DETECTION OF DENIAL OF SERVICE ATTACK AND SIDE CHANNEL ATTACK IN SELF-SERVICE CLOUD USING GROUP TESTING STRATEGY

By

RAKESH JAYARAM

Bachelor of Engineering in Computer Science

Visvesvaraya Technological University

Bangalore, India

2012

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
December, 2015

DETECTION OF DENIAL OF SERVICE ATTACK AND

SIDE CHANNEL ATTACK IN SELF-SERVICE CLOUD

USING GROUP TESTING STRATEGY

Thesis Approved:

Dr. Johnson P Thomas

Thesis Adviser

Dr. Christopher Crick

Dr. David Cline

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Dr. Johnson P Thomas for his excellent guidance, support, and providing me with inspiring instruction for doing research. His guidance helped me to successfully complete my thesis. Besides my advisor, I would like to thank the thesis committee: Dr. Christopher Crick, Dr. David Cline 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: RAKESH JAYARAM

Date of Degree: DECEMBER, 2015

Title of Study: DETECTION OF DENIAL OF SERVICE ATTACK AND SIDE CHANNEL ATTACK IN SELF-SERVICE CLOUD USING GROUP TESTING STRATEGY

Major Field: COMPUTER SCIENCE

ABSTRACT:

The Self-service cloud computing model splits administrative privileges between a system-wide domain and per-client administrative domains. As user clients have administrative privileges there is a possibility for clients to be malicious. In order to detect these malicious clients we introduce a Virtual shield in the system-wide domain. The Virtual Shield (which is a detection model) is a new computing model designed to detect denial of service attack (with respect to RAM usage), side channel attack (with respect to bandwidth usage) and a combined attack for a large population of clients. Results show that our proposed approach detects these attacks with small false positive/negative error rate and low latency. The Virtual Shield has the capability to handle individual attacks and a combined denial of service and side channel attacks.

Keywords: Self-service cloud computing, Denial of Service, Side Channel attack, Group testing, Virtual shield.

.

TABLE OF CONTENTS

.

Chapter Page

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 MOTIVATION

Lately there have been enormous developments in Cloud Computing. It is anticipated that by 2020, more than a quarter of all applications will be accessible through the cloud [1]. Around 56 percent of endeavors consider the cloud to be a key differentiator while around 58 percent of ventures spend more than 10 percent of their yearly plans on cloud services [1]. It is expected that there will be more than 8.2 billion active cell phones by 2020, each generating around 2.7 gigabytes of traffic every month, with more and more applications migrating to the cloud [1]. A study by the technical support firm FixYa states that the top concerns of cloud storage users are security service and storage limitations [3].

There are various security issues in current cloud infrastructures that depend on the administrative domain, which has privileges to control and monitor the client virtual machine. Moreover they have powers to inspect the contents of the client virtual machine. However, in the Self- service cloud [4], the administrator is not permitted to specifically check the computational code of a client virtual machine. This guarantees privacy of client information and protection against malicious cloud administrators. Clients are protected from malicious cloud administrators which is the fundamental motivation behind the Self-service cloud. However there is a possibility of user-clients to be malicious. As clients are in a virtualized cloud environment, they are not aware of the overall cloud architecture or the security model of the system as their main aim is to utilize cloud resources. Therefore such clients can be attacked by malicious clients either by a denial of service attack or side channel attack and a malicious client may get the access to a client’s virtual machine and cause damage. In order to provide security for client virtual machines we design a method to detect denial of service attack and side channel attack using a Group Testing approach [5]. Our focus is designing a method that will detect the attack at the initial stages and also handle the situation when both attacks happen at the same time.

1.2 PROBLEMS WITH SELF SERVICE CLOUD

Our work focuses on security issues of the self-service cloud [4]. Self-service cloud computing implements an upgraded type of cloud architecture where client virtual machines are monitored and controlled by user clients rather by the cloud administrator. The Self-service cloud architecture splits the administrative powers and provides more power and flexibility to the client to maintain and perform system tasks on its own. The cloud administrator is not allowed to directly check the data or computational code of client virtual machines, thereby ensuring protection of client sensitive data and privacy against malicious cloud administrators. Here clients are protected from malicious cloud administrators which is the main purpose of the self-service cloud, but there is a possibility of user-clients to be malicious.

The virtual machines communicate with one another over the network which opens a way for the guest to visitor attack where one virtual machine tries to attack the other virtual machines. Besides it is hard to keep track of all the virtual machines. In this situation two virtual machines communicate with one another over a network. Virtual machine 1 can get data in regards to virtual machine 2 by sending queries while communicating. Virtual machine 1 might be an intruder or a malicious user. Since it is hard to stay informed about virtual machines it is hard to figure out who the malicious client is and what data has been compromised. The attacker may fake information about a client and can directly get access to the virtual machine. These malicious clients may launch a denial of service attack, perform side channel attacks and obtain sensitive data of other users who share the same hardware and they may be a threat to the entire cloud. Since the cloud utilizes virtualization, it needs to be up-to-date with the patches for all the virtual machines which is hard to oversee.

1.3 DENIAL OF SERVICE ATTACK

A Denial of Service attack [6] 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 services, 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). 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 [8].

Denial of Service attacks misuse the network bandwidth capacity around the Internet 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.

However, a few 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 intend to 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.

1.4 SIDE CHANNEL ATTACK [9]

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 [10]. 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.

1.5 PROPOSED APPROACH

The proposed approach detects Denial of Service attack and Side Channel attack using a group testing strategy [6]. The proposed architecture is based on the Self-service cloud computing model which splits administrative privileges between a system-wide domain and per-client administrative domains. The Trusted Computing Base (TCB) of the cloud infrastructure is split into two parts, a system level TCB and a client level TCB**.** A virtual shield that exists between the host and client Meta domain is designed with a detection model which will predict the probability of the attack by using the group testing algorithm. The group testing algorithm aims to detect suspected clients based on ram usage and bandwidth usage of the victim server. In a large population of clients, identifying malicious clients can be faster when tested in groups rather than one by one. Our detection model embeds multiple virtual servers within each physical back-end server. Then virtual shield will assign each client’s VM in a round-robin fashion to these virtual servers. By periodically monitoring resource and bandwidth usage of the virtual servers, and comparing them with some pre-calculated thresholds, virtual servers can be judged as “safe” or “unsafe”. In our approach, whenever the detection model senses any virtual server to be unsafe it recommends a reconfiguration of suspected client virtual machine, rather than the removal of suspected client from the system. Hence we detect an individual attack or combination of attacks at the initial stage and try to mitigate it before it harms the system.

1.6 OUTLINE OF THE THESIS

The rest of this thesis proposal is organized as follows: Chapter 2 provides a literature review of Self-service cloud computing, Denial of Service attacks and the Side Channel attack in cloud computing. Chapter 3 presents our proposed design of a virtual shield which is designed to detect Denial of Service and Side Channel attack and the decision maker which identifies and confirms the attack. Chapter 4, presents implementation of virtual shield in CloudSim toolkit and simulation results. Finally, Chapter 5 is the conclusion.

CHAPTER II

LITERATURE REVIEW

2.1 SELF-SERVICE CLOUD

Modern cloud infrastructures rely on virtual machine monitors (VMMs) that execute a trusted computing base (TCB) to virtualize the underlying hardware (CPU, memory and I/O) and manages client VMs [4]. In VMMs, the TCB has two sections - the hypervisor and an administrative domain. The hypervisor specifically controls physical hardware and runs at the highest processor privilege level. The administrative domain, hereafter called dom0, is a privileged VM that is used to control and monitor client VMs. Dom0 has privileges to start/stop clients VMs, change clients VM setup, monitor their physical resources usage, and perform I/O for virtualized devices.

Providing dom0 with such privileges prompts two issues:

1. Security and privacy of the client virtual machine: Dom0 has the privileges to examine the clients VMs, e.g., the contents of their CPU registers and memory. This can be misused by attacks against the dom0 software stack and by malicious system administrators. This is a real danger, since dom0 regularly executes a full-fledged operating system with supporting client level utilities.
2. Inflexible control over client virtual machine: Virtualization encourages exclusive services to the clients. It can possibly empower services like migration, check pointing

and VM introspection. However the deployment of these services in the present cloud architecture is under the control of the cloud infrastructure provider. The client virtual machines have no power over the selection of these services. Upon the request of the Client, the virtual machines are designed with these services. The client virtual machines may require distinctive security components for various types of attacks. Hence the present cloud architecture has unyielding control over the client VMs.

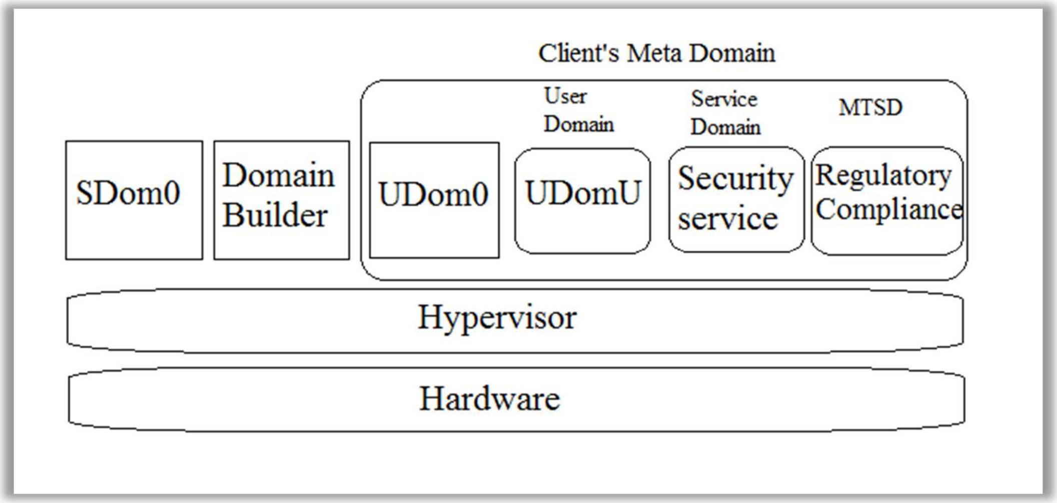


Figure 1: Self Service Cloud Computing model [4]

The Self-Service Cloud [4] divides administrative privileges between a system-wide domain and per-client administrative domains. Every client can manage and perform system tasks on its own Virtual Machine (VM), hence providing adaptability. The system-wide administrative domain cannot assess the code or information of client Virtual Machine, thereby guaranteeing security and protection. The Self-Service Cloud also permits cloud providers and clients to use commonly trusted services that can check administrative consistence while respecting client privacy.

SSC presents a novel privilege model that diminishes the power of the administrative domain and gives clients more adaptable control over their own VMs. SSC's[18] privilege model divides the responsibilities into a system-wide administrative domain (Sdom0) and per-user administrative domains (Udom0s) as shown in figure1.

SDom0 (System side administrative domain) is the system-wide administrative domain in SSC. Sdom0 holds the power to start/stop Udom0 domains upon request by clients, and to run drivers for virtualized devices. Sdom0 manages resources, including scheduling time-slices and I/O quotas. SSC's privilege model disallows Sdom0 from inspecting the state of the client's domains, along these lines guaranteeing the security and protection of client VMs.

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

<u>Client Meta Domain</u>

UDom0: is the client side per user administrative domain that can monitor and control the set of VMs of a specific client. It has the privileges to perform system services on the client virtual machines.

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

SDs (Service Domains): SDs permits clients to perform privileged system tasks on their VMs. SDs gives clients more adaptable control over their VMs. Clients uses SDs to implement services such as memory introspection to verify VM integrity and intrusion detection.

MTSDs (Mutually trusted service domains): MTSDs execute privileged services that check regulatory compliance in a manner that is mutually agreed upon between the cloud provider and the client. MTSD balance the tension between the cloud provider's need to retain control and the client's security and privacy objectives.

2.2 DENIAL OF SERVICE

A denial of service is an attempt to make a server or network resource unavailable to its intended users. Denial of Service attacks typically target sites or services hosted on high-profile web servers such as banks, credit card payment gateways, and even root name servers, business attacks and website attacks. Denial of Service attacks generally target computer networks, but recently it is also used in reference to CPU resource management.

One common method of attack involves saturating the target machine with external communications requests, so that it cannot respond or responds slowly to legitimate traffic. Such attacks usually lead to a server overload. Denial of Service attacks are implemented by either forcing the targeted computer to reset, or consuming its resources so that it can no longer provide its intended service or obstructing the communication media between the intended users and the victim.

Denial of Service attack have lately moved from network level to application level, by consuming more server resource leading to application Denial of Service attack. By abusing the flaws in application configuration and implementation, application Denial of Service attack can bypass general detection techniques used in traditional Denial of Service attack. These attacks usually don't bring about congestion at the network level, therefore bypassing network-based defense techniques.

Malicious traffic can be classified into two types:

- High inter-arrival rate.
- Utilizing more cloud resources.

Existing methods for Denial of Service Detection:

Whenever the cloud provider receives a message that indicates the server may be under a Denial of Service attack, that server is subject to initially passing a set of one or more challenges. If the set of challenges are passed the future resource request may be processed, else the resource request may be dropped. The work flow is as follows [13]:

- Receive the message that predicts a server may be under a Denial of Service attack. That server is subjected to pass at least one or more set of challenges. Cloud administrator provides a valid pass ID to all the clients who have passed the set of challenges.
- If the server receives a resource request from the client, request is checked for pass ID. If valid, client is granted requested resources.
- If the pass ID is not valid, client is presented with another set of challenges. If client doesn't pass the challenges then that indicates it is part of the Denial of Service attack and the request is blocked.
- If client passes the challenge, set of pass ID and request is resubmitted.

An example of one such challenge is CAPTCHA-based defenses, where the cloud provider would pick a word, and produces a distorted and noisy image of that word. This approach introduces service delays for legitimate clients and are also requires human interaction services [11].

2.3 SIDE CHANNEL ATTACK

There is always a co-residency in channels posing risk due to virtualization in the cloud. Recent research has shown how the Side Channel in shared hardware may enable attackers to access sensitive data across virtual machines. To avoid this problem cloud providers will give isolated resources to selected clients. But still the problem persists until the virtual machines are physically separated.

Using Amazon EC2 service as a case study [12], it is possible to depict the internal cloud infrastructure and identify where a particular target VM is likely to reside, and then instantiate new VMs until one is placed co-resident with the target VM. After the successfully placement of the instantiated VM next to the targeted VM, the confidential information is extracted from the targeted VM through Side Channel attack.

Stage 1: Depict the EC2 service to understand where potential targets are located in the cloud. It begins with the hypothesis that different availability zones (EC2 is divided into 3 availability zones i.e. zone1, zone2, zone3) are likely to correspond to different internal IP address ranges and the same may be true for instance types (EC2 is divided into 5 instance types i.e. m1.small, c1.medium, m1.large, m1.xlarge and c1.xlarge) as well. Thus, depicting the use of the EC2 internal address space allows a malicious user to determine which IP addresses correspond to which creation parameters.

Stage 2: Identify where a target VM is likely to reside. Network based co-residence check is performed.

- Matching Dom0 IP address: An instance's network traffic's first hop is the Dom0 privileged VM. An instance owner can determine its Dom0 IP from the first hop on any route out from the instance. One can determine an uncontrolled instance's Dom0 IP by performing a TCP SYN traceroute to it (on some open port) from another instance.

- Numerically close internal IP addresses: Uses the manner in which internal IP addresses appear to be assigned by EC2. The same Dom0 IP will be shared by instances with a contiguous sequence of internal IP addresses.

Stage 3: Instantiate new VMs until one is placed co–resident with the target.

The attacker lists a set of potential target victims. The malicious user then guesses which of these targets belong to a particular availability zone and are of a particular instance type using the depicted cloud from Stage 1. Then, over some period of time the malicious user engages in instance flooding i.e. running as many instances in parallel as possible in the target zone and of the target type. Each probe instance checks if it is co-resident with any of the targets (stage 2). If not the instance is quickly terminated.

Stage 4: Mount cross VM Side Channel attack from a target VM on the same physical machine.

An attacking instance can measure the utilization of CPU caches on its physical machine. These measurements can be used to estimate the current load of the machine; a high load indicates activity on co-resident instances .The measurement of delay generated by a target cache miss may enable attackers to determine the occurrence and frequency of cache misses, which helps to predict confidential data.

Recent research has demonstrated how hostile VMs can potentially extract sensitive data, such as passwords and cryptographic keys, from other VMs resident on the same physical machine by using memory caches as Side Channel [9].

Existing method for Side Channel Detection:

A straightforward way to avoid side channel placement is by placing clients VMs on machines that can only be populated by VMs from only their accounts (private cloud).

1) Virtual Firewall Appliance [14]: A firewall is a set of related programs that protects the resources of users from other networks and intruders or adversaries. A virtual firewall is implemented in the back end server of the cloud. Intruders identify the targeted VM in the cloud infrastructure and then instantiate a new VM co-resident with the targeted VM to extract confidential information. The detection model detects this placement in the side channel attack by implementing a virtual firewall in the cloud server.
2) Fusion –Based Intrusions Detection System [15]: Multi-sensor data fusion in the cloud infrastructure is based on the concept that data received from multiple intrusion detection sensors can enhance the quality of the resulting information. Two or more intrusion detection sensor nodes in the network serve as collecting and analysis point for the data. intrusion detection data fusion model is explained in 3 levels:
    - Level 1 fusion: collection of the observed data is represented as object base. This object base is further analyzed by level 2 and level 3.
    - Level 2 a fusion based intrusion detection system indicates the existence of an intrusion.
    - Level 3 intrusion detection system presents an analysis of the threat of the current situation.

Existing methods needs separate sensor node (hardware) implementation. The data collected from Intrusion Detection System (IDS) sensor node must be transmitted across the network.

Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent malicious user from masking their activities by altering the transmitted sensor data. Confidentiality is required because the transmitted sensor data could be valuable. If the number of Intrusion Detection System sensor nodes are many, then it might create a potential bottleneck in a centralized architecture. However in our approach we use a Group testing algorithm wherein we group the suspected clients based on threshold based anomaly detection. We detect the placement technique of side channel attack, thus it may overcome the limitations of the current model.

CHAPTER III

PROPOSED ARCHITECTURE

3.1 INTRODUCTION

We propose an architecture that is based on the self-service cloud model which uses the Group testing approach to detect a Denial of Service and Side Channel attack. This architecture can be easily extended to detect other types of attacks.

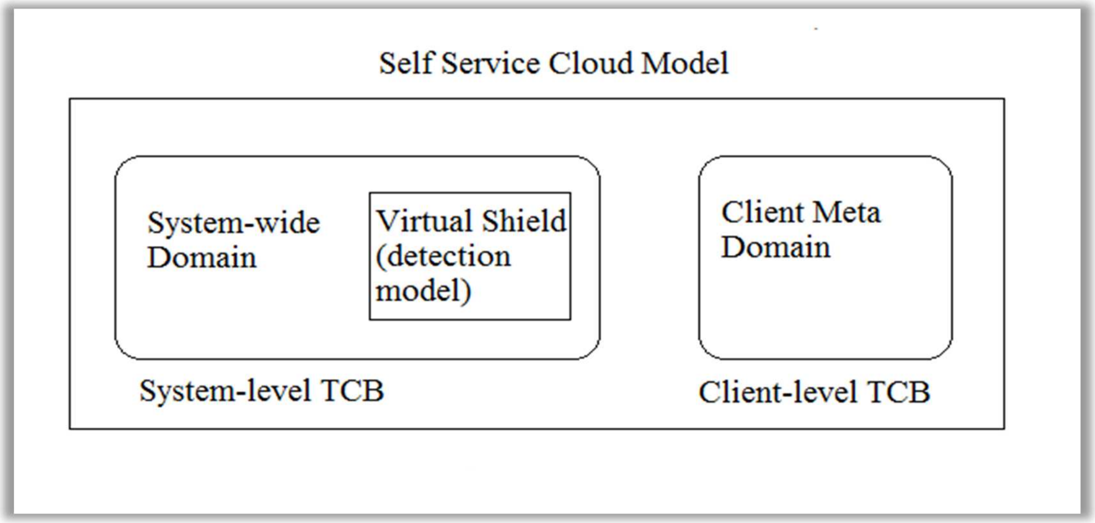


Figure 2: Proposed self-service architecture

The proposed self-service cloud model**,** splits administrative privileges between System-wide domain and per-client administrative domains. System-wide domain (system level Trusted Computing Base) is controlled by a cloud provider which consists of hypervisor, Domain Builder (domB), BIOS and Virtual shield (VS). Per-client administrative domains (client level Trusted

Computing Base) are controlled by clients which consists of client's User Domain0 (Udom0), Service Domain (SDs), and Mutually Trusted Service Domain (MTSDs). The Virtual shield (VS) plays a vital role in this architecture, as it detects attacks. The VS predicts the probability of the attack and triggers the reinforcement learning (RL) technique. The RL technique determines an appropriate action or reconfiguration to the system in order to mitigate the effect of the attacks. In our approach, when the virtual shield senses a malicious activity in the cloud, the detection model identifies the probability of an attack.

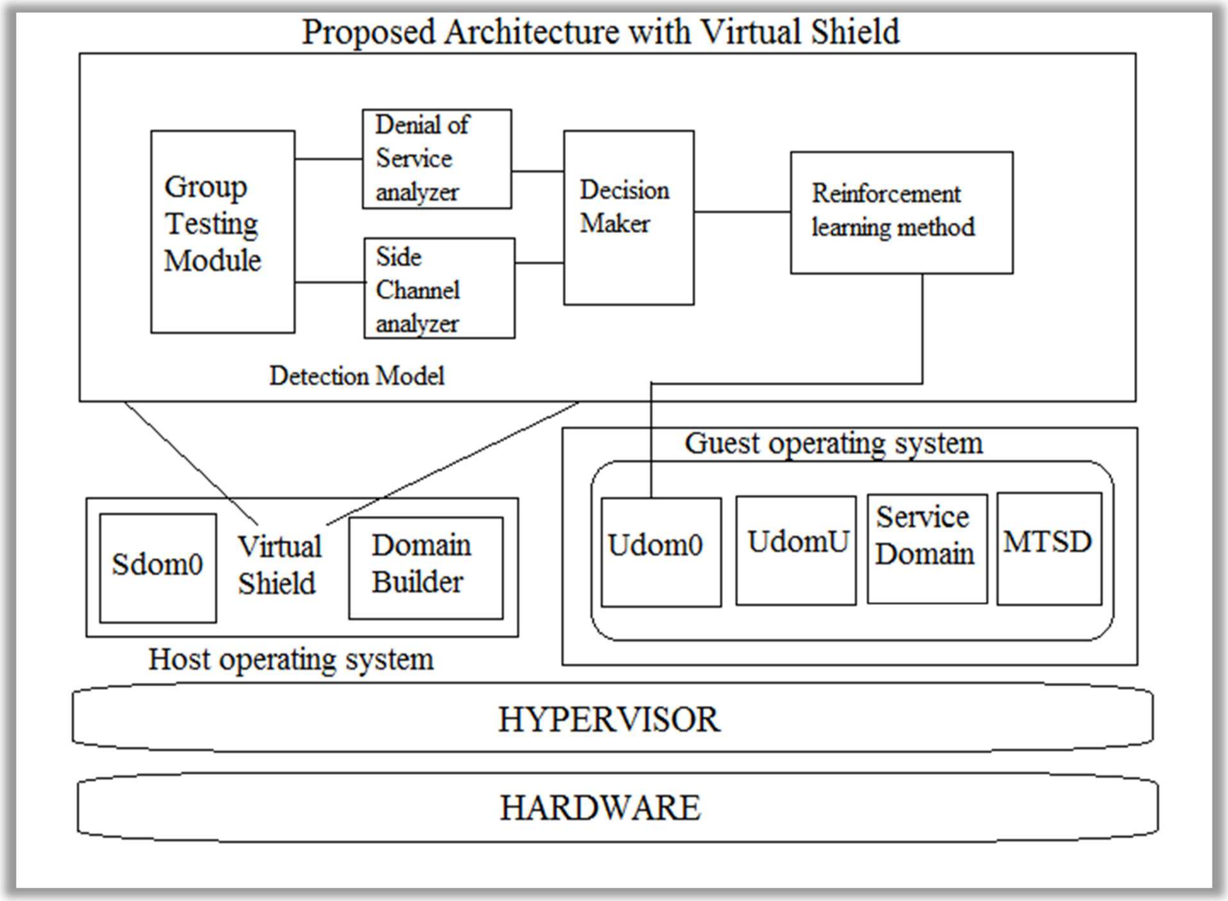


Figure 3: Proposed cloud architecture with Virtual Shield detection model.

SDom0: is a system side administrative domain. Which controls the start and stop of the client virtual machines.SDom0 has no privilege to view the state of the client's virtual machines, i.e. the contents of virtual CPU, virtual memory.

DomB: the domain builder builds the client side Meta domain. Once the client sends a request to SDom0 to build the virtual machines, these request parameters are sent to the domain builder and virtual shield. Domain builder uses these parameters to build the client side Meta domain.

Udom0 (User dom0): is a per-user administrative domain, which monitors and controls the set of Virtual machines (VM) of a particular client. When a client attempts to start a VM, it is assigned its own Udom0 domain. This domain creates the user VMs, which perform the actual work for the client (UdomUs). UDom0 is a privileged client domain that starts and manages the UDomU and has special privileges to access the hardware. UDom0 has drivers for hardware, and it provides virtual disks and network access for UdomU.

Service domains (SDs) are special-purpose user domains, which can perform privileged system services on UdomUs (memory introspection to verify VM integrity, intrusion detection).

MTSDs (Mutually-trusted service domains) executes privileged services that check regulatory compliance in a manner that is mutually agreed between the cloud provider and the client. It is also configured to update the Virtual Shield (VS) regularly with status information, complying with the agreement between cloud provider and the client. The information about detection parameters such as resource utilization, bandwidth utilization and instance count of the clients are provided by MTSDs to virtual shield.

Hypervisor is a program that allows multiple virtual machines to share a single physical hardware. It controls the cloud provider's processor and resources by allocating what is requested by each virtual machine while making sure that they cannot disrupt each other.

Virtual Shield (VS) is designed with the detection model which uses the Group testing algorithm. MTSD provides information about the different parameters used to detect the attack to the virtual shield.

In the self-service cloud, MTSD acts as the regulatory compliance between client virtual machines and cloud providers. Once the client virtual machines are identified to be misusing the cloud infrastructure for malicious activities, their virtual machines are shut down. However, in our proposed architecture, information from the MTSD is used by the virtual shield to detect the attack and trigger the RL technique to take appropriate actions.

Group testing (GT) [5] identifies malicious clients faster when tested in groups rather one by one in a large population of clients. GT detections are merely based on the status of resources and bandwidth usage of the victim's servers. Thus it overcomes the limitations of other approaches as it does not require any signature-based authentications or human interaction. GT approach achieves high detection performance in terms of short detection latency and low false positive/negative rate (incorrect prediction of attack).

In our Group Testing detection model, each physical back-end server embeds multiple virtual servers. The Virtual Shield in a round-robin fashion assigns each client VM's to these virtual servers (so that each virtual server has equal number of clients to serve). By periodically monitoring resource and bandwidth usage of the virtual servers and comparing them with pre-calculated thresholds, the virtual servers can be judged as "safe" or "unsafe".

- If all the virtual servers are judged as safe then, we consider the system as safe and run it in normal mode (see later in the Normal mode and Danger mode section).
- If one or more virtual servers are judged as unsafe then, we consider the system as unsafe and run in danger mode (see later in the Normal mode and Danger mode section), where the detection algorithm detects the probability of an attack.

3.2 DETECTION MODEL (Virtual Shield)

The Virtual Shield comprise of the detection model which consists of the Group testing module, Denial of Service analyzer, Side Channel analyzer and Decision maker.

We assume that each host (Sdom0) has one back-end physical server that works as an independent testing domain, divided into a number of virtual servers. Each of these virtual server's resource utilization such as RAM and bandwidth is compared against the threshold .For example if one or more clients uses more resources in a particular virtual server than that virtual server is marked as unsafe.

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 virtual server by inspecting the resource usage. ERT is monitored to detect initial malicious activities during testing.

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 world-wide, 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)

There are two ways in which an attacker (malicious client) can flood using probe instances. 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 traceroute and inspecting the last hop. (In TCP SYN traceroute 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 traceroute can generate more number of ICMP and UDP packets in the network.

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.

3.2.1 GROUP TESTING MODULE

The group testing module consists of multiple testing rounds (based on the number of suspected clients) and each round has four stages:

1. In the first round, the Virtual Shield generates a matrix M of clients and virtual servers (by assigning a client to a virtual server in a round robin fashion) and in later testing rounds it keeps updating the matrix M of virtual server and clients. The matrix is generated in such a way that each virtual server serves almost equal number of client VMs. In matrix M each row represents the virtual server and each column represents each client. If M(i,j)=1 represents client j is mapped to virtual server i, if M(i,j) =0 , represents that client j is not mapped to virtual server i.
2. The Virtual Shield distributes client requests to virtual servers based on the matrix M.
3. All servers are periodically monitored for their service resource usage, bandwidth usage and instance count of the clients. Specifically the average response time of each virtual server and Instance Count (IC) of clients are recorded by the Virtual Shield and then compared with pre-calculated thresholds. All virtual servers are associated with safe or unsafe outcomes accordingly.
4. The outcomes identify legitimate and potentially malicious clients. By following the detection algorithms, the probability of the attack can be identified within several testing rounds.

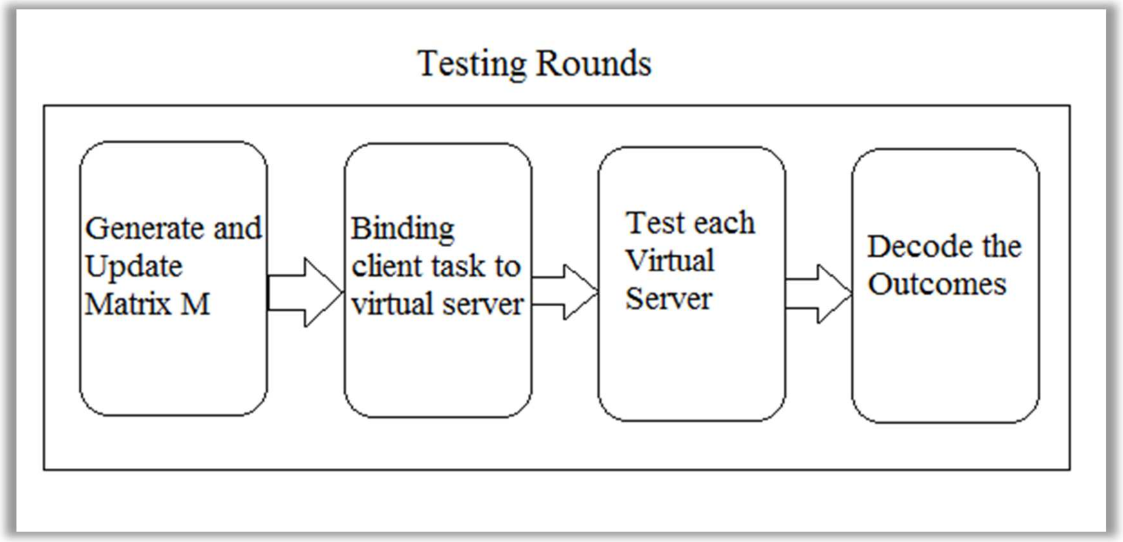

Testing Rounds
Generate and Update Matrix M
Binding client task to virtual server
Test each Virtual Server
Decode the Outcomes

Figure 4: Testing round in group testing module

In the group testing module, if the Estimated Response Time (ERT) or Instance Count (IC) of clients of any virtual server exceeds some pre-calculated threshold, the whole back-end server will transfer from NORMAL mode to DANGER mode and execute the detection scheme. Whenever the Average Response Time (ART) or Instance Count (IC) of client of each virtual server falls below the threshold then the physical server returns to NORMAL mode.

As shown in Fig. 5, the back-end server cycles between two states, which we refer as NORMAL mode and DANGER mode.

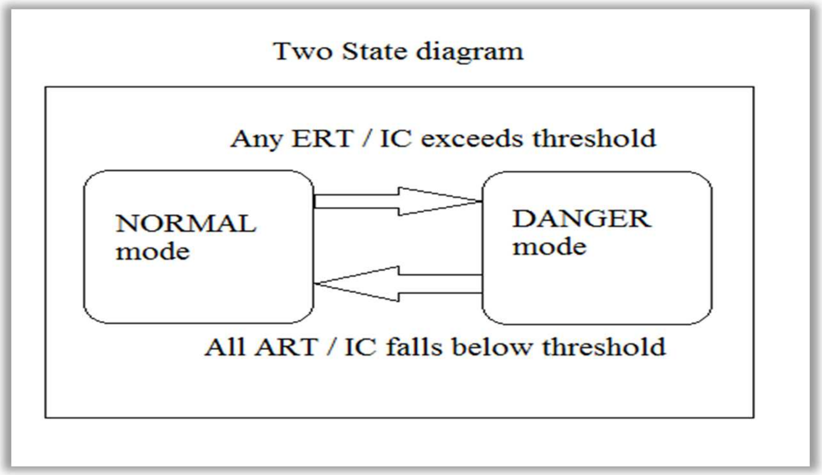

Two State diagram
Any ERT / IC exceeds threshold
NORMAL mode
DANGER mode
All ART / IC falls below threshold

Figure 5: Two-state diagram of the system

The group testing algorithm is presented next:

Notation Used

w = number of clients that can be handled by a virtual server

S = set of suspected clients

k = number of virtual servers in one physical back end server

L = set of clients connecting to safe virtual servers

Q = set of clients using suspected virtual servers

N = total number of clients using the physical server

C = Number of times client misbehaves

3.2.2 GROUP TESTING ALGORITHM

```
for all virtual servers $i$ do
$w \leftarrow S/k$                    //virtual shield evenly assign clients to virtual servers
```

```
 end for                                  //in round robin fashion.
// initial check for resource such as ram and bandwidth utilization in virtual servers.
while $|S|=0$ do                          //Initial testing is done
 $L \leftarrow$ set of clients on safe virtual servers
 $S \leftarrow S / (S \cap L)$                      // $|S \cap L|$ clients are identified as legitimate.
 Randomly reassign $|S|$ suspect clients to $K$ servers, and do not move legitimate clients.
// check for resources such as ram and bandwidth utilization in virtual servers.
 for all servers $i$ under attack do
 $Q \leftarrow$ set of clients on $i$
 if $|Q \cap S| <= 1$ then
 $S \leftarrow S \setminus Q$                 // the clients in $Q \cap S$ are attacker and added into the black-list.
switch ($C$)
case 1 suspect list $\leftarrow S \setminus Q$
          break
case 2 black list $\leftarrow S \setminus$ suspect list
          break
case 3 malicious list $\leftarrow S \setminus$ black list
          break
 end if
 end for
 end while
```

<u>Algorithm Description</u>. The basic idea of the algorithm is, given a suspected clients set *S* with initial size *N*, evenly assign them to *K* virtual servers. For further rounds assign suspected clients to the *K* servers randomly. For servers with a positive outcome, it initializes all the detections entities lists such as list of previous suspects, malicious clients, and blacklisted clients and activates the appropriate attack analyzer. Clients active on this server which are not included in the set of identified legitimate clients (i.e. suspected clients), will still be identified as suspected clients and later will be blacklisted based on value *C*. The attack analyzer in turn activates the group testing algorithm and the algorithm runs till all the suspected clients are either found malicious or safe.

If any virtual server is tested as positive (not normal), then all active clients using that virtual server is cross checked with the set of legitimate clients. Active clients that do not belong to set of legitimate clients, will be marked as suspected clients. In further rounds these suspected clients are randomly assigned to virtual servers and the testing process continues. If there is just one suspected client in a virtual server (positive), then that client is probably an attacker. According to our detection model we assume an attack to be continuous and prominent enough to decrease the efficiency of the system.

<u>Legitimate Profile</u>: The legitimate profile records the distribution of the Instance Count (IC) of clients and ART on a virtual server by receiving only legitimate traffic. Malicious requests will cause problems to the victim server machines, whose ART and Instance Count will be higher than that of normal cases. Therefore, ART and Instance Count of client serve as an indicator of

the application resource usage and instances probing respectively. However, the resource usage and instance creation varies for different time intervals (peak/non-peak time) due to the change of client quantity. We also investigate the ART and Instance Count distributions with respect to each possible number of clients, assuming that there are at most N clients. A sample construction of this profile is: The distributions of ART and Instance Count in legitimate traffic at different time intervals for several times are obtained after the system is established. Legitimate traffic can be achieved by ruling out the influences of potential attacks.

Normal mode and Danger mode: The back-end server provides normal service to the clients, this is referred as NORMAL mode. The Detection model monitors the ERT (Estimated Response Time) and Instance Count of client.

We refer to the recorded distribution of the average response time (ART) on a virtual server by receiving only legitimate traffic. Average Response Time works as an indicator of the application resource usage.

$\mu + 3\sigma$ (μ is the mean and 3σ is the 3rd standard deviation) is the range of value between the mean and 3rd standard deviation Almost 97% of the legitimate values lies under this range.

Since the ART normal distribution provides an approximate value of ART, we use a simplified threshold: If any virtual server has an ERT > $\mu + 3\sigma$, the back-end server is probably undergoing a Denial of Service attack and thus transfers to DANGER mode for detection.

We refer to the recorded distribution of the Instance Count of the client on a virtual server by receiving only legitimate traffic. Instance count (IC) of the client works as an indicator of the instance probing.

We also consider the Instance Count Normal Distribution of clients to predict malicious activity. If any virtual server has an Instance Count > $\mu + 3\sigma$, the back-end server is probably undergoing a Side Channel attack and transfers to DANGER mode for detection.

The Group testing algorithm runs until all the suspected clients are marked either as legitimate or malicious. It outputs the testing results to the attack Denial of Service analyzer and Side Channel analyzer.

As the client's resource usage changes with the time, the initial recorded distribution of ART and Instance Count is updated regularly based on resource usage. Based on the updated distribution the new threshold is calculated and updated regularly.

Testing round and Testing period**:** The Number of Testing rounds is equal to the number of initial suspected clients (if there are 5 initial suspected clients then the number of testing rounds will be 5).In some cases when attackers perform both Denial of Service and Side Channel attacks simultaneous, the detection model requires fewer testing rounds than expected to detect malicious clients, as we have a separate attack analyzer. In some rare cases during testing the resource usage spikes can be wrongly predicted as malicious activity and this will take more testing rounds than expected to detect malicious clients. Each testing round has a specific number of testing periods (the number of periods must be long enough to collect sufficient values to differentiate between legitimate and non-legitimate values). In each period the detection parameters are checked and these values are compared against the threshold. The output of the testing is send to the attack Denial of Service analyzer and Side Channel analyzer.

Denial of Service analyzer

The Average Response Time values of the virtual servers are analyzed in the Denial of Service analyzer

1. The Denial of Service analyzer first checks the recorded ART distribution profile of virtual servers that was generated by receiving only legitimate traffic in the NORMAL mode. The values of the distribution i.e. μ (mean) and σ (deviation) are used to compare with the obtained average response time.
2. In the current testing period, if there is no violation, that is, ART $\geq \mu + 3\sigma$ the virtual server is listed as negative or NORMAL for this testing round.
3. In the current testing period, if ART $\geq \mu + 3\sigma$ occurs, this virtual server is in danger and may be under attack. In this case the Denial of Service analyzer waits till the end of this round (each testing round has a specific number of testing periods) to get all the ART values to make a decision.
4. If the ratio of testing round in "danger" (with ART $\geq \mu + 3\sigma$), to the total number of testing rounds exceeds threshold, then this virtual server will be labeled as positive. Initial recorded distribution of ART is updated regularly based on client's RAM usage. Based on the updated distribution the new threshold is calculated and updated regularly. The new threshold value $\mu + 3\sigma$ specifies that almost 97% of the legitimate values lie under this threshold.
5. For other cases, the virtual server will have a negative outcome.

Side Channel analyzer

The Instance Count values of virtual servers are analyzed in the Side Channel analyzer

1. The Side Channel analyzer first checks the recorded Instance Count distribution profile of virtual servers that was generated by receiving only legitimate traffic in the NORMAL mode. The values of the distribution μ (mean) and σ (deviation) are used to compare with the obtained Instance Count of the client.
2. In the current testing period, if Instance Count $\geq \mu + 3\sigma$ is not true, the virtual server a gets negative label for this testing round;
3. In the current testing period, if Instance Count $\geq \mu + 3\sigma$ occurs, this virtual server is in danger and may be under attack. In this case, the side channel analyzer will wait till the end of this round (where each round can have several testing periods) to get Instance Count values to make a decision.
4. If the ratio of testing round in "danger" (with Instance Count $\geq \mu + 3\sigma$), to the total number testing rounds, exceeds threshold, then this virtual server will be labeled as positive. Initial recorded distribution of Instance Count is updated regularly based on client's bandwidth usage. Based on the updated distribution the new threshold is calculated and updated. The new threshold value $\mu + 3\sigma$ specifies almost 97% of the legitimate values lies under this threshold.
5. For the other cases, the virtual server will have a negative outcome.

If all the virtual servers have negative outcomes, this back-end server is diagnosed as healthy and returns to the NORMAL mode.

If one or more virtual servers have positive outcomes, this back-end server is diagnosed as under attack and the probability of the attack is calculated by the ratio of testing rounds in danger to the total number of testing rounds.

3.3 DECISION MAKER

The Decision Maker obtains input from Denial of Service analyzer and Side Channel analyzer, and performs one of the following:

- If the attack probability is less than the threshold, the decision maker sends the Average Response Time (ART) and Instance Count (IC) report periodically to the reinforcement learning method.
- If the attack probability is getting close to the threshold, the Virtual Shield sends a message to notify the reinforcement learning method the Average Response Time (ART) and Instance Count (IC) of suspected clients.
- If the attack probability is equal to the threshold, the virtual shield sends test results about the attack and alerts the cloud provider and the reinforcement learning method about the resource usage and instance count of suspected clients to indicate that an attack might happen.
- If the attack probability crosses the threshold, then the virtual shield detects and confirms the attack, warns the cloud provider and reinforcement learning method about attack detection and also specifies the suspected clients.

CHAPTER IV

SIMULATION AND RESULTS

4.1 IMPLEMENTATION

CloudSim [16] is used as extensible simulation toolkit that enables modeling and simulation of Cloud computing systems and Cloud security provisioning environments. The CloudSim toolkit supports both system and behavior modeling of Cloud system components such as data centers, hosts and resource provisioning policies. It implements generic application provisioning techniques that can be extended with ease and limited effort. CloudSim is a simulation environment to simulate cloud architectures before actual deployment. CloudSim provides java APIs to design the various elements of the cloud computing architecture [17]. The proposed architecture is designed and simulated in the CloudSim environment through implementing the following entities which satisfies the requirement of the proposed model. In our simulations when an attack is injected, it occurs continuously until the detection model detects it. Once the detection model detects the attack, the attack is stopped.

Different entities in the Proposed Architecture include:

1. SELF SERVICE CLOUD SYSTEM
2. DETECTION MODEL (VIRTUAL SHIELD)
3. GROUP TESTING ALOGORITHM
4. DENIAL OF SERVICE ATTACK AND SIDE CHANNEL ANALYSER

5. CLOUD CONFIGURATION SYSTEM

In the following section we discuss these subsections in detail.

4.1.1 SELF SERVICE CLOUD SYSTEM:

The Self Service cloud system is the main system where the entire architecture is initialized. The Administrative domain (Broker) in the self-service cloud is responsible for receiving the client's requirements and requesting the Datacenter (Hypervisor) to allocate resources to the clients. During the initialization process, entities such as datacenter, host, virtual server, virtual shield, are all initialized and activated.

Simulations with different number of virtual server, clients and attackers are created. Initial binding of clients to virtual servers are handled by the self-service cloud system. The access to information about the current client's configuration is sent to the Virtual shield. The Self-Service cloud (Cloud Provider) has privileges to start/stop the client's virtual machines.

The client's utilization model in the CloudSim domain is initialized with different attack models and configuration parameters. The Detection model uses the utilization value to detect malicious clients. Based on different simulation scenarios we randomly select clients and inject attacks using parameters such as RAM and Bandwidth.

4.1.2 DETECTION MODEL (VIRTUAL SHIELD SYSTEM):

The Virtual Shield is designed in such a way that it utilizes the information from the utilization models and keeps checking for current ram and bandwidth usage against the predetermined threshold. A utilization model is a CloudSim entity which provides fine-grained control and information over resources utilized by client tasks.

In the initial check if the virtual shield finds that the current usage of either ram or bandwidth is beyond the threshold it initializes all the detections entities lists such as list of previous suspects, malicious clients, blacklisted clients and activates the appropriate attack analyzer. The attack analyzer in turn activates the group testing algorithm and the algorithm runs till all the suspected clients are either found malicious or safe.

The Virtual Shield is responsible on deciding how clients are assigned to different virtual servers and it tracks all the clients based on which virtual server they are currently using. In our model the clients are distributed to the available virtual server in a round robin fashion. If the initial check is positive then the clients on victim virtual servers are redistributed to the next available virtual server.

According to our detection model we assume an attack to be continuous and prominent enough to decrease the efficiency of the system. Low intensity attacks are not considered unless it is harmful for the system.

4.1.3 GROUP TESTING ALGORITHM:

The group testing algorithm aims to detect malicious clients in a large population with the minimum number of tests where each test is applied to a subset of suspected clients, instead of testing them one by one.

The Group testing algorithm in our detection model is designed to detect multiple attacks such as Denial of Service and Side Channel. If the initial check in the Group testing algorithm is positive, it activates either the Denial of Service or Side Channel attack analyzer based on the type of resource usage to detect the malicious client. It marks all the clients that are using the victim virtual sever as suspects and redistributes each of them to the next available virtual server.

In the second stage it keeps track of previous suspected and safe clients and the redistributed information to detect the malicious client and filters out the safe clients.

4.1.4 DENIAL OF SERVICE ATTACK AND SIDE CHANNEL ATTACK ANALYSER:

The Attack analyzer is active until all the suspected clients are classified as either malicious or safe. During the process of group testing, configuration of the client and binding of clients to the virtual server changes based on Denial of Service and Side Channel attack.

Denial of Service and Side Channel attack analyzers keep track of suspected clients with respect to misbehavior in terms of RAM or bandwidth. The attack analyzer detects the attack if it is continuous and strong enough to decrease the efficiency of the system.

4.1.5 CLOUD CONFIGURATION SYSTEM

The Cloud Configuration file defines the properties of the host and virtual server such as computing capacity in terms of millions of instructions per second, image size, memory, number of CPU, and bandwidth allocated. The configuration file holds the information of different configuration parameters for different clients. If the detection model detects an attack, then configurations of the suspected client is sent to the Reinforcement Learning method for appropriate action to mitigate the attack.

Configurations of the Host, Virtual servers and clients include: MIPS, IMAGE SIZE, MEMORY SIZE, CPUS, LENGTH and BAND WIDTH.

- MIPS defines millions 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 server.
- Length defines the size of a client task to be executed in a Cloud Resource in terms of instruction size. For a given client configuration as the number of instruction to be executed increase, the time taken to complete the task increases.
- Bandwidth defines the network bandwidth (number of bits transmitted per second).

4.2 COMMUNICATION PROTOCOL

The communication protocol in Figure 6 explains how each of the ENTITES in the proposed model interact with each other.

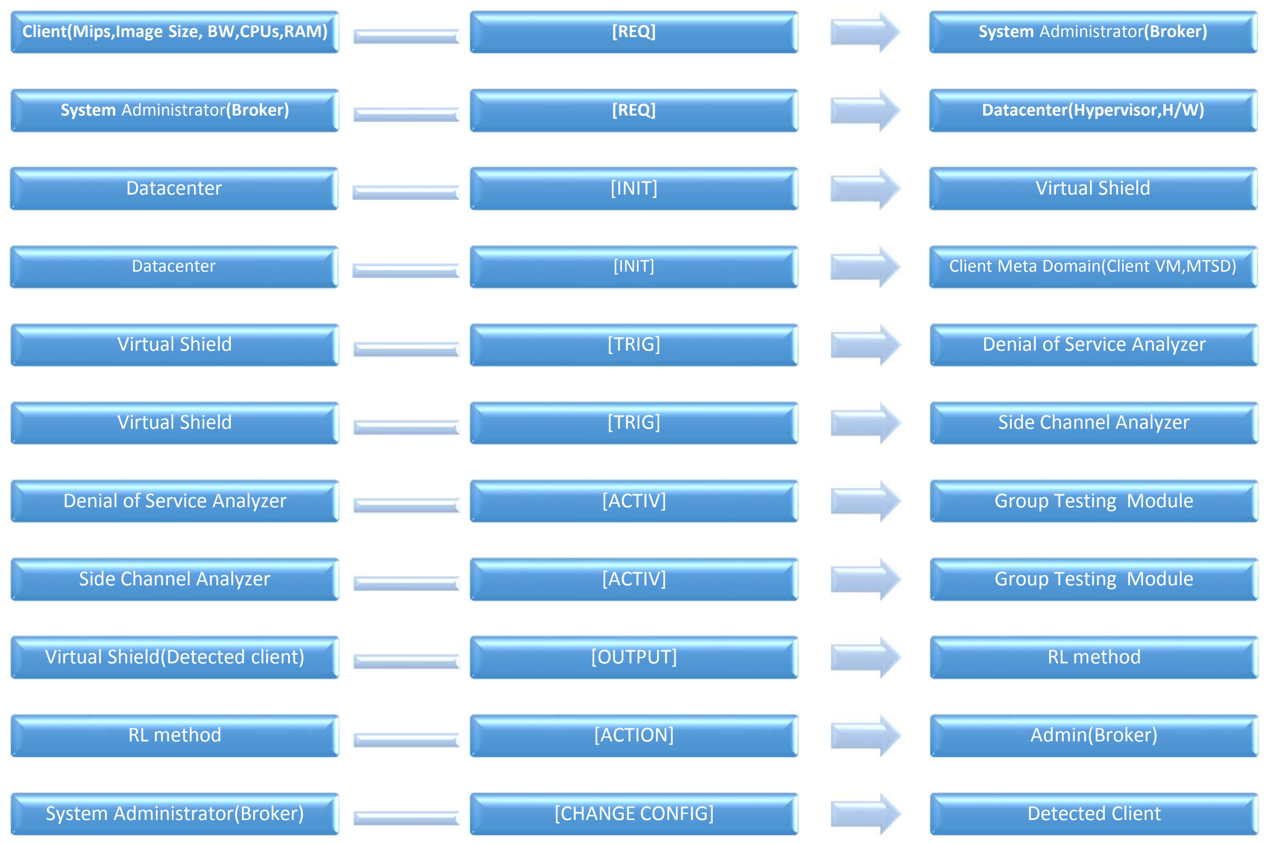


Figure 6 Communication protocol

The client sends the required configuration parameters to the system administrative domain (broker) and requests for the virtual machine. The system administrative domain (broker) requests the datacenter i.e. the hypervisor to provide the requested resources to the client. In this process, the datacenter initializes the virtual shield and the client's meta domain.

If the client tries to misuse ram resources, the virtual shield triggers the Denial of Service attack analyzer. The Denial of Service attack analyzer activates the group testing module and the algorithm runs until all the suspected clients are resolved either as safe or malicious. If the client tries to misuse bandwidth resources, the virtual shield triggers the side channel attack analyzer. The side channel attack analyzer activates the group testing module and algorithm runs until all the suspected clients are resolved either as safe or malicious. The detection result is sent to the RL method for further action. The RL method checks the severity of the attack and updates the detected client's configuration. The cloud system administrator changes the detected client's configuration in the system accordingly.

4.3 RESULTS

The purpose of the simulation using the CloudSim toolkit [17] 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 [16].

We show the efficiency of our detection model by varying the simulation environment settings as follows:

- Increasing number of Virtual Servers *k*;
- Increasing number of malicious clients (attackers) *x*;
- Increasing number of clients *n*.

4.3.1 Increasing number of Virtual Servers *K*:

The number of client's *n* is set to 200 and number of attackers *x* is set to 10 and the number of virtual servers is increased. Simulated Detection model took about 76 to 90 milliseconds to detect each Denial of Service attack and about 70 to 80 milliseconds to detect each Side Channel attack.

From Figure 7a it can be seen that for the given number of clients and attackers the average false positive error rate (predicting the safe client as malicious client) drastically reduces if there are 40 virtual servers or more. The Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage and combined attacks (injecting both attacks simultaneously) all have almost the same average false positive error rate because as the number of virtual servers increase the number of clients using a virtual server decrease. Attackers are exposed as the number of clients per virtual server is minimal.

From Figure 7b we can see that for the given number of clients and attackers the average false negative rate (unable to identify the malicious client) drastically reduces if there are 40 virtual server 40 or more. The Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage has almost the same average false negative error rate. The attackers are exposed as the number of clients per virtual server is minimal. We also observe that the combined attack has comparatively less average false negative rate initially because the attacker is detected by two attack analyzers. Although both the attacks use the same group testing algorithm they have their own separate list of previous suspects and malicious clients.

Combined attack is injected in the system in 2 ways

- Injecting both Denial of Service and Side Channel attack simultaneously in single client.
- Injecting both Denial of Service and Side Channel attack simultaneously in adjacent clients.

From Figure 7c we can see that for the given number of clients and attackers the average latency of detection of malicious client is minimum when the number of virtual server is 50 and above We can observe that although the average false positive and negative error rate are at a minimum when the number of virtual servers is 40 and over, the detection latency is high because the order in which the attacks are detected varies. If attack is detected in the order in which it is injected, the average detection latency is small. Using 50 virtual servers, the number of clients on each virtual server is reduced and as and when the attack is launched the detection model detects it quickly.

Simulation Graphs are obtained by running the simulation for 100 times for each value in X-axis and is plotted in Y-axis.

We observe that for the number of clients n =200 with number of attackers x=10, the ideal number of virtual servers is 50. The false positive and negative error rates are minimal and the average latency of detecting malicious client is also low.

4.3.2 Increasing number of clients *n*:

We set the number of virtual servers *k* = 60 and number of attackers x=10, and analyze the efficiency of the detection model by simulating it with increasing number of clients. Simulated Detection model took about 68 to 117 milliseconds to detect each Denial of Service attack and about 60 to 110 milliseconds to detect each Side Channel attack.

From Figure 8a we can see that for the given number of virtual servers and clients the average false positive error rate (predicting the safe client as malicious client) linearly increases for number of clients *n* = 450 and above. Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage and combined attack all have almost the same average false positive error rate. As the number of clients increase the load on each virtual server increase. As the number of clients increase there are higher chances of wrongly predicting a safe client as malicious. It becomes difficult to differentiate between a sudden resource usage spike and an actual attack.

Average False error rate (%) is calculated by running the simulation for 100 times for each value in X-axis and the false error rate percentage is plotted the in Y-axis.

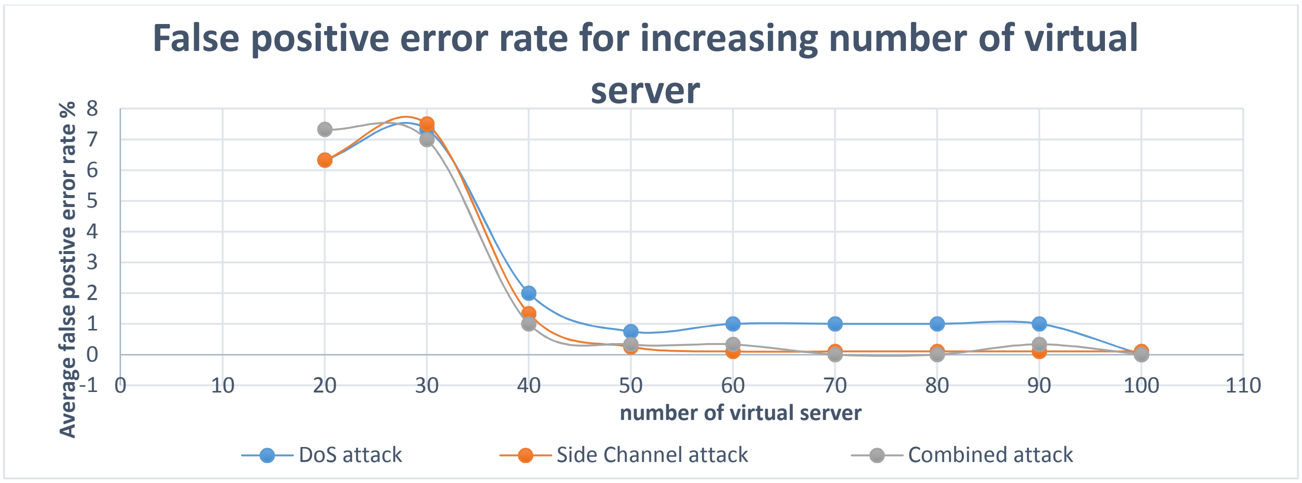
From Figure 8b we can see that for the given number of virtual servers and attackers the average false negative rate (unable to identify the malicious client) drastically increase for number of clients *n*=450 and up. The Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage has almost the same average false negative error rate. We also observe that the combined attack has comparatively less average false negative rate. In combined attack the number of attackers handled by each attack analyzer is fewer compared to individual attack.

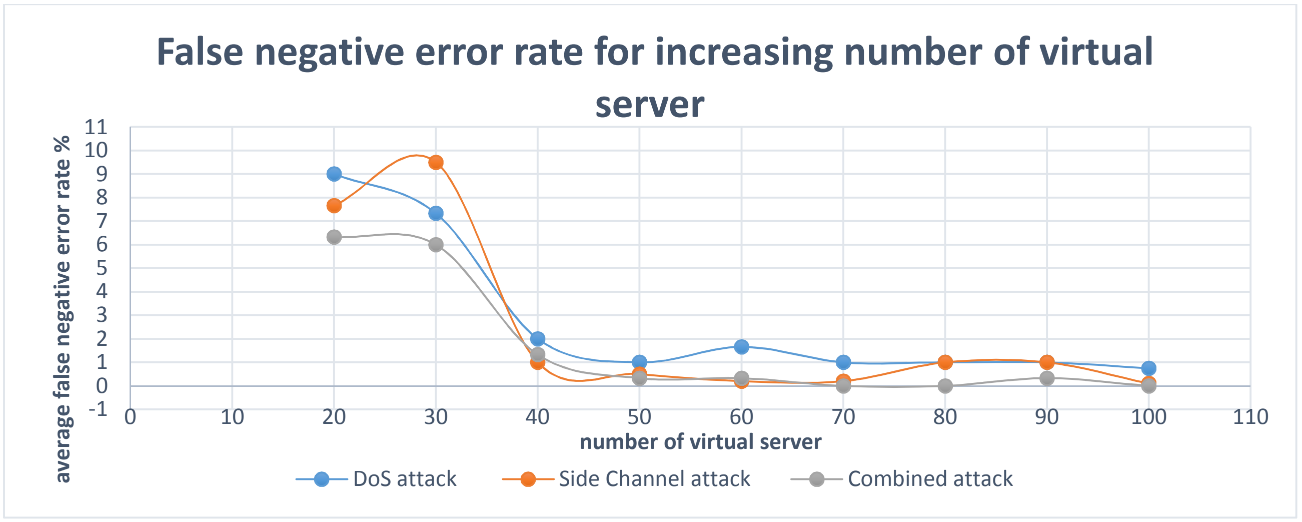
From Figure 8c we can see that for the given number of virtual servers and attackers the average latency of detection of malicious clients is minimum for fewer number of clients. Latency increases as the number of clients increase because the virtual server serves many clients. We can observe that the average false positive and negative error rates increase as the number of clients increase beyond *n*=450, and the detection latency increases linearly.

We can observe that for number of virtual server *k*=60 with attackers *x*=10, approximately 450 clients can be efficiently handled. The false positive and negative error rates are minimal and the average latency of detecting malicious client is also low.

4.3.3 Increasing number of attackers *x*

We set the number of virtual servers *k*= 30 and number of clients *n*=200. We analyze the efficiency of the detection model by simulating it with increasing number of attackers. Simulated Detection model took about 52 to 102 milliseconds to detect each Denial of Service attack and about 50 to 101 milliseconds to detect each Side Channel attack.





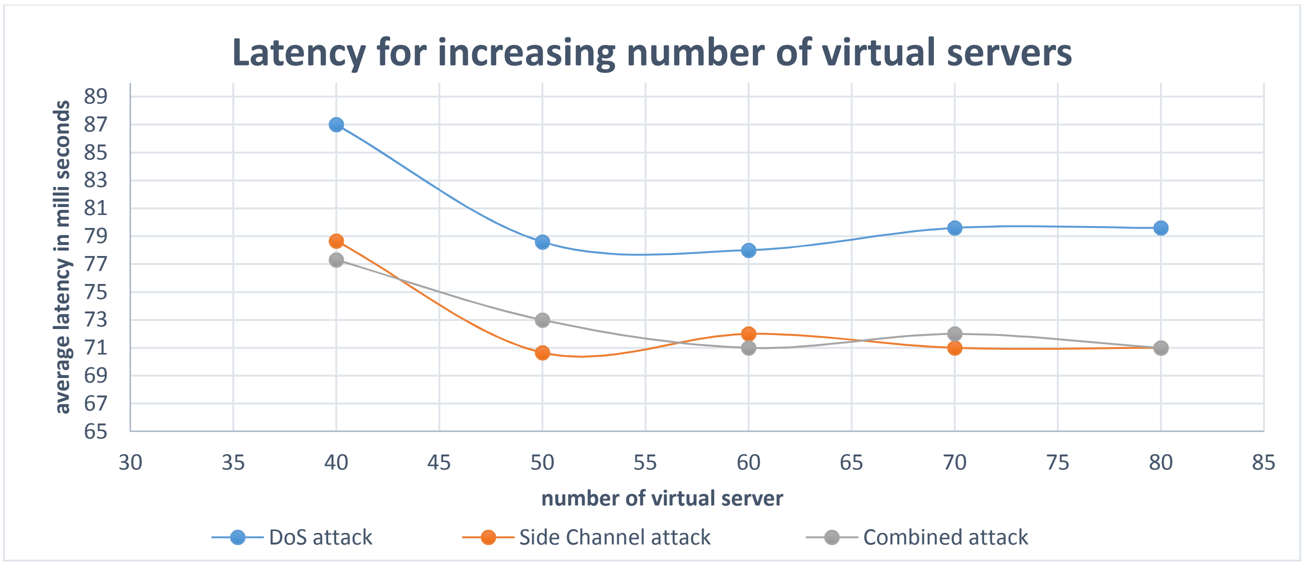
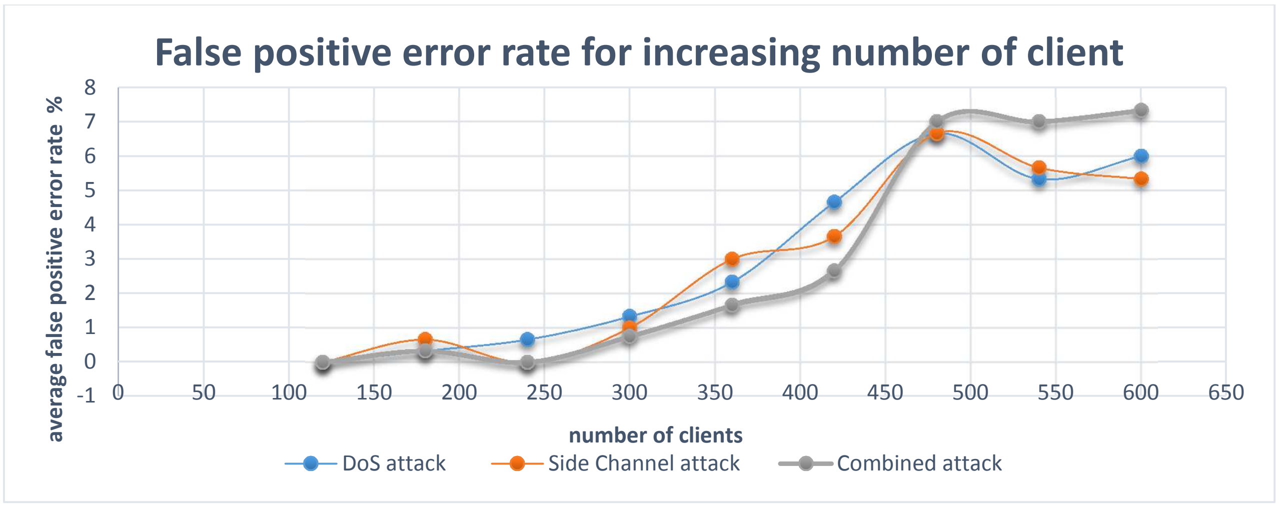
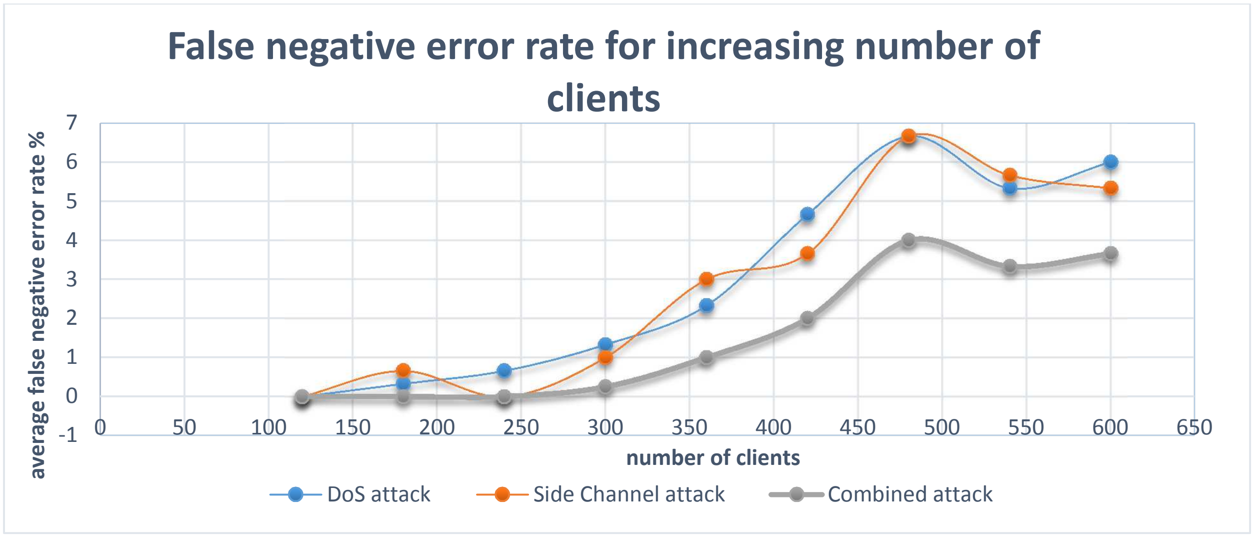


Figure 7a, 7b, 7c: Increasing number of virtual servers: (a) False positive error rate; (b) False negative error rate; (c) Latency





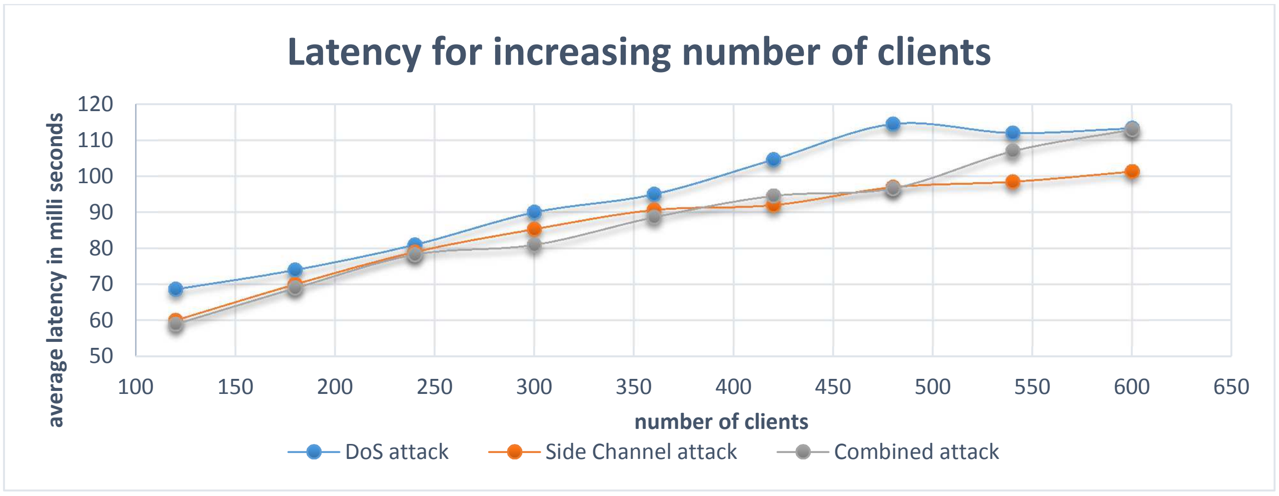
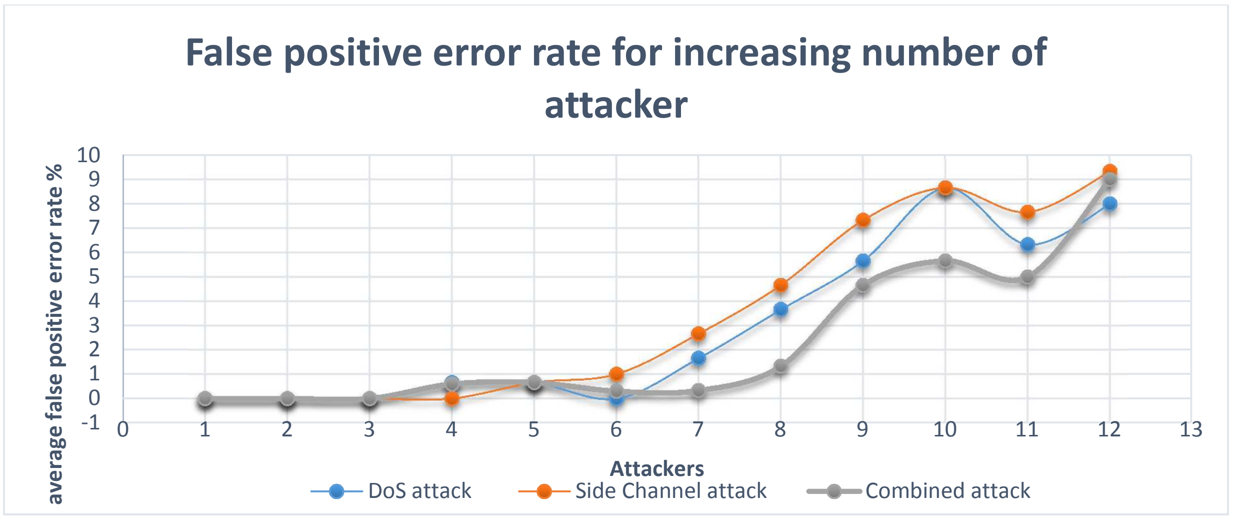
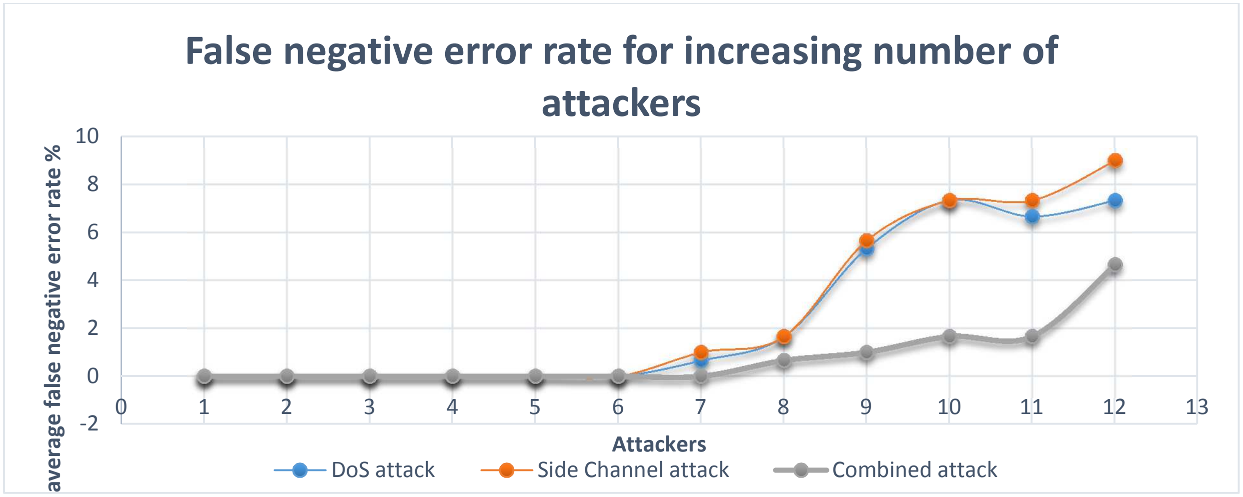


Figure 8a, 8b, 8c: Increasing number of clients: (a) False positive error rate; (b) False negative error rate; (c) Latency





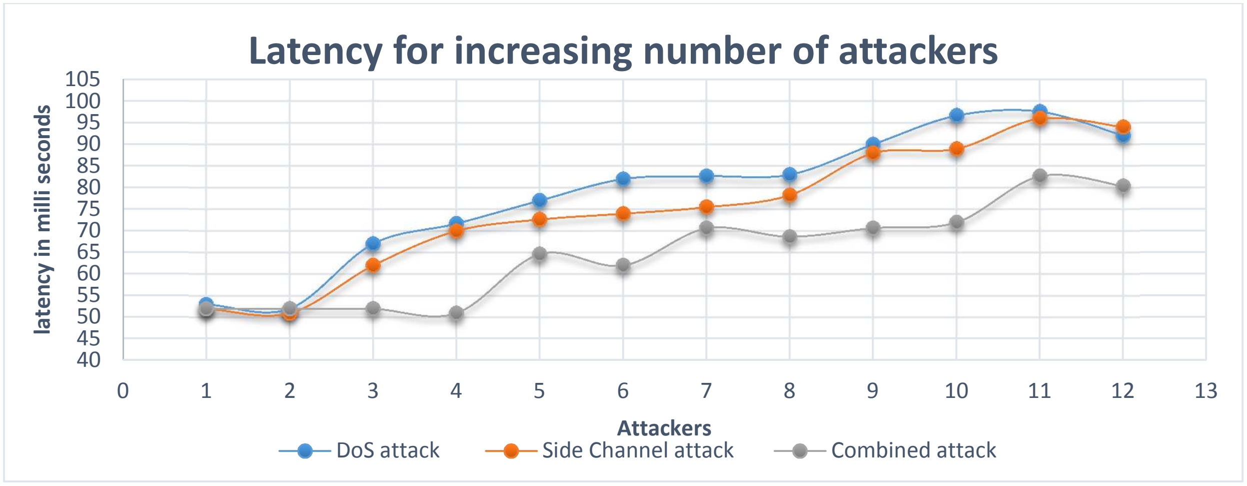


Figure 9a, 9b, 9c: Increasing number of attackers: (a) False positive error rate; (b) False negative error rate; (c) Latency

From Figure 9a we can see that for the given number of virtual servers and clients the average false positive error rate (predicting the safe client as a malicious client) linearly increases for number of attacker *x*=7 and above. The Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage has almost the same average false positive error rate. However, the combined attacks has lower false positive error rate at every instance.

From Figure 9b we can see that for the given number of virtual servers and attackers the average false negative rate (unable to identify the malicious client) linearly increases as the number of attackers *x*=7 and higher. The Denial of Service attack with respect to RAM usage and Side Channel attack with respect to bandwidth usage has almost the same average false negative error rate. We also observe that the combined attack has comparatively less than the average false negative rate.

From Figure 9c we can see that for the given number of virtual servers and clients the average latency of detection of malicious client is minimum with fewer number of attackers. Latency increases as the number of attackers increase. We can observe that even though the average false positive and negative error rate increases as number of attackers increases beyond 7, the detection latency increases linearly as the number of attackers increase.

We can observe that for number of virtual server *k*=60 with number of clients *n*=200, the detection model can efficiently detect up to 7 attackers. In this case, the false positive and negative error rates are minimal and the average latency of detecting malicious clients is also low.

CHAPTER V

CONCLUSIONS

Self-service cloud (SSC) computing provides more administrative power and flexibility to the client to maintain and perform tasks. The cloud administrator is not allowed to directly check the data or computational code of client virtual machines, thereby ensuring protection of the client's sensitive data and privacy against malicious cloud administrators. However inter virtual machine attacks and client VM attacks on the administrative domain and hypervisor can occur in this architecture.

We proposed a SSC with virtual shield (Detection Model) to detect denial of service attack (with respect to RAM usage), side channel attack (with respect to bandwidth usage) and a combined attack for a large population of clients with low latency for detection, low false positive/negative error rate.

The proposed detection model has the capability to detect individual attacks and a combined denial of service and side channel attacks in the initial stages. The attacks detected along with configuration file is sent as output to the Reinforcement Learning algorithm to change the configuration of client virtual machines to mitigate the attacks and make the self-service cloud more secure. Simulation results on the detection model demonstrated the efficient performance of the proposed model in terms of low detection latency and low false positive/negative error rate.

The proposed detection model is light weight as both the attacks are detected using a single group testing method. The proposed approach can therefore be applied to different kinds of attacks. If the detection model fails due to hardware problems, there is no way to defend the system from client attacks. As future work, therefore we can have a secondary backup detection model which keeps track of previous suspects and continue detecting attackers as quickly as possible.

The proposed architecture can be enhanced with different methods of distribution of clients to virtual servers. Thereby the load is equally balanced between all the virtual servers during the group testing. We can implement machine learning techniques in the attack analyzer modules to efficiently detect an attack based on previous resource usage of the clients.

The detection model uses group testing method that can be used to detect other types of attacks efficiently for large population of clients in the self-service cloud.

REFERENCES

[1] Media Temple.Net, http://mediatemple.net/blog/news/2015-will-be-the-year-of-the-cloud-or-was-that-2007/. [July 7, 2015]

[2] Mistakes in the IaaS cloud could put your data at risk, http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/mistakes-in-the-iaas-cloud-could-put-your-data-at-risk.pdf [May 1, 2015].

[3] ZD Net, http://www.zdnet.com/article/the-top-concerns-with-cloud-storage-services/. [July 7, 2015]

[4] 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.

[5] Ying Xuan, Incheol Shin, M. T. Thai, Taieb Znati, "Detecting Application Denial-of-Service Attacks: A Group-Testing-Based Approach" *IEEE transactions on Parallel and Distributed Systems*, August 2010.

[6] 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.

[7] Stephen de Vries. "A Corsaire White Paper: Application Denial of Service Attacks", http://media.techtarget.com/searchSecurity/downloads/Appdosattacks.pdf. V1.0 Released, 2004. [July 10, 2015].

[8] 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.

[9] 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.

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

[11] Greg Mori, Jitendra Malik "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA" *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[12] Thomas Ristenpart, Eran Tromer, Hovav Shacham, Stefan Savage "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds"*, in Proceedings of 16th ACM Conference on Computer and Communications Security,* November 2009, Chicago, Illinois, USA.

[13] Lee Hahn Holloway, Srikanth N. Rao, Matthew Browning Prince, Matthieu Philippe François Tourne, Ian Gerald Pye, Ray Raymond Bejjani, Terry Paul Rodery, Jr., "Mitigating a

denial-of-service attack in a cloud-based proxy service". Publication number -US20140047542 Cloudflare, Inc., California.

[14] Bhrugu Sevak "Security against Side Channel Attack in Cloud Computing" *International Journal of Engineering and Advanced Technology* ISSN: 2249 – 8958, Volume-2, Issue 2, December 2012.

[15] William Y Chang, Hosame Abu-Amara, Jessica Feng Sanford "Transforming Enterprise Cloud Services" *Springer Dordrecht Heidelberg, London New York ISBN 978-90-481-9845-0*, 2010.

[16] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose and Raj Kumar Buyya "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Published online 24 August 2010 in Wiley Online Library (wileyonlinelibrary.com).

[17] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne http://www.cloudbus.org/cloudsim/[11/13/2015].

[18] Balaji Ganesula "Reinforcement learning to reduce the attack surface in Self-service cloud computing", Oklahoma State University, Publication number-1553404, December 2013.

VITA

Rakesh Jayaram

Candidate for the Degree of

Master of Science

Thesis: DETECTION OF DENIAL OF SERVICE ATTACK AND SIDE CHANNEL ATTACK IN SELF-SERVICE CLOUD USING GROUP TESTING STRATEGY.

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2015.

Completed the requirements for the Bachelor of Science in Computer Science at Visvesvaraya Technological University, Bangalore, India in 2012.

Experience:

Technical Assistant, Orgapp Technologies Pvt. Ltd, India Nov 2012- June 2013

Worked on developing front-end web applications application using J2EE, spring. Worked on developing front-end web applications and unit test and debugged the Java program code. Developed documents related to program.