domain controls the client virtual machines. The start and stop of the client virtual machines is done by SDom0. Though this component has the same functionalities as SDom0 in SSC, it has additional capabilities which don't exist in SSC SDom0[1].

The SSC SDom0 has no privilege to view the state of the client virtual machines, i.e. the contents of virtual CPU, virtual memory etc. But in our proposed new architecture the SDom0 will be designed to have access if the client virtual machine is found to be malicious. The virtual shield provides these capabilities to SDom0 by providing the privileges to access the client virtual machines states.

3.3.2. Mutually trusted Service Domain (MTSD)

MTSDs (Mutually trusted service domains) execute privileged services that check regulatory compliance in a manner that is mutually agreed upon between the cloud provider and the client.

3.3.3. Virtual Shield

The virtual shield is designed with different functionalities and security measurements. MTSD designed in the SSC are the key source of information to the virtual shield, which provides the information about the type of attack and the severity of the attack. In SSC the MTSD act as the regulatory compliance between client virtual machines and cloud providers. In SSC once, the client virtual machines are identified to be misusing the cloud infrastructure for malicious activities the virtual machines are shut down and they lose its state.

In our architecture, the virtual machines are not shut down immediately. Once the MTSD identify the client virtual machine to be malicious, it triggers the virtual shield with the information.

The information from the MTSD is used by the virtual shield for the reinforcement learning process designed to virtually configure the virtual machines to maximize security. The virtual shield holds a table with appropriate actions to be taken based on the state of the machine. Each state has a reward value. The actions are the virtual configurations between the host virtual machine and the client virtual machine. Virtual shield holds one table for each virtual machine.

3.3.4. Domain Builder

DomB, the domain builder builds the client side Meta domain. Once the client sends the request to build the virtual machines, these parameters are send to the domain builder and virtual shield. Domain builder uses these parameters to build the client side Meta domain. The construction of Meta domain is similar to SSC, whereas the MTSDs are configured to trigger the virtual shield when the client misuses the cloud infrastructure.

3.3.5. Client Meta Domain

The Client Meta domain holds UDom0, UDomU, SDs and MTSD. All these components are assumed to have the same functionalities as in SSC, except the MTSD. The MTSD is modified to trigger the virtual shield when the guest virtual machines try to perform security attacks.

3.4 METHODOLOGY IN VIRTUAL SHIELD

The MTSDs are configured to regularly update the **Virtual Shield** (**VS**) with status information about complying to the agreement between the cloud provider and the client. The VS keeps a log of usage patterns, generates metadata about data in the cloud repository that the client is accessing and generates a sensitivity report of the data. The decision tree in the VS re-assigns the access control rights based on a data sensitivity report, information from the MTSD, along with the output from the usage pattern analyzer, context similarity analyzes and internal states of the VM. The status information from the MTSD, along with the output from the usage pattern analyzer, internal states of the VM and access control rights output by a decision tree is input to a reinforcement learning process which will recommend a reconfiguration to maximize security if malicious activity is detected. The VS also holds a table with appropriate actions to be taken based on the state of the machine. Each state has a reward value. The actions are the virtual configurations between the host virtual machine and the client virtual machine.

The proposed Virtual Shield will dynamically re-configure guest VMs when the big data cloud is under attack by a VM.

The cloud is designed to dynamically re-configure the system when an attack takes place based on the observed states. This work defines re-configuration to be one of the following: (a) *re-configure resources* such as allocated processors and memory to a VM; (b) *re-define the set of privileged instructions* a VM can execute; (c) both *re-configure resources and re-define the set of privileged instructions* a VM can execute; (d) *shut down* a VM.

We use an ensemble approach for reconfigurations (a), (b), (c) or (d). An ensemble approach is used because the results of a set of reconfigurations when combined together yields a better security solution rather than just applying one approach. The input to the reconfiguration is compliance status from the MTSD, the internal states of the VMs.

3.4.1 Using Reinforcement Learning

Reinforcement Learning (RL) is used for re-configuration (a), that is, re-configure allocated resources to a VM. RL provides a knowledge-free trial-and-error methodology in which a learner tries various actions in numerous system states and learns from the consequences of each action. A big advantage therefore is RL will learn even if there is no available training set. That is RL does not depend on supervised training with known attack types. RL can therefore learn new previously unseen attacks. However, RL suffers from poor scalability whose search space grows exponentially with the number of state variables. Moreover, due to the absence of domain knowledge, the initial security improvement achieved by RL may be poor. Instead of conducting RL search in the whole configurable state space, we first reduce the search space to a much smaller but "promising" state set. In our approach domain knowledge and security parameters are used to guide the reduction in the search space. This avoids performance degradation caused by random exploration.

One of the reinforcement learning methodologies will be used for the learning process in the virtual shield [6].

A Wide range of applications can be framed as reinforcement learning problems. The application in the virtual shield is framed to one of reinforcement learning tasks and provided with the method to learn.

The aim of the reinforcement learning problem is learning from interaction to achieve a goal. The decision maker is called the agent. The agent interacts with the environment, that is, everything that is outside the agent. This is a continuous process; the agent selects the actions and the environment responds to those actions and provides new situations to the agent. The environment also provides the numerical rewards, which the agent tries to maximize over time. A task is defined as a configuration change, which is one instance of the reinforcement learning problem.

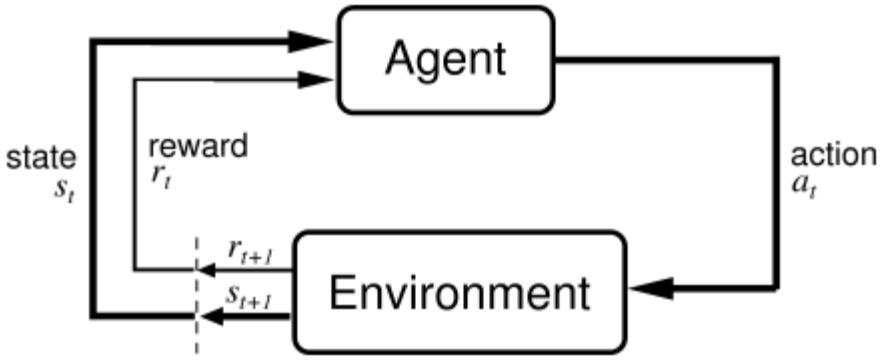

Figure 3.2 The agent-environment interaction in reinforcement learning

At discrete time steps, *t=0, 1, 2, 3 ...* the agent and the environment interact with each other. At each step *t*, the agent is provided with some representation of the environment's state $S_t \in S$, where *S* is the set of possible states and S_t is one of the states of **S**. $A(S_t)$ are the set of actions available in that state. After selecting the action $A_t \in A(S_t)$, the agent receives a numerical reward, $r_{t+1} \in R$ and enters a new state.

At each time step, a mapping from states to probabilities of selecting each passible action is implemented. This mapping is called the agent's policy $\prod_t$, where $\prod_t(S,A)$ is the probability that $A_t = A$ if $S_t = S$ where *S* is State and *A* is Action). In reinforcement learning the agent changes its policies as a result of experience. The agent's goal is to maximize the total amount of reward it receives over time. In our system the mutually trusted service domain is framed as the environment and the virtual shield is framed to be the agent.

3.4.1.1 Q-Learning as a model free based approach

Q-Learning is a model free reinforcement technique. It can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). It works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. When an action-value function is learned, the optimal policy can be constructed by selecting the action with the highest value in each state. One of the advantages of Q-Learning is that it is able to compare the expected utility of the available actions without requiring a model of the environment.

Algorithm:

The model consists of an agent, states S and a set of actions per state A. By performing an action $a \in A$, the agent can move from state to state. Executing an action in a specific state provides the agent with a reward (a numerical score). The goal of the agent is to maximize its total reward. It does this by learning which action is optimal for each state. The action that is optimal for each state is the action stat has the highest long term reward. This reward is a weighted sum of the expectation values of the rewards of all future steps starting from the current state, where the weight for a step from a state Δt steps into the future is calculated as $\gamma^{\Delta t}$. Here, γ is a number between 0 and 1 ($0 \leq \gamma \leq 1$) called the discount factor.

The algorithm has a function that calculates the Quantity of a state-action combination

$Q: S \times A \rightarrow R$

Before the start of learning, Q returns an (arbitrary) fixed value. Each time the agent selects an action, and observes a reward and a new state that may depend on both the previous state and selected action, "Q" is updated. It assumes the old value and makes a correction based on the new information.

$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \cdot \left(\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right)$$

Where R_{t+1} is the reward observed after performing a_t in s_t, and where $\alpha_t(s, a)$ $(0<\alpha\leq 1)$ is the learning rate.

Learning Rate:

The learning rate determines to what extent the newly acquired information will override the old information. A factor of 1 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information. A constant learning rate is used for implementation of the algorithm, such as $\alpha_{t\,(s,\,a)} = 0.1$ for all t.

Discount Factor:

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge.

Initial Conditions (Q_0)

As Q-Learning is an iterative algorithm, it implicitly assumes an initial condition before the first update occurs.

PSEUDO-CODE:

// s, s' → states

// a, a' → actions

// Q→ state-action value

// γ, α → learning parameters (learning rate, discount factor)

1. Initialize Q (s, a) arbitrarily
2. Observe current state s
3. repeat
    i. Take action a observe reward r, state s'
    ii. $Q(s,a) \leftarrow Q(s,a) + \alpha[r+ \gamma . \max_a . Q(s',a')-Q(s,a)]$
    iii. $S \leftarrow s'$
4. Until termination

Action Selection Strategies:

In each state (except the terminal state), the agent must select an action. There are several ways in which to decide which action to take. The simplest form is greedy selection: the agent always selects the action that the highest state-action value. This method is pure exploitation. Boltzmann selection is another action selection strategy where there would be a balance between exploration and exploitation.

Boltzmann selection:

Boltzmann selection involves probability, but takes into account the relative values of the state-action values. The probability that an action is selected depends on how it is compared to the other state-action values. If one value is much higher, it is most likely to be taken, but if there are two actions with high values, both are most equally likely.

At a state s, an action a is selected with probability

$$p = \frac{e^{\frac{Q(s,a) - max_b Q(s,b)}{T}}}{\sum_a e^{\frac{Q(s,a) - max_b Q(s,b)}{T}}}$$

where T is called the temperature, and increases as the exploitation rate increases. High temperatures cause the actions to be all (nearly) equiprobable. Low temperatures cause a greater difference in selection probability for actions that differ in their value estimates.

CHAPTER IV

SIMULATION METHODOLOGY

4.1 IMPLEMENTATION

CloudSim [14] is used to simulate the cloud environment. Cloudsim is a simulation environment to simulate the cloud architectures before actual deployment. Cloudsim provides java APIs to design the various elements of the cloud computing architecture. The underlying architecture contains different subsystems. Each subsystem is designed and simulated to satisfy the requirements of the whole architecture.

Different subsystems in the architecture includes

1. SSC SUB SYSTEM

2. MTSD SUB SYSTEM

3. VIRTUAL SHIELD SUB SYSTEM

4. ATTACK SYSTEM

5. CONFIGURATION SYSTEM

4.1.1 SSC SUB SYSTEM

The SSC sub system is the main system which initializes the entire architecture. Upon request from the client, the administrative domain (Broker) in the SSC sub system requests the Data Center (Hypervisor) to allocate resources to the client. During the initialization, the other subsystems are also activated or initialized. The current client configuration will be written to the virtual shield. The mutually trusted service domain is initialized with the different attack models and configuration parameters, which are in turn used to detect the malicious clients.

4.1.2 MUTUALLY TRUSTED SERVICE DOMAIN (MTSD)

During the client initialization, the mutually trusted service domain is also initialized with the different attack models to check the client's attacks.

The Mutually Trusted Service Domain periodically checks the network packets transmitted to identify the malicious clients. During this process if the MTSD identifies that the client is misusing the application, it notifies the virtual shield with the attack type. This information is being received in a configuration file, which contains all the system information and the attack specifications.

4.1.3 VIRTUAL SHIELD

The mutually trusted service domain triggers the virtual shield periodically with the activities of the client. If the MTSD identifies that the client is misusing the cloud infrastructure, depending upon the type of attack, severity of the attack and the existing configuration, it triggers the virtual shield and the virtual shield reads the configuration file and requests the administrator to change the configuration of the client for the predicted attack.

The Virtual shield uses the simple reinforcement learning mechanism to allocate the different configuration parameters to the client.

4.1.4 CONFIGURATION SYSTEM

Configuration defines the properties of the virtual machines such as computing capacity in terms of million instructions per second, image size, memory size, number of cpus, and bandwidth. The configuration system is a database which holds the different configuration parameters for different clients. It has the different configurations for different type of attacks. The virtual shield allocates these configurations to the clients by analyzing the existing configuration and type of attack the client performed.

Different parameters in the configurations include: MIPS, IMAGE SIZE, MEMORY SIZE, CPUS, and BAND WIDTH.

MIPS (Million instructions per second) define the number of instructions to be executed per second.

Image size defines the size of the operating system image.

Memory size defines the size of the internal memory.

CPUs define the number of cpus required by the virtual machine.

Bandwidth defines the network bandwidth (number of bits transmitted per second).

4.1.5 ATTACK SYSTEM

The attack system is a database which holds different attacks metrics. These attack metrics are used by the Mutually Trusted Service Domain to identify the malicious client and notify the virtual shield with the type of attack and severity of the attack. There can be different metrics to identify the attack types. In our architecture for the purpose of simulation we used a 15 alphanumeric coded value to identify the attack metrics. An example of 16-digit alphanumeric coded value would be as follows.

00000000000001F

00 - Virtual machine id in hexadecimal

00 - Host machine id in hexadecimal

00 - Client id in hexadecimal

00 - Datacenter id in hexadecimal

00 - Resource event / reason for the attack in hexadecimal

00 - Type of the attack in hexadecimal

01 - Current state of the when the attack is predicted

F - Usage parameter (Alphabetical A-J)

Type of attack Encodings:

00 - Denial of Service

01 - Side Channel Attack

Resource event Encodings:

00 – RAM

01 – Bandwidth

02 – Cpu Usage

03 – Memory Size

04 – MIPS

State Encodings

00 – Good

01 – Warn

02 – Critical

03 – Resume

04 – Alarm

Usage Parameter Encoding

A – 0%-10%

B – 10%-20%

C – 20%-30%

D – 30%-40%

E – 40%-50%

F – 50%-60%

G – 60%-70%

H – 70%-80%

I – 80%-90%

J – 90%-100%

4.1.6 VIRTUAL MACHINE TERMINATION

The mutually trusted service domain periodically checks the clients meta domain for attacks. These periodical updates are notified to the virtual shield to calculate the rewards for individual configurations. The individual rewards of the allocated configurations to the client are aggregated to identify the overall rewards of the client's virtual machine. This aggregated reward is used to determine the threshold for the client termination. Once the virtual shield identifies the total score is less the threshold designed by the cloud provider or the administrator, the client's virtual machine is terminated.

The client is notified every time the configuration changes. If the client still tries to misuse the cloud infrastructure, the overall reward eventually decreases and finally results in the termination of the client's virtual machine. The threshold is defined by the cloud provider for each and every client virtual machine. If the overall reward of the client virtual machine is less than the threshold, the virtual shield informs the administrator to terminate the client's virtual machine. This process is explained below.

$$O(c) = \sum A(c) \quad \text{for } I = 0 \text{ to } N$$

$$\text{if } O(c) < T(c) \text{ terminate}$$

$O(c) \rightarrow$ Client Score

$A(c) \rightarrow$ Reward of individual Configurations

$T(c) \rightarrow$ Threshold

$N \rightarrow$ Number of Configurations.

4.2 COMMUNICATION PROTOCOL

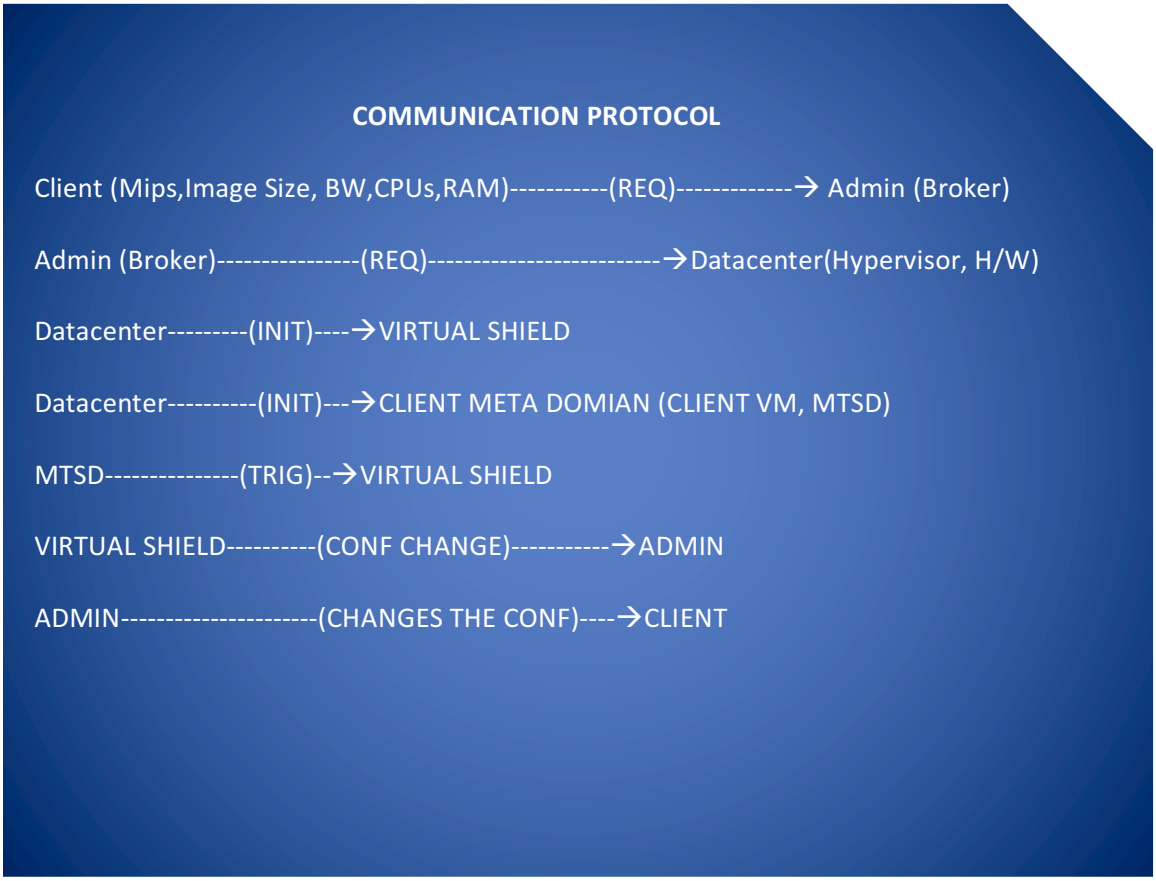

Figure 4.1 Communication Protocol

The communication protocol in the above figure4.1 explains how each subsystem in the architecture interacts with each other. After the initialization of Sdom0, the client requests the system side administrative domain for the virtual machine by passing the configuration parameters.

The Dom0 requests the datacenter i.e. the Hypervisor to provide the requested resources to the client.In this process, the hypervisor initializes the virtual shield and clients meta domain.

If the client tries to misuse the resources, MTSD triggers the virtual shield. Based on the type of the attack the usage parameters, and action would be taken using Q-Learning from Reinforcement Learning.

4.3 SIMULATION ALGORITHM

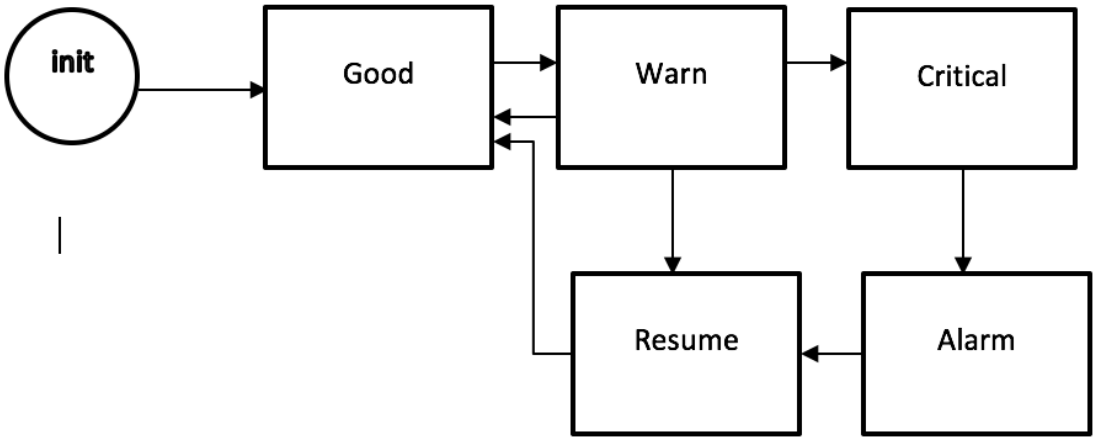

Figure 4.2 State Diagram

There are five different states on which the system has been designed. Initially the system is in state init, which represents the initialization of the variables and environment. The next flow of the states is shown in the figure 4.2. Good represents that the system is in a good state, when there is a probability of an attack it moves to the warn state. If the attack has been reconfigured and successfully defended, it moves back to the good state, through the resume state which changes the configurations accordingly. The system retains in the good state as long as there are no chances of attack on the system. If the attack has not been defended, it moves to the critical state and then to the alarm state sequentially. Each state has its own set of decision tree mapping; by which we could select on from the possible actions that could be taken in the system. The

reinforcement learning system determines what actions could be probably taken in each state by learning over a period of time.

The Virtual Shield gets information from the MTSD, to determine if the event (observed behavior) matches a known attack pattern. If it is determined that the observed behavior is representative of a known attack pattern, respective actions are taken corresponding to the current state. If it does not match a known attack pattern, the information layer agent may determine the probability that the observed behavior represents a previously unknown attack pattern.

Each action has its own reward value which is used to calculate the Q value which has the ordered pair of state and its action. For simulation purposes, the following are the reward values for each action taken. Since Shutting down a VM, is the ultimate possible way that a system can defend itself it is assigned the highest reward value. The virtual agent learns through experience; this is known as unsupervised learning.

(a) re-configure resources such as allocated processors and memory to a VM -50.

(b) Re-define the set of privileged instructions a VM can execute -100

(c) Both (a) & (b) -150

(d) Shut down a VM -200

The algorithm goes as follows

1. Set the gamma parameter (which is in between 0 & 1) and environment rewards in matrix R, which is the mapping reward values of state and actions.
2. Initialize Q values to zero
3. For each iteration (when there is a probability of an attack)
    a. Select the initial state, which can be obtained through previous data in which state the VM is, if there is no data present, assume the state to be in good state.

b. Do while the goal state hasn't reached

   i. Select on among all possible actions for the current state

   ii. Using this possible action, consider going to the next state

   iii. Get maximum Q value for this next state based on all possible actions

   iv. Compute: Q (state, action) = R (state, action) + Gamma* Max [Q(next state, all action)]

   v. Set the next state as the current state

   End do

4. End For

CHAPTER V

RESULTS

The purpose of the simulation using the CloudSim toolkit [14] is to validate the proposed detection model. We have made several assumptions to simplify the implementation of the simulation environment. The simulation results provide a reliable overview of the practical performance of the proposed detection model.

The CloudSim toolkit is a simulation environment used to simulate cloud architectures. CloudSim provides java APIs to design the various elements of the cloud computing architecture. We show the efficiency of our detection model by varying the simulation environment settings as follows:

5.1 Single Attack on a single Virtual Machine

Each attack (Denial of Service and Side Channel) have been simulated separately, where the attackers tries to target a single Virtual machine. The figure 5.1(a) shows the Attack graph, where 0 represents the attacks has happened, and 1 represents attack has been defended. We can observe that the system has been stable from 23^{rd} run for the denial of service attack, and 15^{th} run for the Side Channel Attack. The Stabilization implies that the Attack has been defended successfully and the system has become stable.

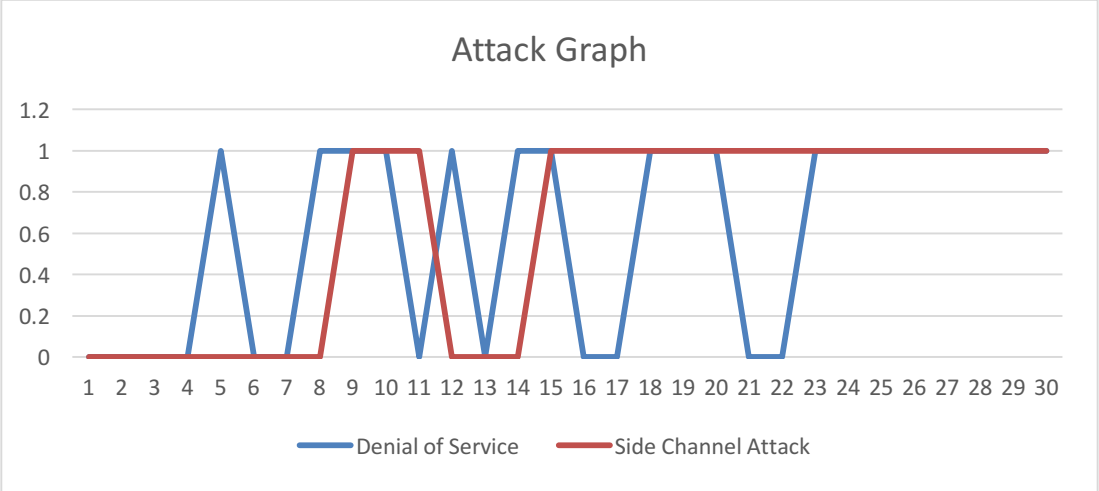

Fig 5.1(a) Attack graph for a single virtual machine

The following figure 5.1(b) shows the reward rates for each attacks simulated separately on a single virtual machine. The initial reward rate is assumed to be 300, it reduces when an improper decision or when an attack happened and increases when the system is successfully able to defend the attack. Each time the action to be taken is dependent on Boltzmann selection, which gives the probabilities for the action to be taken. The reward rates have been rounded off to the nearest integer to have a smooth graph.

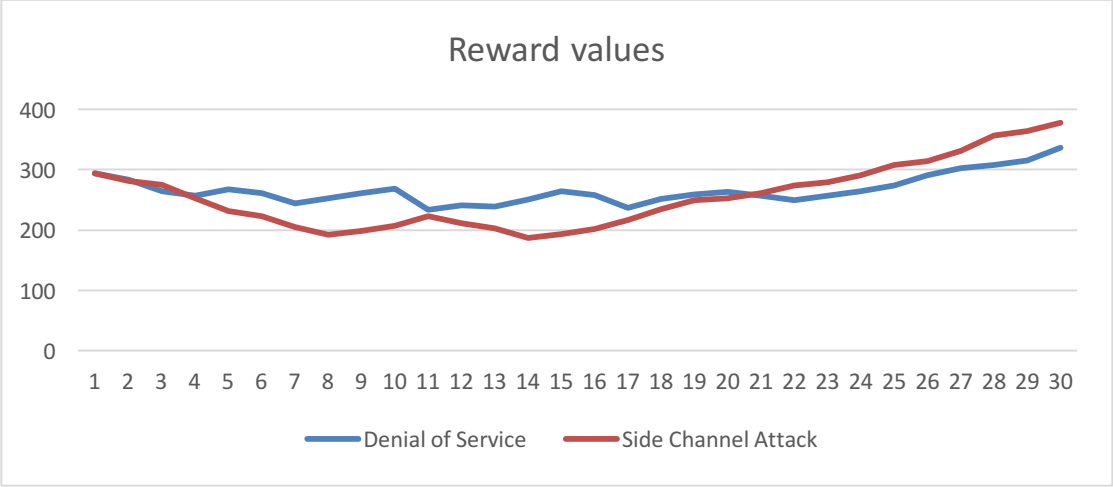

Fig 5.1(b) Reward values graph for a single virtual machine

5.2 Combined Attack on a single Virtual Machine

We have made the simulation combining both the attacks (Denial of Service and Side Channel Attacks) using the same initial reward rates. Fig 5.2(a) Attack graph and Fig 5.2(b) Shows the reward rate graph for 50 runs. The figure 5.3 shows that the attacks have been defended at 37th run, and hence there were no more drops in the graph making the system stable. The X-axis shows the number of runs (Discrete time steps) and the Y-axis shows for the attack graph shows 0 if the system has been attacked or 1 if its been defended whereas the Y-axis on the rewards graph shows the reward values for each run.

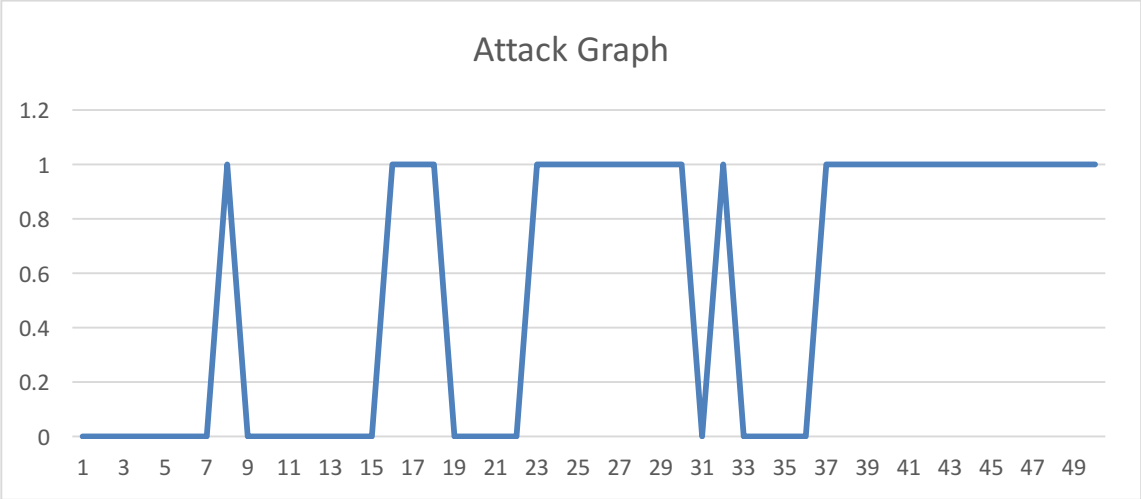

Fig 5.2(a) Attack graph for single virtual machine with combined attacks

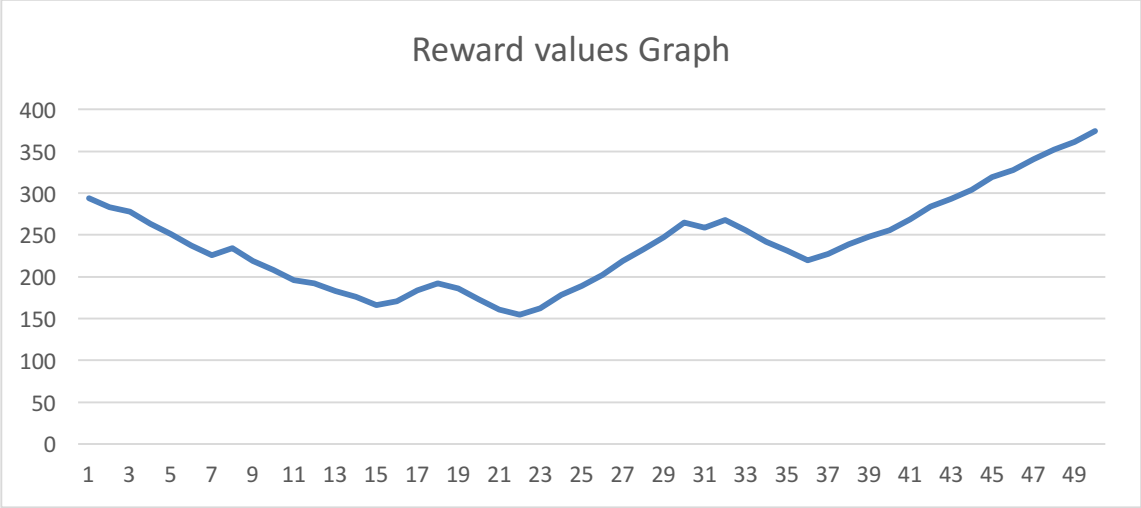

Fig 5.2(b) Reward values graph for single virtual machine with combined attacks

5.3 Combined Attack on Multiple Virtual Machines

Combination of the attacks have been simulated where the attackers try to attack the system simultaneously on ten virtual machines. Fig 5.3(a) shows the attack graph for simultaneous attacks. Since the attacks happen on multiple virtual machines, the system comes to a stable point quicker than expected (comparing with single virtual machine) as in each run there would be ten attacks happening at the same time, which gives the system to learn more quickly. Virtual machine 6 has been subjected to continuous attacks, by which even after reconfiguring it several times, the configuration score has dropped over the threshold set by the administrator, which led to the shut down of the malicious client virtual machine. The graph going down below 0 indicates that the virtual machine has been shut down. Fig 5.3(b) shows the reward values for combined attacks on multiple virtual machines.

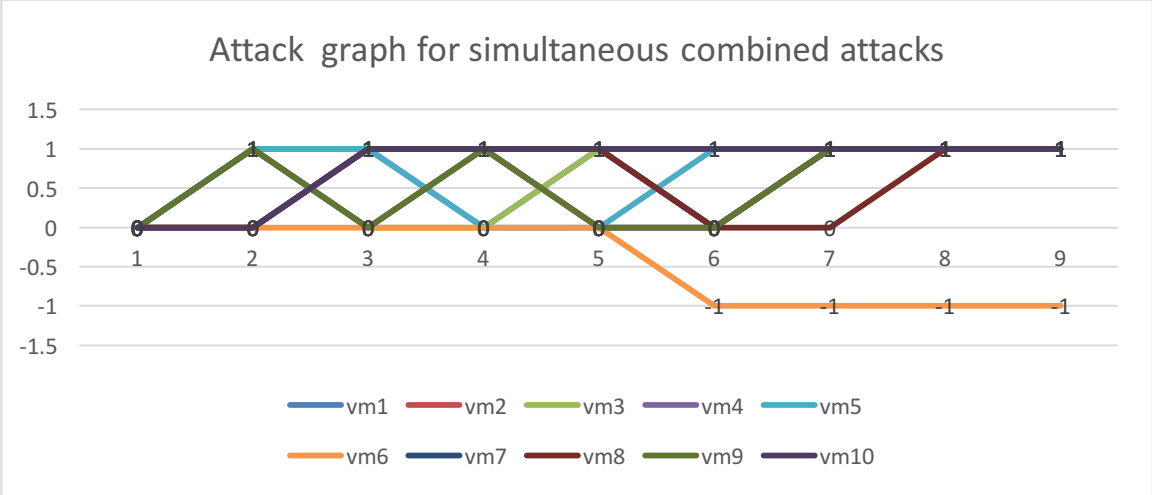

Fig 5.3(a) Attack graph for simultaneous combined attacks

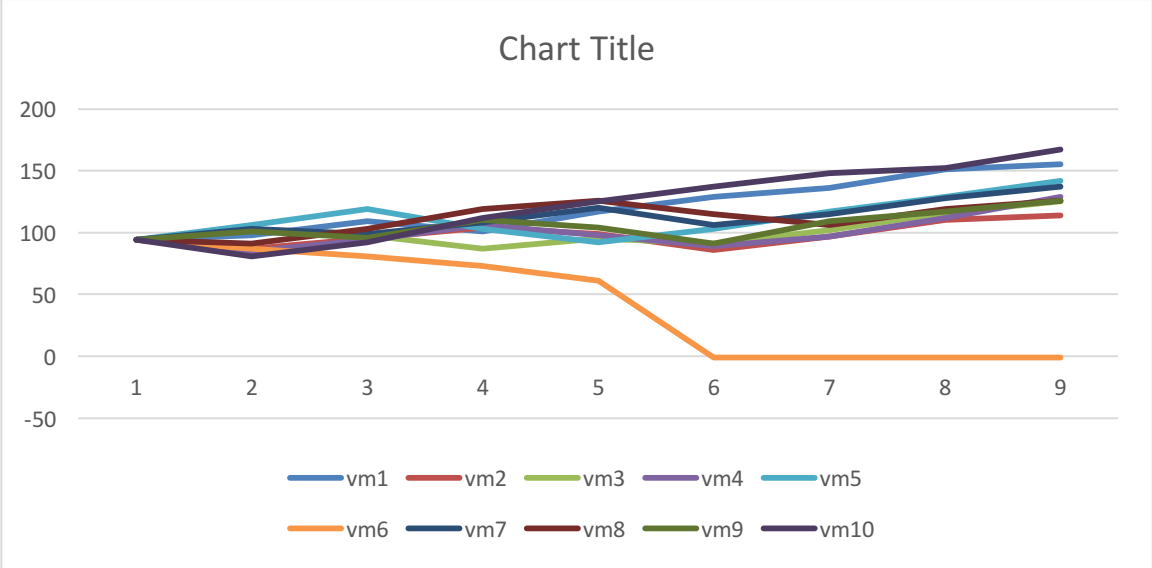

Fig 5.3(b) Reward scores graph for simultaneous combined attacks

CHAPTER VI

CONCLUSIONS

Self-service cloud computing reduces the attack surface of the traditional cloud architecture by transferring most of the privileges to the clients Meta domain. However this architecture is not designed to protect the inter virtual machine attacks and clients vm attacks on the administrative domain and hypervisor.

In the proposed architecture (SSC with Virtual Shield), the client side attacks have been mitigated by dynamically configuring the virtual machines based on the type and severity of the attacks performed by the clients.

SSC with virtual shield is a new computing model designed to protect the host virtual machine from various attacks by the guest virtual machines. The proposed new design has the capability to shift the privileges between the system side administrative domain and client side administrative domain. This dynamic configuration of virtual machines reduces the attack surface and makes the cloud more secure.

If the client tries to misuse the cloud infrastructure, the configuration changes according to the information present in the configuration subsystem. The simulation and results section explains the configuration changes and virtual shield termination.

In the proposed architecture, the reinforcement learning algorithm has been used to make the virtual shield learn from the environment. This algorithm holds good for a minimum number of client virtual machines, since a single virtual shield runs this algorithm to calculate the rewards. The overhead on the virtual shield increases to handle multiple clients and multiple virtual machines. Moreover, if the virtual shield fails, there is no way to protect the entire system from the attacks of the client. The virtual shield can fail because of hardware problem or may be due to the extra over head in handling multiple virtual machines.

The proposed architecture can be enhanced by removing the single point of failure by having a backup virtual shield called a Stand-by Virtual Shield which performs backup tasks by snapshotting. Snapshotting is the process of identifying the virtual machines state and securely storing the states in external devices. If the active Virtual Shield fails, the stand-by virtual shield can be made active. Multiple clients can be simultaneously handled by introducing the concept of multiple VM clusters in the virtual shield. Multiple virtual machines are associated with single virtual shield. This works by replacing the reinforcement learning algorithm with map and reduce functions running on the cluster of VM's. This removes the overhead on the system and can provide more accurate results by analyzing the system log files for different kind of attacks.

REFERENCES

[1] Shakeel Butt, H.Andres Lager-Cavilla, Abinav Srivastava and Vinod Ganapathy, "Self Service Cloud Computing", *ACM Conference on Computer and Communications Security*, pages 253-264, October 2012.

[2] Nils Gruschka and Mieko Jensen, "Attack Surfaces: A taxonomy for Attacks on Services" *3rd International Conference on Cloud Computing*, pages 276-279, 2010.

[3] Asoke K Talukder, "Analyzing and Reducing the Attack Surface for a Cloud-ready Application" *Indo-US Confeence on Cybersecurity, Cybercrime and Cyberforensics*, August 2009.

[4] Trend Micro White Paper, "Cloud Computing Security", website: http://www.securecloud.com/cloud-content/us/pdfs/business/whitepapers/wp_cloudsecurity-unlock-opportunities.pdf , May 2010

[5] Pratyusa K. Manadhata and Jeannette M. Wing, "A Formal Model for a System's AttackSurface" *Technical Reports HPL-2011-115*, website: http://www.hpl.hp.com/techreports/2011/HPL-2011-115.html, 2011.

[6] Richard S. Sutton and Andrew G. Barto, " *Reinforcement Learning An Introduction*" A Bradford Book, 1988

[7] Shyam Patidar, Dheeraj Rane and Pritesh Jain , "A Survey Paper on Cloud Computing" *Second International Conference on Advanced Computing & Communication Technologies*, pages 394-398, 2012

[8] Flavio Lombardi and Roberto DI pietro ," Secure virtualization for Cloud computing" *Journal of Network and Computer Applications*, volume 34, issue 4, pages 1113-1122, June 2011

[9] Farhan Bashir Shaikh and Sajjad Haider, " Security Threats in Cloud Computing" *6TH International Conference on Internet Technology and Secured Transactions*, pages 214-219, December 2011

[10] S VivinSandar and SudhirShenai , " Ecnomic Denial of Sustainbility in Cloud Services using HHT and XML based DDoS Attacks*" International Journal of Computer Applications*, volume 41, issue no-20, pages 11-16, March 2012.

[11] Guojun Wang, Qin Liu and Jie Wu, " Hierarchical Attribute-Based Encryption for Fine- Grained Access Control in Cloud Storage Services" *17th ACM conference on Computer and communication security*, pages:735-737, 2010.

[12] Balachandra Reddy Kandukuri, Ramakrishna Paturi V and Dr. Atanu Rakshit, "Cloud Security Issues" *IEEE International Conference on Services Computing,* pages: 517-520, 2009.

[13] Tien-Hao Tsai, Yen-Chung Chen, Hsiiu-Chuan Huang, Pei-Ming Huang and Kuo-Sen Chou, "A Practical Chinese Wall Security Model in Cloud Computing" *Network Operations and Management Symposium*, pages:1-4, 2011

[14] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, *CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE)*, Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011.

[15] Using the amazon EC2 console to create an alarm to stop an instance Website: http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/UsingAlarmActions.html

[16] M. T. Thai, Y. Xuan, I. Shin, and T. Znati, "On Detection of Malicious Users Using Group Testing Techniques," *in Proceedings of International Conference on Distributed Computer Systems*, 2008.

[17] R Udendhran " New Framework to Detect and prevent Denial of Service attack in Cloud Computing Environment" *Asian Journal of Computer Science and Information Technology,* 4/12 2014

[18] Yinqian Zhang, Ari Juels, Alina Oprea, Michael K. Reiter "HomeAlone: Co-residency detection in the cloud via side channel analysis" *in Proceedings of the IEEE Symposium on Security and Privacy,* 2011.

[19] Vidhyalakshmi Parthasarathy "Cloud Risk Management" *International Journal of Research in Marketing*, Volume 2, Issue 2, February 2012.

VITA

SAI SRAVAN GUDIPATI

Candidate for the Degree of

Master of Science

Thesis: DEFENSE AGAINST ATTACKS IN SELF-SERVICE CLOUD USING REINFORCEMENT LEARNING

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in July, 2016.

Completed the requirements for the Bachelor of Science in Computer Science at GITAM University, Vizag, Andhra/India in 2013.

Experience:

Software Developer Intern May 2012-Aug 2012
Symbiosys Technologies, Vizag, Andhra/India.