A Novel Approach to Prevent Cloud Infrastructure against Cache Attacks

Bharati Sanjay Ainapure (bharatiainapure@gmail.com)
Vishwakarma University https://orcid.org/0000-0002-2354-5738

Research Article

Keywords: Cloud Computing, Fuzzy logic, Fuzzy rules, Eucalyptus, Private Cloud, Public Cloud, Cache Attack

Posted Date: May 5th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-493941/v1

A Novel Approach to Prevent Cloud Infrastructure against Cache Attacks

Dr. Bharati Sanjay Ainapure
Associate Professor
Vishwakarma University
Pune, India
bharatiainapure@gmail.com

ABSTRACT

In cloud computing resources like software and hardware are shared among multiple users by creating multiple instances of virtual machines (VMs). System cache is one of resource which is shared by Virtual machines in cloud environment. Such Virtual machines are targeted for abnormal activity like side channel attacks. Cache based side channel attack is one of the side channel attacks on cloud environments which leaks the private information of the client. The proposed approach includes the detection and prevention of cache based side channel attack in cloud infrastructure. The approach is based on two components: virtual machine status collection in the form of log files and use of fuzzy logic to detect and prevent the cache attack. Fuzzy logic defines set of fuzzy rules to identify the cache attack from the log file. The proposed approach could attain a maximum of 78.15%, 80.04%, and 82.16% of accuracy at 10, 20 and 30 sec. of time intervals.

Key Words: Cloud Computing, Fuzzy logic, Fuzzy rules, Eucalyptus, Private Cloud, Public Cloud, Cache Attack

INTRODUCTION

Cloud computing is a service offered over the network in which, multiple tenants (clients) are allowed to share actual physical resources. Physical resources in cloud computing are shared among multiple clients which makes the client to own the resources as per their use by paying some cost. Therefore, these clients are called as multi-tenant clients. With the help of virtualization and hypervisor multi-tenant clients are able share the actual physical resources available in cloud environment. With the help of Virtual Desktop Infrastructure (VDI), cloud providers can run multiple clients on physical resources and clients can work on their favorite desktop and application. These virtual machines are hosted on physical resources on the cloud. The advantage of VDI is that the clients are allowed to utilize their desktop anywhere and anytime through the devices like laptop, desktop, thin clients, etc. connected to a network over the public cloud.

VDI was introduced in the year 2008. This technology has been rising rapidly with virtualization technology and cloud computing [1]. One considerable advantage of VDI is that, it can install all required drivers, applications, OS and other programs onto a single image so that all clients of cloud can use. Unfortunately, different clients demand different images as per their requirements. To serve such clients, VDI needs to create separate images, and there is a need to update each of these clients' images. This difficulty leads to the elimination of the convenience of the system. Therefore the proposed system uses the private cloud to deploy the experimental setup. However, the proposed work is also capable of handling the cache based attack on the public cloud which is demonstrated using CloudSim public cloud environment. Both the experimental setups are explained in the later section.

Either VDI or private cloud, images of virtual instances needs to be updated regularly. But maintenance of many virtual instance images will become a headache to the cloud provider and less security issues. As everyone knows that cloud computing depends on virtualization, where resources

are used in sharing basis will create side channel attack problems. There are many side channel attacks present in a cloud environment, to name few, power analysis attack, fault attacks, acoustic attacks and cache side channel attack. Among these cache side channel attack is more powerful and it is proved on the cloud environment in 2009 on Amazon EC2 cloud. Cache based side channel attacks can be performed in a cloud environment on virtual machines.

The multi-tenancy, where sharing of resources is provided, leads to the security problem in both VDI and private cloud. Therefore, the virtualization adds new attack surface and new challenges to prevent attacks based on the resource sharing. One of the resource sharing attacks is side-channel attacks on virtual machines in the cloud environment. Cache-based side channel attack is one of the side channel attack present in the cloud.

Cache memory is a small memory introduced in the processor to fill the gap between the latency of the memory and speed of the processor. The cache memory may be shared by different processors or may have individual caches depending on the microarchitecture of the processor. Latest processors have multiple levels of cache. Core i7 processor has three levels of cache memory as shown in Figure 1, for a different purpose.

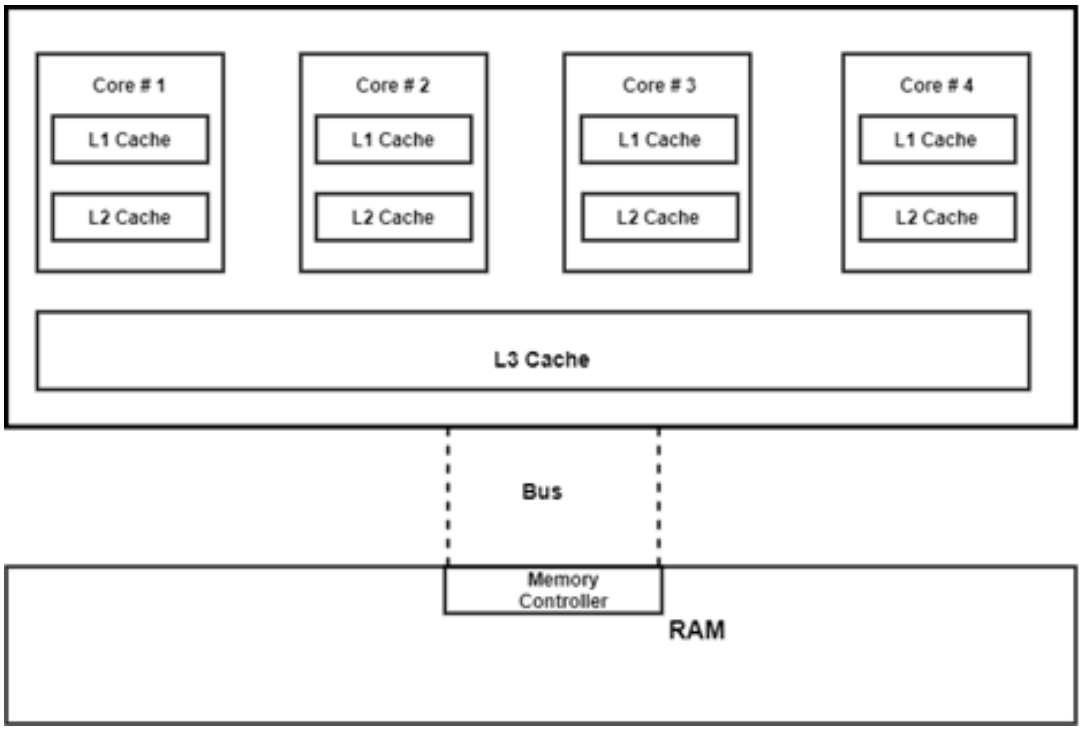

Figure 1. Cache memory hierarchy

Whenever data or instructions are referred by the CPU, it is considered as the first request to L1 level cache as depicted in Figure 1. If the requested data is found in L1 by the CPU, then, it is called as a cache hit. If it is not found in the L1 cache, then it is called a cache miss. If a cache miss is experienced, the data is next looked for in the next level of memory - for an L1 cache miss; this would be the L2 cache. If found data is propagated back through each level of cache that experienced, miss to locate the data on the cache and this process of a cache hit and miss are continued with the next level of caches. Generally L2 cache is much larger in size than L1 and L3 is larger than L2. L3 is the last level of cache, which stores data from multiple cores simultaneously. If a cache misses at L3 cache occurs, the requested data are sought in main memory [2].

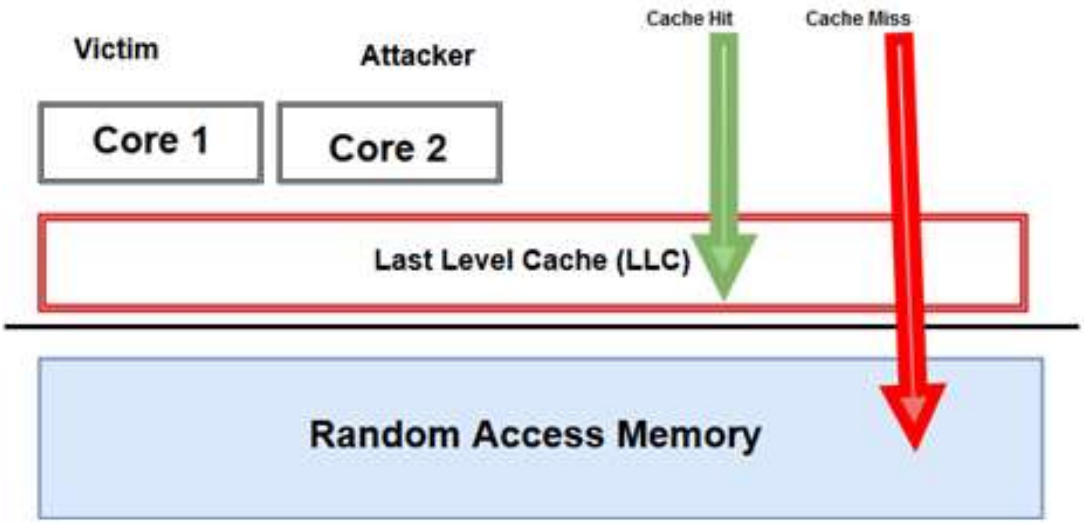

Figure 2. Cache based attack on last level cache

As shown in Figure 2, to use the cache as a side channel to make the attack, some of the other way cache needs to share between attacker and victim. If one can think the scenario of Figure 1, in a cloud environment, it is very easy to share the cache between the different VMs launched on the same CPU core as shown in Figure 2. Typically, a CPU cache can be shared in two ways, either the cache is exclusive to one CPU core, in which case two processes must access the cache sequentially [4]; or cache is shared between the CPU cores, in this case two processes can access the cache concurrently. For the first time, in 2000, J. Kelsey & etl. , in their publication mentioned about the cache based side channel attack based on the cache hit ratio [3].

As mentioned above, the CPU accesses data from the cache, if that data is not found in the cache, then a cache miss occurs [20]. This causes the delay to load the data from memory into cache. This delay gives scope to the attacker to attack the cache. The measurement of this delay enables attackers to determine the occurrence and frequency of cache misses. This technique of measuring the delay due to a cache miss is called as cache-based side channel attack as shown in Figure 2.

This paper proposes novel approach to detect and prevent the cache based attack in cloud using fuzzy controller. The system makes use of a log file generated by VMs to identify the attacker to reduce the cache-based side channel attack.

The rest of the paper is followed by the literature review, system overview and four different inputs for the fuzzy controller, the cache-attack detection and mitigation using the fuzzy system experimental setup, results, comparison analysis and lastly conclusion of the paper.

LITERATURE REVIEW

Cache-based side channel attack is not a new concept in literature. This type of attack is possible in the cloud due to resource sharing. For the first time in 2009, cache-based side channel attacks were proved on the Amazon EC2 cloud environment [12]. This section will discuss researchers approach and different techniques used to perform the cache-attack and their mitigation.

Cache-based attack was proved on the victim machine using memory bus contention. In this attack the x86 machine was used to prove the attack by exploiting the instruction level cache. This attack was accomplished with the help of some hardware alteration which was resulted into overhead in processors and also mitigation techniques were not proposed [6].

Authors in [4] developed two cache-based side channel attack mitigation strategies by investigating the purpose of CPU-cache based side-channels in the cloud compared to the conventional side-

channel attacks. It considered two server-side defenses, of which one focused on sequential side-channels that had a technique to control the side-channels occurrence and an algorithm to minimize overhead. The other defense focused on parallel side-channels using a cache coloring technique to prevent the side channel's occurrence and to enhance the cache efficiency. Here, the cache was flushed between the prime and trigger steps. Thus, the implemented defense could prevent the side-channel attacks. However, the process of flushing is time-consuming.

Authors have presented a technique that influenced dynamic cache coloring to prove that stealing cryptographic information in the cloud could be a threat by finding a cache-based side-channel attack against an encryption process. Dynamic cache coloring referred to a technique, where the VMM was informed of changing the related data to a safe cache line. Thus, the approach had reduced cache-based side-channel ensuring effective resource sharing. The implementation of this approach depending on Xen and the overhead performance had demonstrated its applicability. The performance reduces in cache isolation while returning pages when the protection system is stopped [6].

Authors in [8] developed a detection approach, called CSDA, to detect cache-based side channel attacks and thereby, reduce the security threat in cloud-based on the side channel effects in resource utilization. The technique was comprised of two-stage detection mode: host detection and guest detection, which combined two tests, shape, and regularity tests to mine the features that attack from hosts and guests. This utilizes pattern recognition techniques to identify the malicious VMs from the original VMs. However, the test based detection techniques degrade the performance.

The machine learning technique is applied to detect cache-based side channel attack. Authors have mainly focused on flush and reload method to detect the spy planted on the victim through cache attack. In this paper, authors used the neural network in the supervised mode to detect the attacks [8].

Analysis of cache miss patter is made in [21] to secure the VMs from cache side-channel attack. Authors have implemented the algorithm to detect the cache miss in the system. They have used the cache profiler tool to analyze the cache miss rate. After analyzing the cache pattern, the authors have introduced the noise in the cache to confuse the attacker which prevents the cache-attack. But noise introduction has created the overhead to the system which has degraded the system performance.

In the above mentioned literature authors have found the solution to mitigate or detect the cache side channel attack by implementing the software or they modified the hardware. However, it is challenging to propose a new preventive measure which will not involve any of the overhead and without altering any hardware component. So, the proposed approach deals with the cache side channel in a cloud environment, which collects system calls in the form of logs from virtual machines running on the same hypervisor. These system calls are used to record the Input/output operation carried out by virtual machines. This system call collector is supposed to run at the hypervisor level. There are many suitable, system call tracer tools which are available to run at cloud environment. Few of such tools are explained as below.

Nitro is Virtual Machine Introspection (VMI) tool based on hardware System Call Tracing. It is integrated with KVM (Kernel Virtual Machine) hypervisor to monitor the VMs. It is hardware-based system call tracing and monitoring tool. Intel x86 architecture provided three different types of system calls is traced by this tool. During monitoring of VM, this tool collects the information of systems calls. The information collected is raw bits, and then this tool converts these raw bits into human-readable form to detect the rootkit attacks on VM. The advantage of this tool is that it can collect system calls from 64-bit and 32-bit Window and Linux based VMs running on a hypervisor [9].

Ether is one of the VMI tool. It is used to detect the malwares in the system. It makes use of the Intel VT extension to detect the malware in the system through the system call tracing. This uses XEN open source hypervisor to detect the malware. Basically, its three functionalities namely systems call tracing, instruction traces and memory writes [10].

LibVMI is an open source application programming interface (API). This uses C library with Python bindings to create Virtual machine introspection functions. Low-level information of the VM running on core is collected by this tool. It extracts information about vCPU registers, hardware events, and memory of the guest. Open source hypervisors like XEN and KVM used to run the LibVMI [11].

Among various attacks in cloud computing, cache-based side channel attack is one that leaks private information regarding users based on the shared resources. Here, as the shared resource is the cache, a process can utilize the cache usage of another by cache contention. Cache sharing provides a way for the attackers to gain considerable information so that the key used for encryption can be inferred. Discovering this channel attack is a challenging task considered in the proposed system. This requires identification of a feature that influences the attack, which is also a challenge. Even though there are various techniques available [6][8][21] in the literature to mitigate cache side channel attacks but an effective solution to reduce the cache-based side channel attack is still an issue. Therefore, a novel approach is proposed in this paper to detect and alleviate the cache attacks using security properties for effective and secured communication.

SYSTEM OVERVIEW

The proposed approach incorporates a novel method of detecting and mitigating the cache based side channel threat criticality over VMs. Hence the prime motive of the undertaken method is to identify the vulnerable VMs considering that Cloud Service Provider is harmless. In the proposed approach, the focus is majorly laid on depicting the pattern of malicious activities like cache-based side-channel attack of an attacker towards the VMs. Hence, the method will be designed that will furnish all the known cache-based side channel threats in the system and all sorts of VM connectivity information giving a some true s picture for predicting the possible cases of vulnerabilities for uncompromised VMs. The system consists of four major components: 1. Malicious user in the cloud environment, 2. the target VM in the cloud environment, 3. System calls collector Nitro [9], and 4. Method of detection and mitigation of cache-based side channel attacks, the Fuzzy Controller as shown in the Figure 3. The cache-based side channel attack is carried out by placing the malicious as co-resident with the target machine [12]. After placing VM as co-resident, the attacker has to establish a side channel with target VM. Then perform the cache based side channel attack on the target VM [19]. The system call collector at hypervisor needs to collect all system calls as logs, at the time of the system call trap. The log file includes the information about the five parameters, which are explained in further section. This technique of system call tracing has been elaborated in next section.

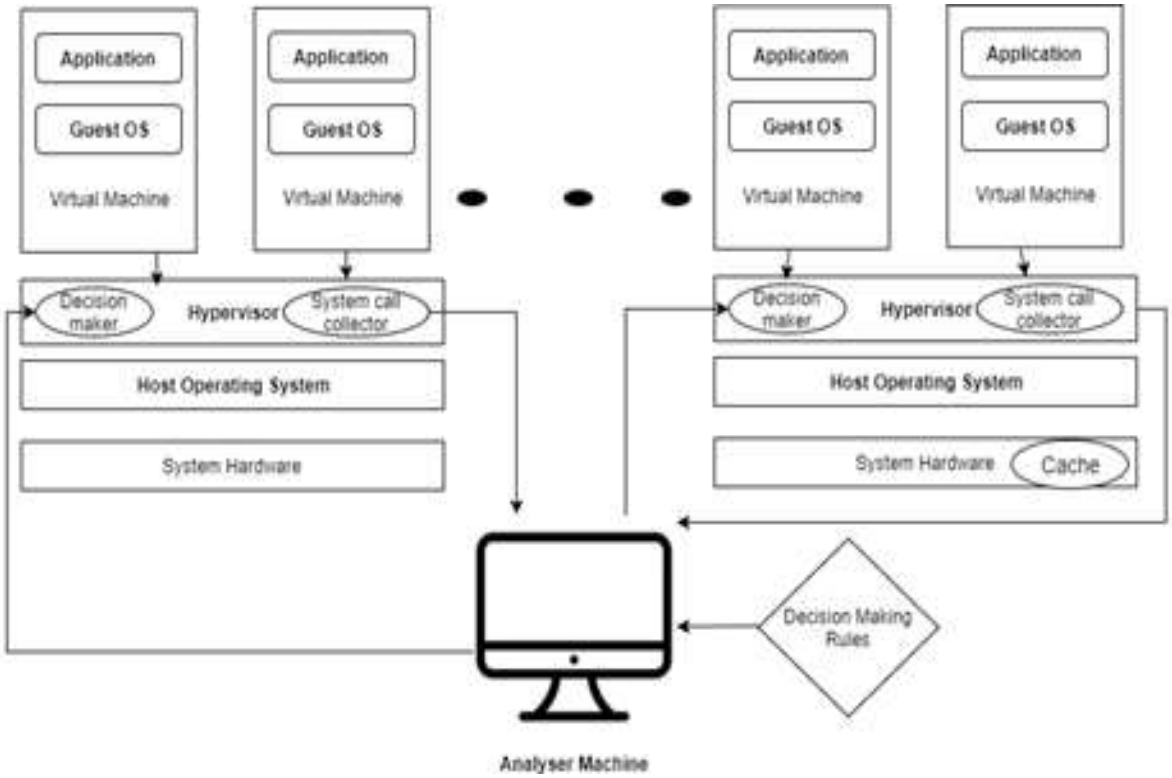

Figure 3. System Architecture for proposed system

Method of Collecting System Call Traces

The VM running in the cloud environment needs to generate a system call to the hypervisor so that hypervisor can trap this system call. Such system call trapping is performed in a virtualized environment with the help of operation known as VMX (Virtual machine extensions) [23]. There are two VMX operations performed in a virtualized environment by processors those are VMX root and VMX non root [23]. Normally VMX root operation is executed by VMM, and VMs running on VMM executes VMX non root. The VM running in user space executes the VMX non-root operation, which does not have any permission to perform the kernel space commands like input/output (I/O). In this case, if the VM is generating such system calls, then this VM needs to perform VMX non-root operation to VMX root operation. Such transition from non-root to root operation will generate a trap to the hypervisor. So VM exit will occur from user space to kernel space. It is shown in Figure 4. Then hypervisor records the calls generated by vm exit and VM are allowed to perform I/O in kernel space. Once I/O is executed, and then again VM entry is performed, which returns the call back to the user space of VM which is VMX non-root operation through the hypervisor. The system call collector, running at hypervisor needs to collect all the system calls at the time of the system call trap. At some specific interval of time, this collected data is sent to analyzer machine using the socket interface. The socket interface uses method send () to send the data from the system call collector to the analyzer. The socket stream may be written as follows:

Socket() ---> send(socket stream, VMIP, port-number);

Detection of cache-based side channel attack can be performed by measuring the loads on the cache. The load on the cache is detected using cache access during I/O operations performed by VMs running in the cloud environment. These I/O accessed data are collected by the system call collector which is running on a hypervisor that is outside the VM to protect it from attacks. This system call collector will collect cache load variation in VM.

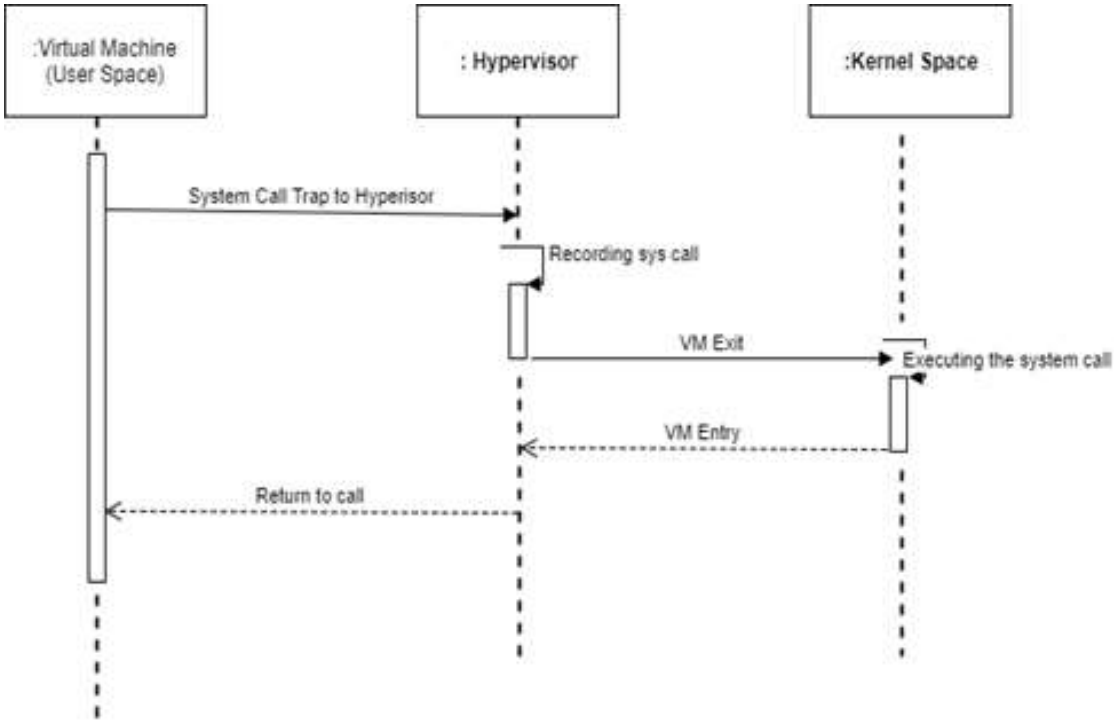

Figure 4. Sequence to track the system call

Detection of cache-based side channel attack can be performed by measuring the loads on the cache. The load on the cache is detected using cache access during I/O operations performed by VMs running in the cloud environment. These I/O accessed data are collected by the system call collector which is running on a hypervisor that is outside the VM to protect it from attacks. This system call

collector will collect cache load variation in VM. This load variation on cache data collected by system call can be concluded using following five parameters:

- *Cache Data Access (CDA):* This feature is used to understand the quantity of cache resource utilized by a user throughout the cache data access. Cache miss occurs due to more utilization of cache. It enables to predict about the attacker. Cache data access is given by following equation.

$$C_d = C_a + C_m * p$$

Where, C_d is cache data access rate, C_a is cache access time, C_m is cache miss rate and p is the cache miss penalty.

- *Size of Cache Data Access (SCDA):* The amount of space utilized by cache user is identified by this feature. In cloud environment the user one who utilizes large amounts of the cache is identified as an attacker. Due to this, cache miss rate for legitimate user increases, since the user tries to consume a large sized cache. The size of cache data access is given by following equation.

$$C_s = C_d * C_m$$

Where $C_s$size of cache data access is, C_d is cache data access and C_m is cache miss rate.

- *Cache Miss Rate (R):* This feature is used to detect cache side-channel attacks. Prime and probe operation are used by cache contents attacker to read and write cache memory continuously. Due to prime and probe operation, cache will get evicted frequently. This will result into a huge cache miss rate during the attack. Cache miss rate of the system is given by following equation.

$$C_m = 1 - C_h$$

Where, C_m is cache miss rate and C_h is cache his ratio.

- *Virtual Memory Utilization Pattern (M):* Due to massive cache misses rate utilization of virtual memory changes as virtual memory is occupied and released continuously during the attack. So this becomes the fourth feature to measure the cache based side channel attack. Virtual memory utilization rate is given by using following equation.

$$R_v = (P_n * P_z)/V_v$$

Where, R_v is rate of virtual memory utilization of VM, P_n is number of pages the VM has in RAM, P_z is the size of page and V_r is RAM allocated to VM,

- *Virtual CPU Utilization Pattern (V):* CPU utilization pattern is one of the most important features in finding cache-based side channel attack in a cloud environment. As cache miss rate is more in attack due to more read and writes operation during cache load measurement, which is not computationally intensive operation. Due to this more time CPU will make switching between the waiting for data in the cache and processing, which leads less CPU utilization. In this way, one can measure the less CPU utilization and can identify cache-based side channel attack. . The virtual CPU utilization rate is given by following equation.

$$V_u = V_r/V_C$$

Where, V_u is rate of utilization of virtual CPU, V_r is required rate of CPU for application running in VM i.e. busy time and V_c is the CPU capacity of VM which is virtual CPU busy time and idle time.

The next section explains how proposed system makes use of the above-mentioned parameters to detect the cache-attack in the cloud.

Detection and Mitigation of Cache Attack on Cloud

The proposed system uses the fuzzy controller to detect the malicious user in the cloud environment. Fuzzy logic is useful for proposed system because firstly it builds on user-supplied human language rule-based approach i.e., the linguistic variables for accepting imprecise and incomplete information. Secondly, the data collected from VMs for detection in less compared to large datasets. Lastly, it is very difficult to identify true positive and false negative rate of the malicious user when data is uncertain. Therefore the fuzzy controller is used in the proposed methodology. Once the log is collected at the hypervisor level from virtual machines, then this log is used to detect the cache-attack using a fuzzy controller as shown in Figure 5.

The recorded log file is used as input to the fuzzy system to determine the behavior of the user. Fuzzy processing involves the execution of IF...THEN rules, which are based on the input conditions. The input to the system is fuzzy sets with set members. The fuzzy logic system takes a decision from the knowledge base with the aid of a rule base involved in the fuzzy inference [18]. Defuzzification refers to the technique of converting the fuzzy details on the solution space into a crisp value. Thus, the fuzzy rule system decides whether the user is an authenticated user or an attacker based on their behavior.

The fuzzifier consists of fuzzy sets as follows:

1. Cache data access denoted by CDA
2. The size of cache data access denoted by SCDA
3. Minimal cache miss rate R
4. Virtual CPU utilization rate, V and
5. Virtual memory utilization rate M

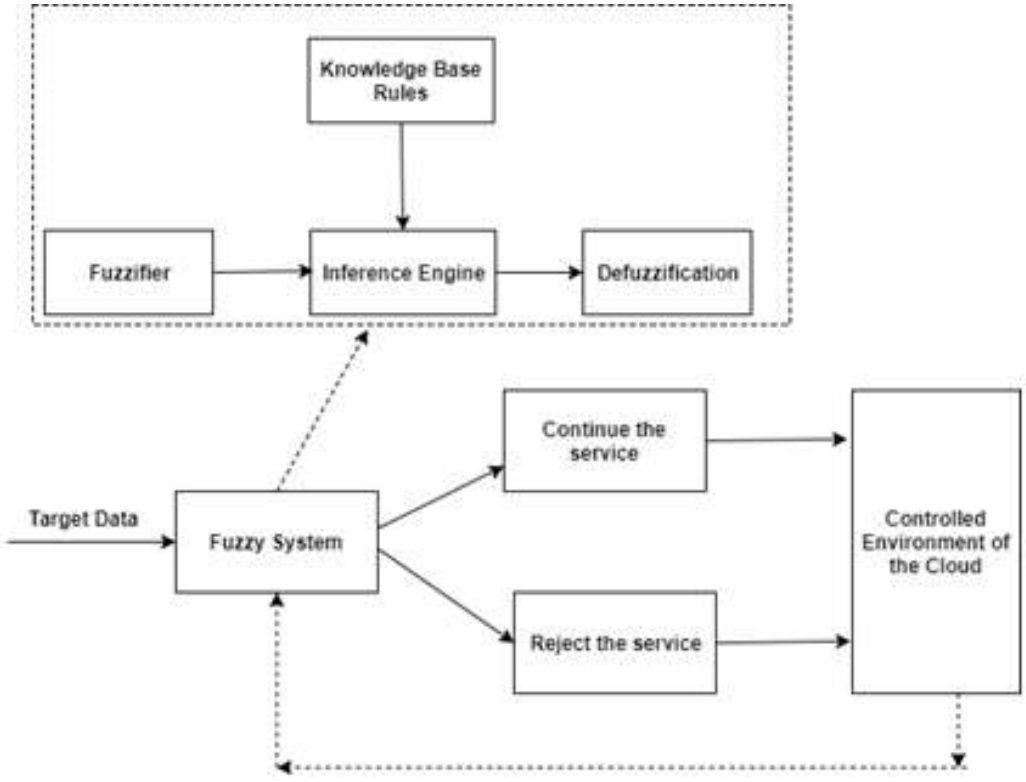

Figure 5. Fuzzy controller system

Each fuzzy set contains members and each member in the fuzzy set is identified with some degree of membership functions. These functions act as an input variable and are shown as below:

i) $CDA = \{Y\}$

ii) $SCDA = \{S, L\}$

iii) Cache miss rate, $R = \{\text{S, L}\}$

iv) Virtual CPU utilization rate, $V = \{\text{S, L}\}$

v) Virtual memory utilization rate, $M = \{\text{S, L}\}$

Where Y denotes Yes, S and L denotes Small and Large, respectively, and degree of membership function varies from [0,1]. Depending on the data collected in log file, the variables and the labels of fuzzy sets can be combined to create rules know as knowledge base rules as given below:

1. Rule 1: If CDA is Y and $SCDA$ is S, then the decision is to continue with service of the client. This decision is communicated to the decision maker present in the proposed system.
2. Rule 2: If CDA is Y and $SCDA$ is L, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
3. Rule 3: If CDA is Y and R is S, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
4. Rule 4: If CDA is Y and R is L, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
5. Rule 5: If CDA is Y and V is S, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
6. Rule 6: If CDA is Y and V is L, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
7. Rule 7: If CDA is Y and M is S, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.
8. Rule 8: If CDA is Y and M is L, then the decision is that the client running on that virtual machine is attacker. This decision is communicated to the decision maker present in the proposed system.

The first rule states that if CDA is 'Yes' and $SCDA$ is Small, the fuzzy system decides that the user is an authenticated user. Hence, the system holds the process. The statement in rule 2 suggests that when CDA is 'Yes' and $SCDA$ is 'Large' then one who accesses the cloud is an attacker, as the user tries to consume a large-sized cache. Similarly, the rules 3 to 8 are framed for the minimal value of the resource utilization, minimum miss rate, and minimal memory utilization. Whenever the CDA is 'Yes' and R is 'Small', then the fuzzy system finalizes the user as a genuine user and continues to communicate. Therefore, the process of communication is rejected by fuzzy system, and same is communicated to decision maker. Then this decision maker may relocate the attacker VM to different physical machine or may reject the service for the attacker. Thus, in a specific time interval, the fuzzy logic system records the information regarding the user in the log file, based on which it develops the rules and determines whether to hold or reject the process. With the decision made, the proposed system discovers the attacker thus, lessening cache-based side channel attacks.

Rules 1, 3, 5 and 7 represents the mathematical operation AND. Hence, from both fuzzy sets, the {min} value of the membership function should be taken into consideration. The mathematical operation OR is used represent the Rules 2, 4, 6 and 8. Therefore, from both fuzzy sets, the {max} value of the membership function should be taken into consideration. The mathematical representation of fuzzy rules and their matrix representation are as follows:

Rule 1 & 2

Table 1. Fuzzy rules 1 & 2

Rule No.	CDA	SCDA
1	Y	S
2	Y	L

$$\mu(\overline{cda} \cup \overline{scda}\,(x)) = \max(\mu_{\overline{cda}}(x),\, \mu_{\overline{scda}}(x)), \forall\, x \in X \quad (1)$$

$$\mu(\overline{cda} \cap \overline{scda}\,(x)) = min\left(\mu_{\overline{cda}}(x),\, \mu_{\overline{scda}}(x)\right), \forall\, x \in X \quad (2)$$

Rule 3 & 4

Table 2. Fuzzy rules 3 & 4

Rule No.	CDA	R
1	Y	S
2	Y	L

$$\mu(\overline{cda} \cup \overline{r}\,(x)) = \max(\mu_{\overline{cda}}(x),\, \mu_{\overline{r}}(x)), \forall\, x \in X \quad (3)$$

$$\mu(\overline{cda} \cap \overline{r}\,(x)) = min\left(\mu_{\overline{cda}}(x),\, \mu_{\overline{r}}(x)\right), \forall\, x \in X \quad (4)$$

Rule 5 & 6

Table 3. Fuzzy rules 4 & 5

Rule No.	CDA	V
1	Y	S
2	Y	L

$$\mu(\overline{cda} \cup \overline{v}\,(x)) = \max(\mu_{\overline{cda}}(x),\, \mu_{\overline{v}}(x)), \forall\, x \in X \quad (5)$$

$$\mu(\overline{cda} \cap \overline{v}\,(x)) = min\left(\mu_{\overline{cda}}(x),\, \mu_{\overline{v}}(x)\right), \forall\, x \in X \quad (6)$$

Rule 7 & 8

Table 4: Fuzzy rules 7 & 8

Rule No.	CDA	M
1	Y	S
2	Y	L

$$\mu(\overline{cda} \cup \overline{m}\,(x)) = \max(\mu_{\overline{cda}}(x),\, \mu_{\overline{m}}(x)), \forall\, x \in X \quad (7)$$

$$\mu(\overline{cda} \cap \overline{m}\,(x)) = min\left(\mu_{\overline{cda}}(x),\, \mu_{\overline{m}}(x)\right), \forall\, x \in X \quad (8)$$

The mathematical representation of Rule 1, 3, 7 and 7 are AND operation, so {min} value of the membership values of both the fuzzy sets should be taken into consideration. The mathematical representation of Rule 2, 4, 6 and 8 are OR operation, so {max} value of the membership values of both the fuzzy sets should be taken into consideration.

The fuzzy logic system will produce crisp set in the form of output. These types of crisp sets are produced due to defuzzification process in fuzzy controller. This crisp set is used to mitigate the

cache-attack by continuing the service of the process by providing data access to the cloud user or reject the process.

EXPERIMENTAL SETUP

This section presents the experimental setup and results of the proposed system providing cloud security to offer virtual machines to the user. The experimental setup is explained in two different ways considering the public and private cloud properties. Moreover, the performance of the proposed approach is evaluated in a comparative analysis are discussed in the following subsections.

Experimental Setup for Private Cloud

Experiments for the proposed work are conducted on the private cloud framework, Eucalyptus [22]. It includes mainly three components: 1. Cluster Controller. 2. Cloud controller and 3.Node Controller. Cloud controller (CLC) is an entry point for all the stakeholders of the cloud. CLC gets information about resources and makes the high-level decisions about scheduling. Based on this decision Cluster controller (CC) schedules the virtual machines on node controllers (NC). CC has network connectivity between the NC and CLC. CC gets the information from NC and manages the virtual machine scheduling. NC hosts the VM instances. NC controls VM activities like execution, termination, and inspection of VM instances. The experimental set up is depicted in Figure 6 with a physical machine having a minimum of 4GB RAM and 500 GB hard disk. The physical machine should have VT-enabled processor.

The proposed methodology used the system call trace to detect the cache-attack on virtual machine. Nitro [9] is used to collect the system call traces, and it is installed on the node controller. All the system call collected at NC is sent to the CC using socket interface. A fuzzy controller is configured on the CC. This fuzzy controller running on the CC produces the crisp set which is communicated to CLC to take a high-level decision. Then CLC decides about the user services, whether to relocate the VM or continued on the same host.

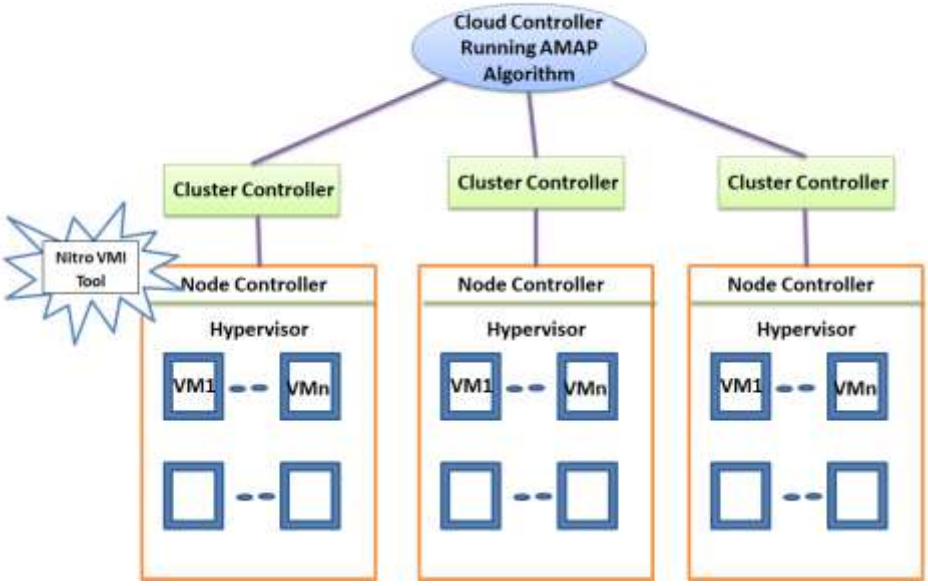

Figure 6. Experimental setup for private cloud

Experimental Setup for Public Cloud

The simulation of the proposed method is executed in a system operated with Windows 10 having the following configurations: Intel processor CPU 2.16 GHz, the memory of 2GB and 64-bit OS. AC and

cloud servers are simulated using the ClouSim [24] tool with the setup of five PMs that consist of 12 VMs through which the users can access the cloud data. The cloud users programmed using JAVA and interfaced with cloudsim. A random model considering a specific number of attackers is also simulated. Depending on the multiple parameters such as CDA, SCDA, V, M, and R, the behavior of the attackers will be simulated.

RESULTS AND DISCUSSIONS

The proposed system considers three parameters, such as number of genuine users, number of attackers, and accuracy to evaluate its performance.

Accuracy of the System

Performance of the proposed system measured using one of the known parameter, accuracy. It is used defines measure the degree of closeness of a value to a standard value. It is represented in the following equation.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (9)$$

Where, TN is True Negative, TP is True Positive which identifies the correct cache attacker and genuine users. FN is False Negative, which identifies the falsely genuine user as attacker and FP is False Positive, which is falsely identified as the attacker.

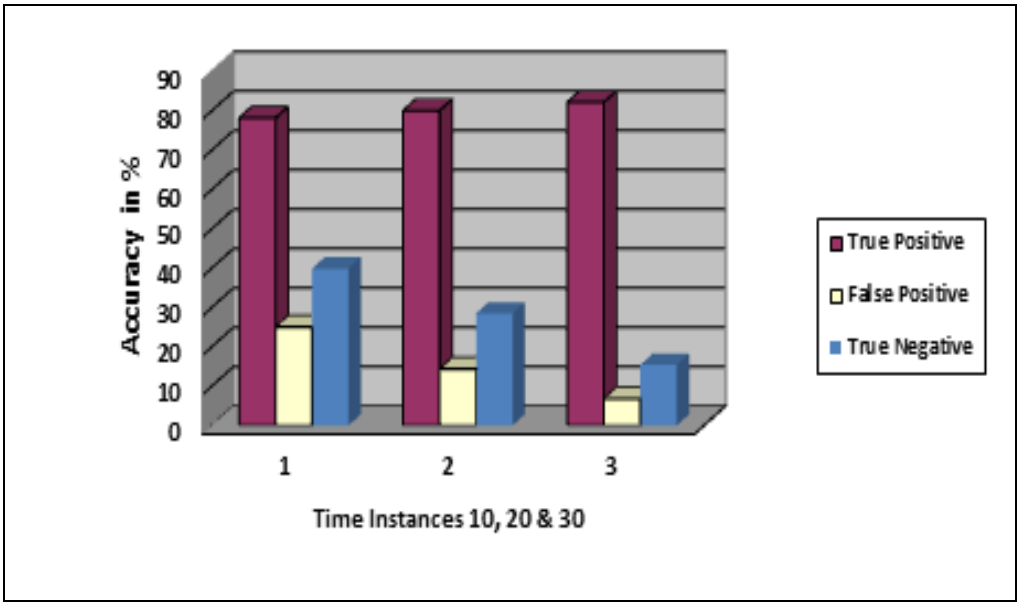

Figure 7. Accuracy graph of proposed system

Figure 7 presents a graph showing the estimated details regarding the accuracy of the proposed system to identify the true positive, false positive and true negative rate of users.

The true positive rate in proposed methodology identifies number true cache attackers in the cloud environment. The details are shown for the time instants 10, 20 and 30 sec. This graph is shown in the cloud, in which there are five cloud servers, and twelve users virtual machine are running. The number of cache attackers details identified *is* 78.15%, and 80.04%; and 82.16% and in the proposed, at time 10, 20, and 30 sec, respectively. The maximum increment is obtained at time 30 sec, with 82.16% in proposed protocol.

The false positive rate in proposed methodology identifies a number of cache attackers as genuine users in the cloud environment. The number of cache attackers details identified is 25.06%, 14.38%,

and 6.85% in the proposed, at time 10, 20 and 30 sec, respectively. The very low value of false positive rate is obtained at time interval 30 sec with 6.85%, in the proposed system.

The true negative rate in proposed methodology identifies a number of genuine users identified as cache attackers in the cloud environment. The number of cache attackers details identified is 39.90%, 28.5%, and 15.4% in the proposed, at time 10, 20 and 30 sec, respectively. The very low value of true negative is obtained at time interval 30 sec with 15.4%, in the proposed system.

Performance Comparison

The proposed system detects and prevents the user's virtual machines from cache attack in a cloud environment. In this section proposed system is compared with other similar systems in performance analysis in two different ways. Firstly, the performance overhead analysis carried out and the performance of cache attack detection analysis. The proposed system used the equation (10) to calculate the performance overhead:

$$P = \frac{DT(VM) + MT(VM)}{Time\ taken\ by\ VM\ in\ normal\ environemt} * 100 \qquad (10)$$

Where P is the performance overhead, $DT(VM)$ is the time taken by the proposed system to detect the attack and, $MT(VM)$ is the time taken by the proposed system to mitigate the attack.
As shown in the Figure 8 performances overhead of the proposed system is very less that is only 1% as compared to performance overhead with existing systems [21].

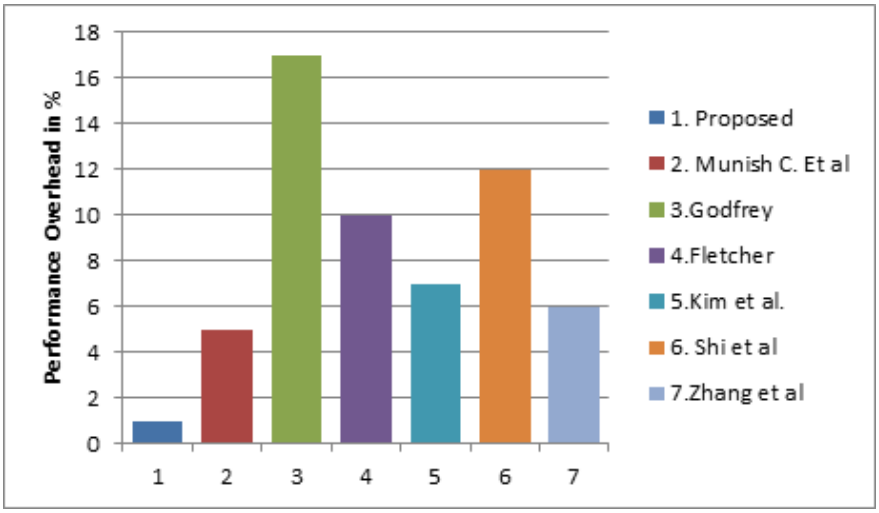

Figure 8. Analysis of performance overhead [21]

Detection accuracy of the proposed system is compared with three existing techniques [4][7][21]. Table 5 depicts this comparison. Changes to the hardware are not required to implement the proposed system and also it is operating system independent. Overall comparisons show that the proposed approach is more efficient and effective offering better results for cache-based security.

Table 5. Analysis of Accuracy for proposed system

Techniques	Detection Accuracy
Michael Godfrey, etl.	73%
Si Yu, etl.	62%
Munish C. & Halabi H.	> 60%
Proposed Approach	**82.16%**

CONCLUSION AND FUTURE WORK

Recent it is found in the cloud that many side channel attacks are proven due to multitenancy. The cache-based side channel attack is not a new concept, but it has gained focus in a cloud environment because of the resources like cache are shared among multiple users. This paper proposes a novel approach to ensure VM security against cache-based attacks in the cloud environment. A fuzzy controller is used for detecting and mitigating the cache-based side channel attack in a cloud environment. The fuzzy controller makes decisions whether the user is an attacker or a legitimate user. Eight different fuzzy rules formed to categorize the attackers and genuine user. Fuzzy rules are formed from the input values like cache data access, the size of the cache data access, cache miss ratio, virtual CPU utilization and virtual memory utilization. The experimental results prove that the proposed approach reduces the false positive ratio to identify the true attacker and genuine user. The proposed approach could attain a maximum of 78.15%, 80.04%, and 82.16% of accuracy at 10, 20 and 30 sec. of time intervals. However, the proposed approach detects the cache-attack on last level of cache, and introduces only 1% of performance overhead which is negligible. However, in future, the proposed approach is planning to add more fuzzy based rules to prevent all levels of cache attacks as this work is applicable only to last level cache.

REFERENCES

1. K.H. Lee , S.W. Kwon , J.H. Shin , D.H. Kim , D.Y. Moon, & S.Y. Lee. (2016). Resource Allocation Methodology of Virtual Desktop Infrastructure in Cloud Computing BIM: Focusing Korean Small and Medium Sized Architectural Design Firm. *In proceedings of 33rd International Symposium on Automation and Robotics in Construction (ISARC)*(pp 694-704). Auburn, USA.
2. Ainapure B.S., Shah D., & Rao A.A. (2018). Understanding Perception of Cache-Based Side-Channel Attack on Cloud Environment. *In: Sa P., Sahoo M., Murugappan M., Wu Y., Majhi B. (eds) Progress in Intelligent Computing Techniques: Theory, Practice, and Applications.* Advances in Intelligent Systems and Computing (vol 719. pp 9-21), Springer, Singapore.
3. J. Kelsey, B. Schneier, D. Wagner, & C. Hall (2000). Side Channel Cryptanalysis of Product Ciphers. *Journal of Computer Security*, Vol(8, n. 2-3), 141-158.
4. Michael Godfrey, Mohammad Zulkernine. (2015). Preventing Cache-Based Side-Channel Attacks in a Cloud Environment. *IEEE Transaction on Cloud Computing,* Vol(2, no. 4), 395-408.
5. Z. Wu, Z. Xu, & H. Wang. (2012). Whispers in the hyper-space: High-speed covert channel attacks in the cloud. *In proceedings of 21st USENIX conference on security symosium*. (Pp. 9–9), Berkely.
6. Jicheng Shi, Xiang Song, Haibo Chen, & BinyuZang. (2011) Limiting cache-based side-channel in multi-tenant cloud using the dynamic page coloring. *In proceedings of International conference on Dependable systems and Network workshops. (DSN-W)* (pp. 194-199). IEEE.
7. Si Yu, XiaolinGui, & Jiancai Lin. (2013). An approach with two-stage mode to detect cache-based side channel attacks. *In proceedings of International conference on Information Networking*. (Pp. 186-191). IEEE, USA.
8. M. Chiappetta, E. Savas, & C. Yilmaz. (2015) Real time detection of cachebased side-channel attacks using hardware performance counters. *Cryptology ePrint Archive:Report 2015/1034*. Retrieved from http://eprint.iacr. org/.

9. J. Pfoh, C. Schneider, & C. Eckert. (2011). Nitro: Hardware-based System Call Tracing for Virtual Machines. *In proceedings of 6th International Workshop on Security IWSEC*. (Pp. 96–112). Springer.
10. Dinaburg A., Royal, P., Sharif, M., & Lee. W. (2008). Ether: malware analysis via hardware virtualization extensions. *In Proceedings of the 15th ACM conf. On Computer and communications security.* (Pp. 51-62). ACM, New York, NY, USA.
11. Bryan D. Payne. (2012). Simplifying Virtual Machine Introspection Using LibVMI. *Technical report. Sandia National Laboratories No. SAND* 2012-7818.
12. T. Ristenpart E. Tromer H. Shacham, & S. Savage. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. *In Proceeding of 16th ACM conference CCS.* (pp. 199–212). ACM, New York.
13. Acıiçmez O., Brumley B. & Grabher P. (2010). New results on instruction cache attacks. *In Proceedings of 12th international conference on Cryptographic hardware and embedded systems (CHES'10)*. (Pp. 110-124). Springer-Verlag Berlin.
14. Aciiçmez, O., Koç, Ç. & Seifert, J. (2007). On the power of simple branch prediction analysis. *In Proceedings of the 2nd ACM symposium on Information, computer and communications security (ASIACCS '07)*. (Pp. 312-320).
15. Yuval Yarom, & Katrina Falkner. (2014). FLUSH+RELOAD: a High Resolution Low noise, L3 cache Side-Channel attack. *In proceedings of 23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association. (Ppp. 719-732). San Diego, CA
16. Osvik D., Shamir A. & Tromer E. (2006). Cache attacks and countermeasures: the case of AES. *In Topics in Cryptology–CT-RSA. Lecture Notes in Computer Science*. Vol(3860)(Pp. 1–25). Springer, Berlin, Heidelberg.
17. Liu, F. & Lee R.B. (2013). Security testing of a secure cache design. *In Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy (HASP '13).* (Pp. 1-8). ACM Press. New York. USA.
18. Zimmermann HJ. (1991). *Introduction to Fuzzy Sets. Fuzzy Set Theory — and Its Applications.* Springer. Dordrecht
19. Z. Wu, Z. Xu, & H. Wang. (2012). Whispers in the hyper-space: High-speed covert channel attacks in the cloud. *In Proceedings of the 21st USENIX Conference on Security Symposium, ser. Security'12*. (Pp. 9–9). USENIX Association
20. Anne Canteaut, Cedric Lauradoux, & Andre Seznec. (2006). Understanding cache attacks. *Technical Report.* Available at: ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-5881.pdf
21. Munish Chouhan & Halabi Hasbullah (2016). Defence Against Cache-Based side channel attacks for secure cloud computing. *ARPN Journal of Engineering and Applied Science.,* Vol(11, No. 22). (Pp. 13146-13152).
22. D. Nurmi *et al.* (2009). The Eucalyptus Open-source Cloud-computing System. In proceeding of *9th IEEE Int. Symp. Clust. Comput. Grid (CCGrid 09). (Pp. 709-736).* Shanghai, China.
23. B. Aires. (2008). *Hardware Assisted Virtualization Intel Virtualization Technology*. Retrived http://linux. linti.unlp.edu.ar/images/f/f1/Vtx.pdf
24. C. Rodrigo, N. Calheiros1, Rajiv Ranjan, Anton Beloglazov, A. F. De Rose, & R. Buyya. (2010). CloudSim: a toolkit formodeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Journal of Software—Practice & Experience. Vol(41, Issue 1). (Pp. 23-50).* John Wiley & Sons, Inc. New York, NY, USA.

Figures

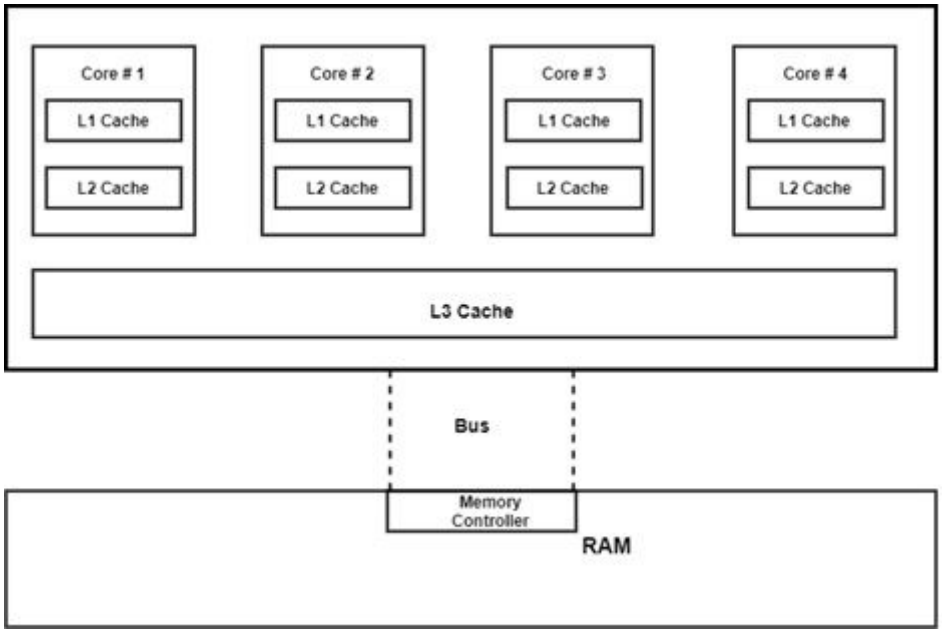

Figure 1

Cache memory hierarchy

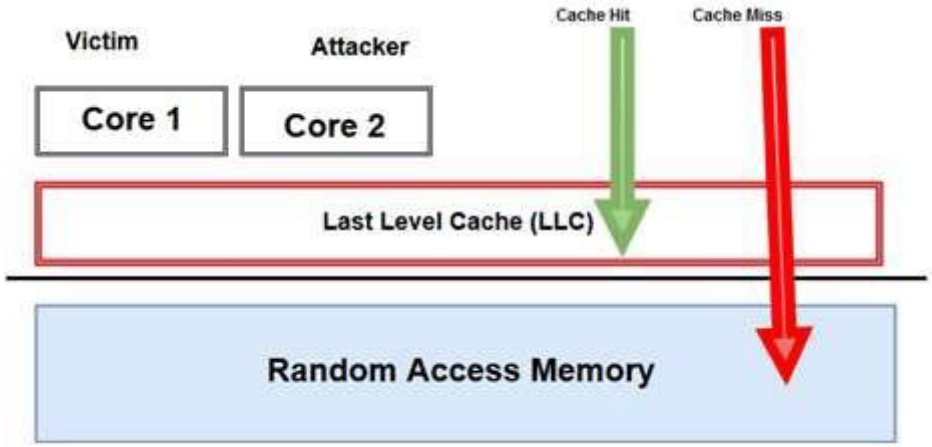

Figure 2

Cache based attack on last level cache

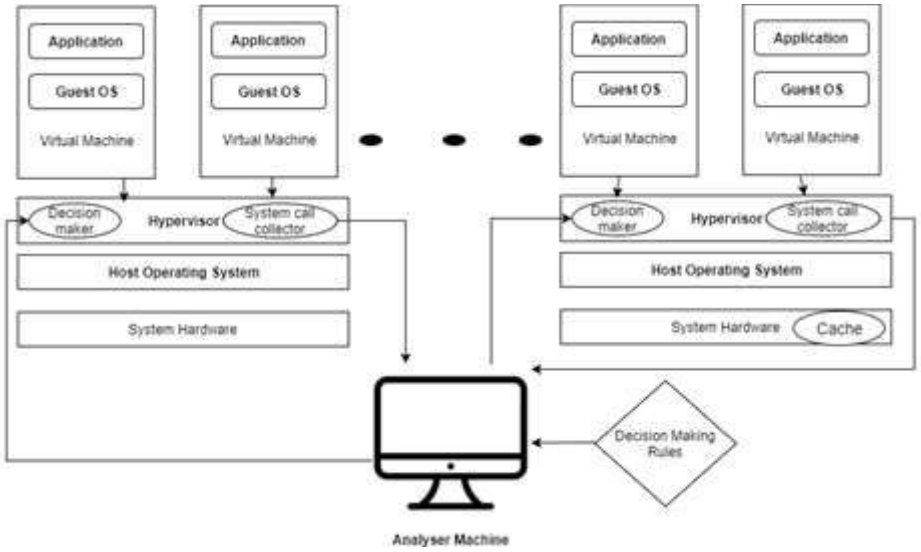

Figure 3

System Architecture for proposed system

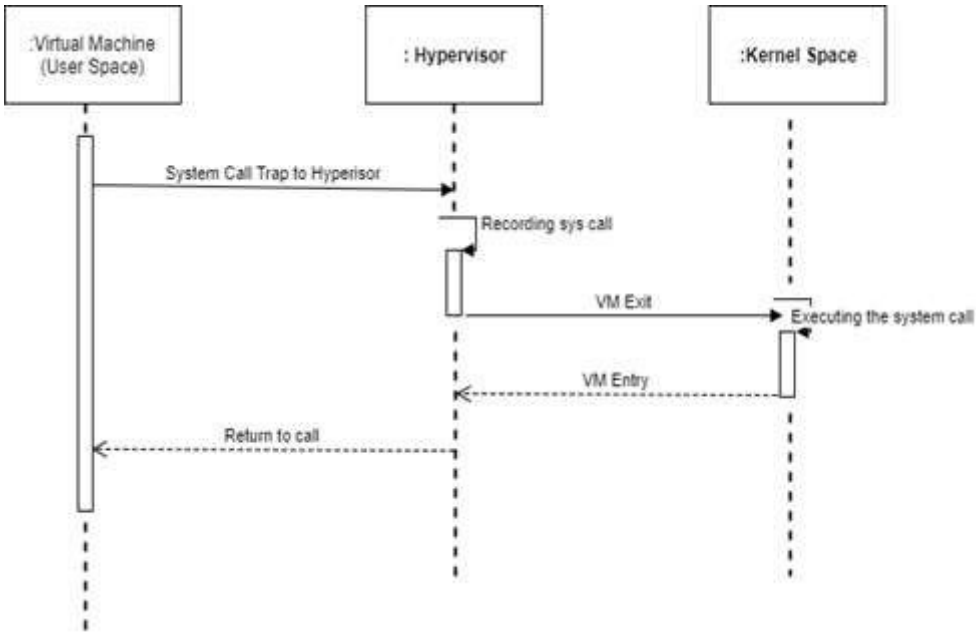

Figure 4

Sequence to track the system call

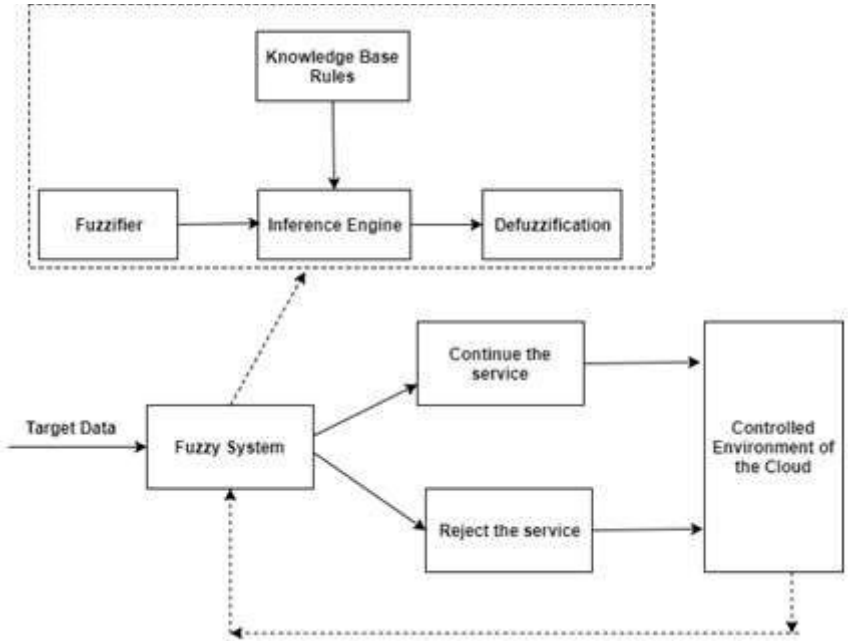

Figure 5

Fuzzy controller system

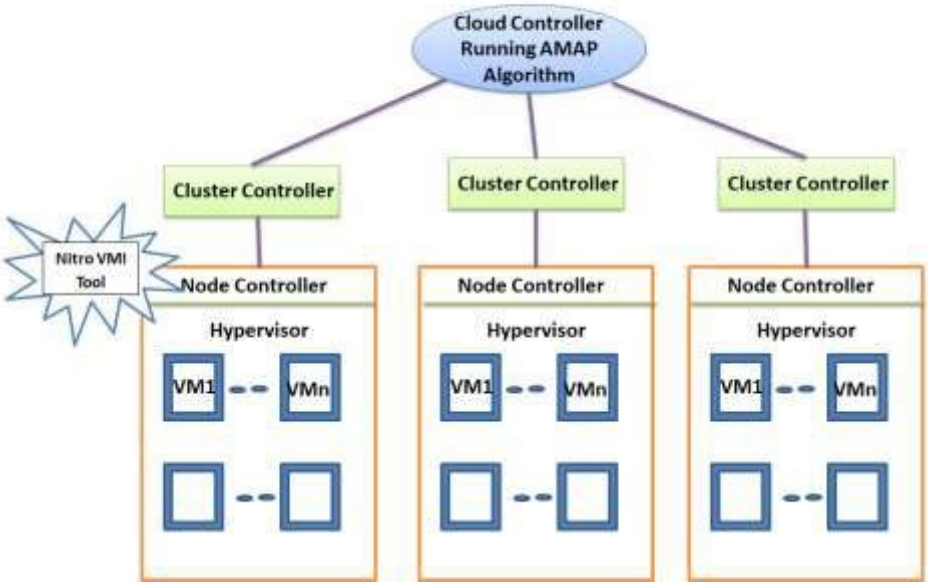

Figure 6

Experimental setup for private cloud

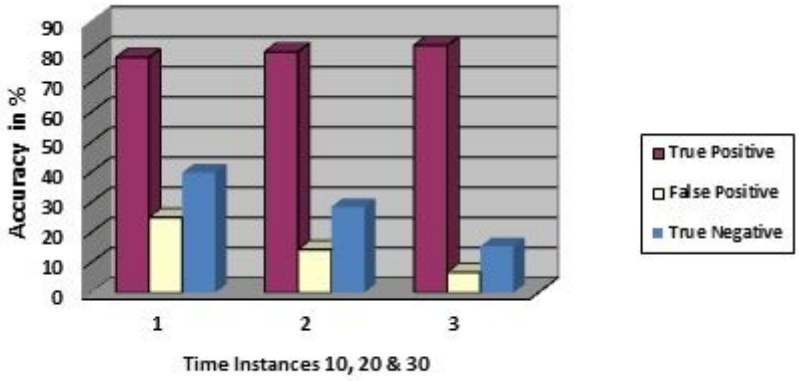

Figure 7

Accuracy graph of proposed system

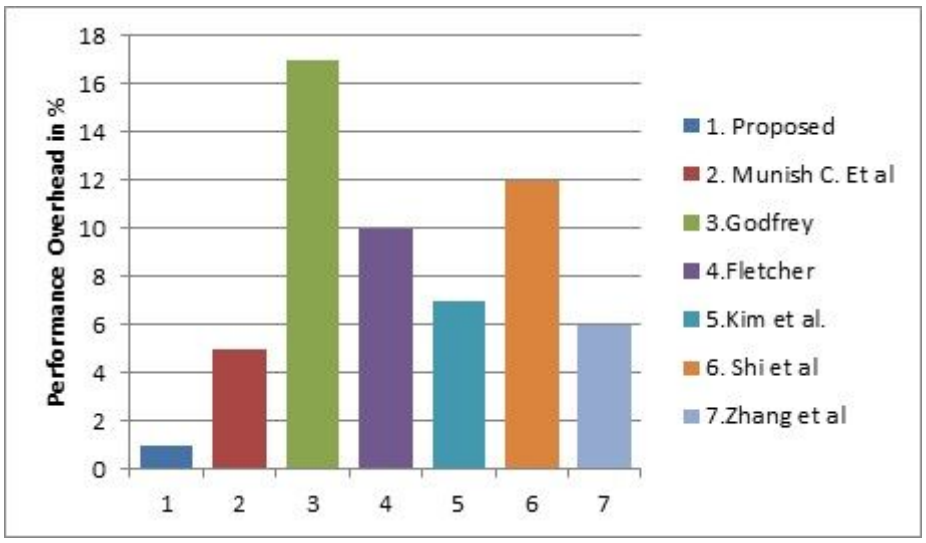

Figure 8

Analysis of performance overhead [21]