A Security Framework for Improving QoS by Detecting and Mitigating Cache Side-Channel Attacks in Virtualized Environments

S.Mahipal and V. Ceronmani Sharmila

Abstract--**Virtualization technology makes cloud resources affordable. Virtual Machines (VMs) are widely used on top of host machines in cloud computing environments. Adversaries target L3 cache memory in such environments as VMs shared L3 cache of the physical machine. Cache side-channel attacks are possible unless there are mechanisms in place to prevent them. The attacker compromises a VM and through which tries to extract sensitive information from the victim's machine. In the literature, many solutions are found to resist cache side-channel attacks. However, an important limitation of the existing approaches is that they are not real-time in nature. In this paper, we proposed a security framework for improving Quality of Service (QoS) by detecting and mitigating cache side-channel attacks in virtualized environments. Two algorithms are proposed to realize the framework. They are known as Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM) and Intelligent Noise Addition for Attack Mitigation (INA-AM). The first algorithm takes care of monitoring and detecting cache side-channel attacks and notifying administrators while the second one is responsible to mitigate the effects of attacks. A machine learning technique assists the SFML-SCAM in detecting attacks based on the abnormality in the CPU counters in real-time. The INA-AM algorithm, on the other hand, has a strategy to add noise to the cache data to deceive the attacker and mitigate the effects of the attack. Experiments are made with open-source tools along with a prototype that implements the proposed algorithms. Information leakage is evaluated for all detection methods against stealth attacks. Experimental results revealed that the proposed framework outperforms the state-of-the-art.**

Index Terms – Cloud computing, Virtual Machine, Side-Channel Attacks, VM Security Framework, Side Chanel Attack Prevention

I. INTRODUCTION

Cloud computing technology enabled a new paradigm shift in computing, storage, and associated services. With virtualization technology, cloud computing has become an affordable platform for sharing computing resources. However, the QoS depends on the smooth functioning of VMs in virtualized environments. There are many security vulnerabilities associated with VMs, hypervisors, and VM migration tasks. Cache side-channel attacks associated with virtualization are a major concern as they can deteriorate QoS performance in cloud environments.

Manuscript received September 11, 2021; revised February 14, 2023.

S. Mahipal is a Research Scholar at the Hindustan Institute of Technology and Science, Chennai, 603103, India. e-mail id: srmahipal@gmail.com.

V. Ceronmani Sharmila is a professor and HOD in the IT department of Hindustan Institute of Technology and Science, Chennai, 603103, India. e-mail id: csharmila@hindustanuniv.ac.in.

Cache side-channel attacks extract secret information by monitoring the cache behavior of a victim. Saeed et al. [4] proposed a methodology to demonstrate the cross-VM network channel attack. With impersonation and mirroring, they could show the vulnerabilities in VM usage in the cloud.VM migration is used to manage resources efficiently. However, it is vulnerable to attacks unless secured. Tao et al. [3] focused on VM management in edge computing environments. They investigated VM migration issues in such scenarios. They found that VM migration in edge computing has vulnerabilities to security attacks. As edge computing is evolving, it is indispensable to leverage security for VM migration in such an environment. Cleemput et al. [21] investigated timing side-channel attacks and proposed a method known as the adaptive compiler strategy for mitigating such attacks. It not only provides security but also reduces system overhead. Mukhtar et al. [23] proposed a countermeasure to prevent cache side-channel attacks. The countermeasure is based on flush + prefetch. Sangeetha and Sumathi [24] proposed a measure that counts processor cycles and memory utilization to detect side-channel attacks. Chiappetta et al. [25] used hardware performance counters to ascertain the presence of side-channel attacks. From the literature, it is understood that there are numerous approaches found to detect and mitigate side-channel attacks. However, there is a need for a real-time approach that can detect side-channel attacks as they occur and provide necessary communication and mitigation steps. Our contributions are as follows.

1. We have built a security framework that takes care of the detection and mitigation of cache side-channel attacks in virtualized environments.

2. We proposed two algorithms namely Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM) and Noise Addition for Attack Mitigation (INA-AM) to detect and mitigate VM side-channel attacks respectively.

3. A prototype is used along with the usage of open-source existing tools appropriately to demonstrate proof of the concept.

The remainder of the paper is structured as follows. Section 2 reviews the literature on different kinds of attacks on VM and countermeasures. Section 3 discusses cache side-channel attacks. Section 4 presents the proposed framework to detect and mitigate cache side-channel attacks. Section 5 provides experimental results reflecting the performance of the proposed algorithms. Section 6 throws light on performance evaluation. Section 7 provides a discussion about the performance of the proposed methodology. Section 8 concludes the paper and gives scope for future work.

II. RELATED WORK

VMs usage in cloud computing throws them to different security challenges. Saeed et al. [4] proposed a methodology to demonstrate the cross-VM network channel attack. With impersonation and mirroring, they could show the vulnerabilities in VM usage in the cloud. Bazmet al. [5] used different performance counters and cache monitoring techniques to detect cache side-channel attacks. However, it seems to lack a real-time approach. Yan et al. [6] investigated the structure of cache memory used by virtualized environments and understood the probabilities of attacks such as side-channel attacks like Evict+Reload and Prime+Probe. Liu et al. [7] investigated the cache side-channel attacks and proposed a secure cache architecture. It is made dynamic with a line number mapper and other components of the hardware. They recommend processor manufacturers come up with security inbuilt. Liu et al. [8] considered performance optimization features to defeat Last Level Cache (LLC) side-channel attacks. Anwar et al. [9] made reviewed different solutions to VM side-channel attacks and cross-VM side-channel attacks. Fiore et al. [10] proposed a method using CAT-enabled CPUs and optimization mechanisms in the partitioning of LLC to prevent cache side-channel attacks.

VM migration is used to manage resources efficiently. However, it is vulnerable to attacks unless secured. Mahfouz et al. [1] proposed a method for the live migration of VM with security. They are proposed real-time runtime monitors that observe the migration process and its critical aspects to protect from attacks. Their methodology involves different phases such as the setup stage, memory transfer stage, storage transfer stage, and network clean upstage. The three categories of threats considered include control plane class, data plane class, and migration module class. Their methodology is theoretical and implementation is yet to be done. Li et al. [2] proposed energy-aware dynamic virtual machine consolidation (EC-VMC) which is meant for limiting VM migration and enhancing the energy efficiency of data centers for QoS improvements. Tao et al. [3] focused on VM management in edge computing environments. They investigated VM migration issues in such scenarios. They found that VM migration in edge computing has vulnerabilities to security attacks. As edge computing is evolving, it is indispensable to leverage security for VM migration in such an environment.

Jiaet al. [11] proposed a secure VM allocation strategy to avoid problems associated with VM co-residence. It has optimization objectives such as energy efficiency, load balancing, and security. A similar kind of work is carried out in [13], [16], and [20]. Mushtaq et al. [12] proposed a tool for the automatic detection of side-channel attacks in virtualization environments. The tool is named WHISPER which could detect attacks such as meltdown, specter, prime+probe, flush+flush, and flush+reload. Liu et al. [14] explored side-channel attacks in IoT-enabled environments for computation offloading where GPU virtualization is used. Yan et al. [15] proposed a framework to defend against cache-based side-channel attacks using a policy known as "Secure Hierarchy-Aware Cache Replacement Policy (SHARP)". Sangakkara et al. [17] investigated electronic side-channel attacks that are used as digital forensics for evidence recovery. Mushtaq et al. [18] proposed a machine learning-based solution for security against side-channel attacks. Yang et al. [19] defined an approach where switching and migrating of multi-executor VMs is done to mitigate side-channel attacks.

Cleemputet al. [21] investigated timing side-channel attacks and proposed a method known as an adaptive compiler strategy for mitigating such attacks. It not only provides security but also reduces system overhead. Shin et al. [22] proposed a technique for analyzing cache side-channel and inferring firewall rules required to prevent attacks. Mukhtar et al. [23] proposed a countermeasure to prevent cache side-channel attacks. The countermeasure is based on flush + prefetch. Sangeetha and Sumathi [24] proposed a measure that counts processor cycles and memory utilization to detect side-channel attacks. Chiappetta et al. [25] used hardware performance counters to ascertain the presence of side-channel attacks. Other important researches found in the literature include novel denial of service (DoS) attacks [26], libraries for secure VM placement [27], energy-aware VM allocation [28], energy-aware adaptive cat swarm optimization [29], and dynamic VM allocation strategy [30]. Mahipal and Sharmila [37] explored VM security issues and countermeasures. Dhavlle et al. [38] focused on lessening side channel leakage issues. Dutta et al. [39] investigated the cross-component covert channels. Eliyan et al. [40] studied the security issues with DDoS attacks while Ranaweera et al. [41] focused on privacy and security in edge computing. From the literature, it is understood that there are numerous approaches found to detect and mitigate side-channel attacks. However, there is a need for a real-time approach that can detect side-channel attacks as they occur and provide necessary communication and mitigation steps.

III. CACHE SIDE CHANNEL ATTACKS

This section provides details of different categories of side-channel attacks that are considered in this paper for detection and mitigation while improving the state of the art. Different kinds of side-channel attacks in virtualized environments are explored in Yarom[31]. For convenience, they are categorized here.

Table I: Shows different categories of cache side-channel attacks in virtualized environments

Category	Side Channel Attack Name	Target
Category 1	Flush+ReloadAttcak	Aimsat L3 Cache
Category 2	Flush+ Flush Attack	Aims at L3 Cache
Category 3	Prime+Probe Attack	Aims at L3 Cache

As presented in Table 1, the side-channel attacks are mapped to different categories, and Sections 3.1 through Section 3.3 throw light on each category. The overview of the problem context and the associated attack model is shown in Fig 1. The physical host machines used by cloud data centers have their main memory, and secondary memory in the form of disks, network cards, and other physical hardware. In modern CPU architectures like Intel x86 multi-core, L1 and L2 cache are associated with each core. Whereas the L3 cache which is known as Last Level Cache (LLC) is shared across the cores. The Virtual Machine Monitor (VMM) of the hypervisor takes care of the creation and management of VMs. Each VM can have a virtual environment similar to a physical machine with Operating System (OS) and applications run by users.

Adversaries can launch Cache side-channel attacks on LLC due to its cache-inclusive property that leads to information leakage through cache memory to the attacker (essentially a user of an application running in VM). In cloud computing environments usage of shared cache memory is common as it helps in the reduplication of data and serves as the memory that reduces memory usage. LLC memory is monitored by the attacker to obtain sensitive information leading to the potential risk to the victim.

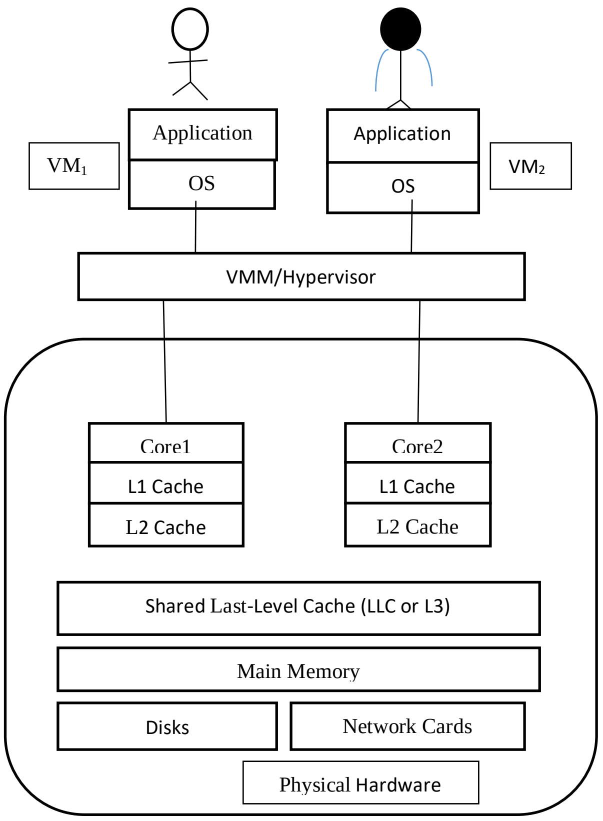

Fig 1: Illustrates cache side-channel attack in the virtualization environment

A. Side-Channel Attack Category 1

The Flush+Reload attack aims at LLC. The modus operandi of the attack has three steps. The first step is known as FLUSH where clflush is the command used by the attacker to flush the specific shared cache line. The second step is known as the IDLE step. In this step, the attacker waits for a certain period while the victim runs different operations that might include sensitive information. The third step is known as RELOAD. In this step, the attacker reloads the cache line and observes how much time it takes. If it is taking more time, it does mean that the victim did not use sensitive data from the shared page. If the time taken is less, it does mean that there is a victim's data filled in the shared page. Thus the time difference strategy is used by the attacker to ascertain access patterns of the victim for sensitive data in LLC. This kind of attack is said to be the highest resolution of the side-channel attack which leads to the extraction of secret keys of cryptographic primitives and also captures keystroke information of the victims. As the attack makes use of clflush, it leads to an increase in the LLC, L1, and L2 miss when the victim tries to access the cache lines available.

B. Side-Channel Attack Category 2

This category of attack (Flush+Flush) targets the shared cache memory or L3 cache. Instead of using the time difference between cache misses and cache hits after giving the clflush command by the attacker (as in category 1), the category 2 attack makes use of the time difference between two clflush commands. Since clflush commands work faster than general memory access instruction it does not lead to LLC, L1, and L2 cache miss or hit. Hence the category 2 attack is faster and stealthy. However, it is not easy to notice the time difference between two clflush commands. As investigated in Yarom[31], the category 2 attack exhibits less accuracy. This attack has three steps namely FLUSH, IDLE, and FLUSH. The first two steps are similar to that of category 1 while the third step differs. In the last FLUSH step, the attacker estimates the time taken by the clflush command. If the time is more, it indicates that the victim has used sensitive data or a probing cache line. On the other hand, if the time taken is less it does mean that the victim did not use the probing cache line.

C. Side-Channel Attack Category 3

The category 3 attack is known as the Prime + Probe attack. It targets the LLC cache as it has memory shared between cores. There is no need for the attacker to prepare shared memory as LLC has shared memory. It is, therefore, the attack that is more used by the attacker. It is characterized by a lower resolution when compared with category 1 and category 2 attacks. To share the cache, set, an attacker makes an eviction set that helps in sharing a cache set between the attacker and the victim. The attacker probes all the lines of the eviction set to know where sensitive data is accessed by the victim. This attack has three steps namely PRIME, IDLE, and PROBE. In the first step, an attacker fills sets with data. In the second step, the attacker waits a certain period while the victim runs certain commands that may be sensitive. In the third step, the probe operation caches the sets with the data that has been prepared. The attacker measures probing time. If the time is more or if there is a change in the eviction set, it does mean that the cache set is used by the victim while evicting some cache lines associated with cache sets.

IV. THE PROPOSED FRAMEWORK

The proposed framework for VM security has two phases namely a) Monitoring and b) Mitigation. In the first phase, the framework uses the prime and probe technique to analyze CPU counters data to identify an attack. In the second phase, noise is added to the cache data to deceive the attacker and mitigate the effects of the attack. In the meanwhile, it alerts the administrator about the attack for making further decisions. The proposed solution has two algorithms implemented. They are known as Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM) and Intelligent Noise Addition for Attack Mitigation (INA-AM). The first algorithm takes care of monitoring and detecting cache side-channel attacks and notifying the administrator while the second one is responsible to mitigate the effect of an attack. Figure 1 shows the overview of the proposed security framework. In the monitoring phase, we used the prime and probe techniques as part of the methodology to detect side cache channel attacks. PAPI (Performance Application Programming Interface) library is used to obtain information about CPU counters and the data is saved. CSV file from time to time. Mohammad-Mahdi [34] tool is used to make different kinds of side-channel attacks. Such scenarios are captured using PAPI and the dataset is generated and saved.

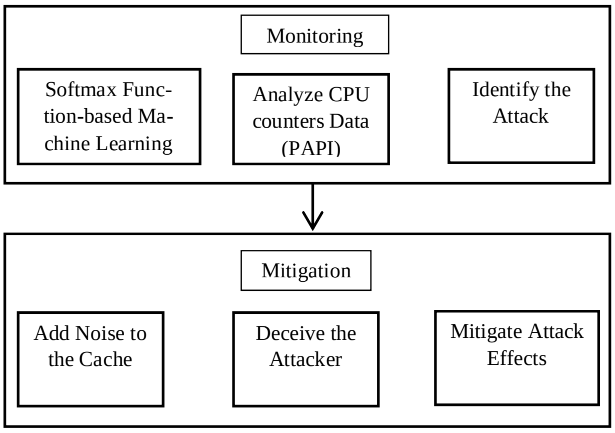

Fig 2: Proposed security framework for improving QoS by detecting and mitigating cache side-channel attacks in virtualized environments

For identification of the attack, a machine learning approach known as multi-label classification is used. Unlike binary classification which supports only two classes, it supports three or more class labels. The supervised machine learning approach used for the detection of attacks is known as the softmax classification model. It makes use of linear regression and the resultant values are subjected to normalization probabilities. Softmax classifier uses both exponentiation and normalization as expressed in Eq. 1.

$$P_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \; for \; i = 1,2,\ldots\ldots..,k. (1)$$

There are different probabilities used in the softmax classification. The probabilities of the given label when added, the sum should not exceed 1. The probability, denoted as p_i, is computed and issued in the classification model. The more in probability, the higher the possibility to belong to a particular class. One hot encoding is used to make one input true for two or more inputs the largest probable value is 1. The softmax function also involves cross-entropy loss that helps in reducing the error between predicted and actual values. The underlying machine-learning model is shown in Figure 3.

The softmax classification model learns from the data and detects different cache side-channel attacks. It not only detects the presence of a side-channel attack but also classifies

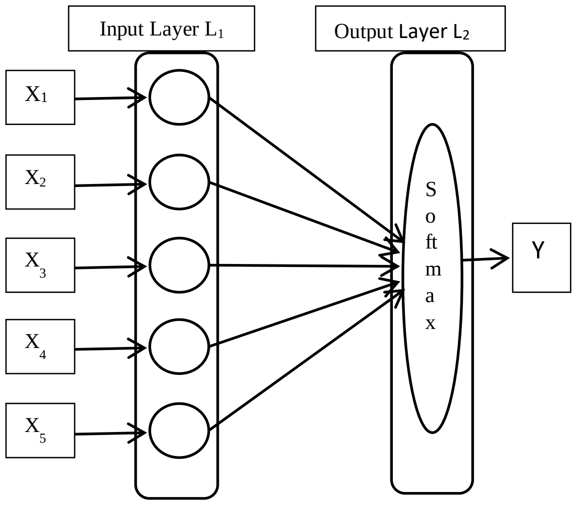

Fig3: Softmax based prediction model for attack detection

the attacks. Attack categories discussed earlier are known as category 1, category 2, and category 3. These class labels denote specific attacks and they are denoted as A1, A2, and A3 respectively while the normal situation is labeled as A0. The ML model used is a single-layer perception (neural network model) which has no hidden layers. The dataset used for experiments is associated with input units. The predicted labels are denoted by the output unit. The performance counters associated with the processor are recorded at runtime using the PAPI library and the data is generated in the form of. CSV file that is reused for the detection of attacks. The detection process works faster to have real-time performance in this paper, unlike the existing methods. The classifier is trained with several thousands of training samples to gain the required prediction knowledge. After training, the model is equipped with intelligence to predict different kinds of attacks. Tensor Flow, a framework from Google, is used for realizing softmax classification as part of our empirical study. Table 2 shows the parameters used to configure Tensor Flow.

Table II: TensorFlow configuration details

TensorFlow Parameter	Value
Learning rate	0.1
Number of epochs	100
Batch size	100

The detection model is implemented using the Python data science platform. The algorithm proposed for the detection of cache side-channel attacks is known as Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM).

A. Attack Detection Algorithm

SFML-SCAM algorithm provides the procedure followed to detect cache side-channel attacks in real-time. It takes. CSV file as input where performance counters data is available. This data is used to detect different kinds of attacks. The data is subjected to softmax classification.

Algorithm 1: Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM)
Input: Training Data D, Test Data T
Output: Detection Results R

```
1.    Start
2.    Initialize training data features vector F1
3.    Initialize test data features vector F2
4.    F1←Extract Features(D)
5.    F2←Extract Features(T)
```

```
6.    model←Train Classifier(F1, Softmax Classifier)
7.    R←Predict(model, F2)
8.    For each result r in R
9.    IF r=A1 Then
10.   Attack category 1 is detected
11.     Else If r=A2 Then
12.       Attack category 2 is detected
13.     Else If r=A3 Then
14.       Attack category 3 is detected
15.     End If
16.   End For
17.   Notify the presence of an attack
18.   Return R
19.   End
```

Algorithm 1: Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring

As presented in Algorithm 1, it takes training data and testing data as inputs and produces attack detection results. For extracting features from the data, it initialized two vectors known as F1 and F2 as in Step 2 and Step 3 respectively. In Step 3 and Step 4, it populate features of training and testing datasets into F1 and F2 vectors respectively. In Step 5, a model is trained using a Softmax classifier based on the features extracted from the training set. In Step 5, test set features are extracted. A model is fit in Step 6 to have a prediction system with the required intelligence. Then in Step 6, prediction is carried out on F2 which has training data features. The prediction results are saved to the vector R. Then from Step 8 through Step 16, an iterative process is carried out to identify the presence of an attack and know which kind of attack is in each instance of the results vector R. Thus it finds different classes of attacks namely A1, A2, and A3. Finally, the algorithm returns the detection results besides notifying the administrator about the presence of the attack.

B. Attack Mitigation Algorithm

An algorithm known as Intelligent Noise Addition for Attack Mitigation (INA-AM) is defined to mitigate the effect of different kinds of side-channel attacks. As presented in Algorithm 2, it is evident that it takes cache hits and cache

```
Algorithm: Intelligent Noise Addition for Attack Mitigation (INA-AM)
Input: Cache Hits H, Cache Misses M
Output: Noise Cache Hits H', Noise Cache Misses M'

1.    Start
2.    Initialize noisy cache hits vector H'
3.    Initialize noisy cache misses vector M'
4.    hcount←CountCacheHits(H)
5.    mcount←CountCacheMisses(M)
6.    noisefunction←ComputeNoiseFunction(H, M)
7.    For each cache hit h in H
8.    IF the noise function recommends noise to h Then
9.    Add noise to h
10.       Add h to H'
11.     End If
12.   End For
13.   For each cache miss m in M
14.     IF the noise function recommends noise to m Then
15.       Add noise to m
16.       Add h to M'
17.     End If
18.   End For
19.   Output H'
20.   Output M'
21.    End
```

Algorithm 2: Intelligent Noise Addition for Attack Mitigation algorithm

misses vectors as inputs and returns the noisy cache hits and noisy cache misses to confuse the attacker to mitigate the effect of cache side-channel attacks. It initializes two vectors named H' and M' for holding noisy cache hits and noisy cache misses in Step 2 and Step 3 respectively. As the adversary depends on the observation of low latency of cache hits and high latency of cache misses, it is essential to have an intelligent noise function to confuse the attacker. The count of cache hits and cache misses are computed in Step 4 and Step 5. Step 6 computes the noise function which guides in adding noise as needed. Based on the noise function an iterative process from Step 7 through Step 12 is used to add noise to cache hits. In the same fashion, another iterative process from Step 13 through Step 18 is used to add noise to cache misses. Finally, the algorithm outputs the noisy hits and misses vectors to ensure that the attacks will not be successful.

V. EXPERIMENTAL RESULTS

Different configurations of VMs are used in the experimental study. The CPU code names are mapped to simplified configuration names for convenience. The original name of the configuration mapped name is presented in Table 3. Experiments are made with open-source tools along with a prototype that implements the proposed algorithms. Experimental results revealed that the proposed framework outperforms the state-of-the-art. Observations are made in terms of detection rate (%), time (seconds), and CPU usage (%).

Table III: Mapping original CPU code names to configuration names for convenience

Original CPU Code Name	Configuration Name
Intel Xeon E5-2620 v4 2.10 GHz (Broad well)	Configuration 1
Intel Xeon E3-1275v6 3.80 GHz (Kaby Lake)	Configuration 2
Intel Core i5-7400 3.00Ghz (Kaby Lake)	Configuration 3
Intel Core i7-7700 3.60GHz(Kaby Lake)	Configuration 4
Intel Core i7-9700 3.60 GHz (Coffee Lake)	Configuration 5
Intel Core i5-5250U 1.6GHz (Broad well)	Configuration 6

Results are observed on normal mode and stress model where the stressing tool is used to incur stress on components like I/O, memory, and CPU. Experiments are also made in virtualized and non-virtualized environments.

A.Attack Detection Rate

Attack detection rate is compared between virtualized and non-virtualized environments. Experiments are made with the six configurations.

Table IV: Detection rate performance for two environments in normal and stress modes

CPU Configuration	Attack Detection Rate (%)			
	VM Env (Normal)	**VM Env (Stress)**	**Non VM Env (Normal)**	**Non VM Env (Stress)**
1	0.984984	0.925925	1	0.95095
2	0.971971	0.919919	0.9981	0.941941
3	0.962962	0.929929	0.9961	0.942942
4	0.987987	0.923923	0.992099	0.943943
5	0.945945	0.929929	0.9951	0.956956
6	0.97097	0.928928	1	0.954954

As presented in Table 4, the attack detection rate is presented in virtualized and non-virtualized environments using normal and stress models against different CPU configurations.

As presented in Figure 4, the CPU configurations are provided in the horizontal axis while the vertical axis shows the detection rate (%). A higher detection rate indicates better performance for the proposed method. The highest performance is 1 indicating 100% detection rate. It is observed that different CPU configurations have exhibited varying detection rates. As performance monitors are associated with CPU monitors, it is bound to vary for each configuration. There are differences in detection rates in virtualized and non-virtualized environments. In the same fashion, there are performance differences between normal and stress modes against all six configurations.

B. Execution Time

Execution time is compared between virtualized and non-virtualized environments. Experiments are made with the six configurations for normal and stress modes.

Table V: Execution time performance for two environments in normal and stress modes

CPU Configuration	Time (seconds)			
	VM Env (Normal)	**VM Env (Stress)**	**Non VM Env (Normal)**	**Non VM Env (Stress)**
1	1.9019	2.1042	1.5015	1.8018
2	1.9019	2.2044	1.6016	1.9019
3	1.8018	2.3046	1.6016	1.9019
4	1.9019	2.2044	1.7017	1.9019
5	1.7017	2.4048	1.5015	1.8018
6	1.8018	2.2044	1.5015	1.8018

As presented in Table 5, the execution time is presented in virtualized and non-virtualized environments using normal and stress models against different CPU configurations. As presented in Figure 5, the CPU configurations are provided on the horizontal axis while the vertical axis shows the execution time (seconds). Lower execution time indicates better performance for the proposed method. It is observed that different CPU configurations have exhibited varying execution times. As performance monitors are associated with CPU monitors, it is bound to vary for each configuration. There are differences in execution time in virtualized and non-virtualized environments. In the same fashion, there are performance differences between normal and stress modes against all six configurations.

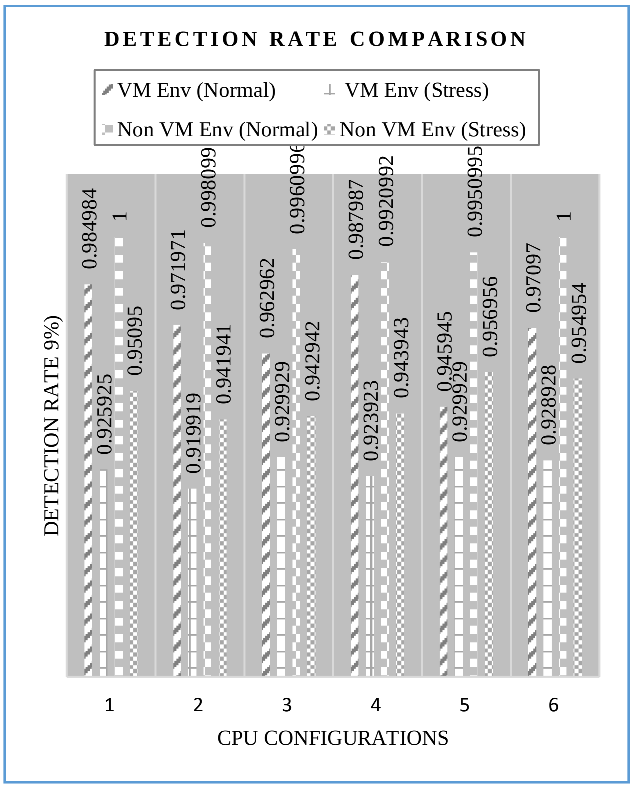

Fig4: Detection rate comparison for virtualized and non-virtualized environments using normal and stress modes

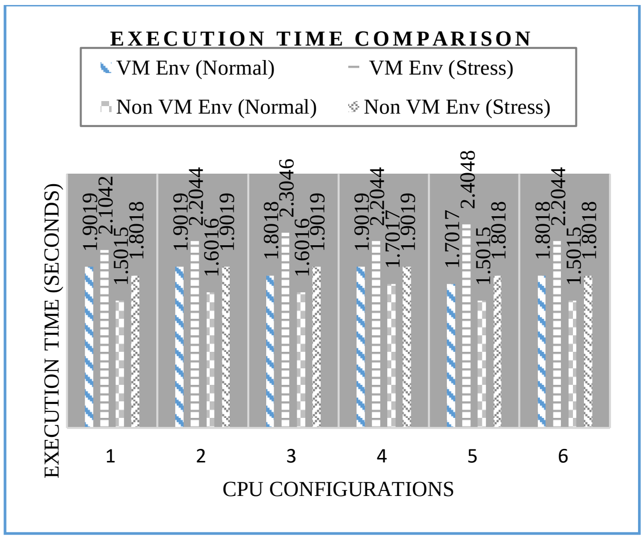

Fig 5: Execution time comparison for virtualized and non-virtualized environments using normal and stress modes

C. CPU Usage

CPU usage is compared between virtualized and non-virtualized environments. Experiments are made with the six configurations for normal and stress modes.

Table VI: CPU usage performance for two environments in normal and stress modes

CPU Configuration	CPU Usage (%)			
	VM Env (Normal)	VM Env (Stress)	Non VM Env (Normal)	Non VM Env (Stress)
1	0.61	0.91	0.71	0.91
2	0.83	0.91	0.71	0.9
3	0.72	0.81	0.61	0.82
4	0.82	1.11	0.71	0.91
5	0.91	1.11	0.72	0.89
6	0.91	0.81	0.63	0.92

As presented in Table 6, the CPU usage is presented in virtualized and non-virtualized environments using normal and stress models against different CPU configurations.

As presented in Figure 6, the CPU configurations are provided on the horizontal axis while the vertical axis shows the CPU usage (%). Lower CPU usage indicates better performance for the proposed method. It is observed that different CPU configurations have exhibited varying usage of CPU. As performance monitors are associated with CPU monitors, it is bound to vary for each configuration. There are differences in CPU usage in virtualized and non-virtualized environments. In the same fashion, there are performance differences between normal and stress modes against all six configurations.

VI. PERFORMANCE EVALUATION

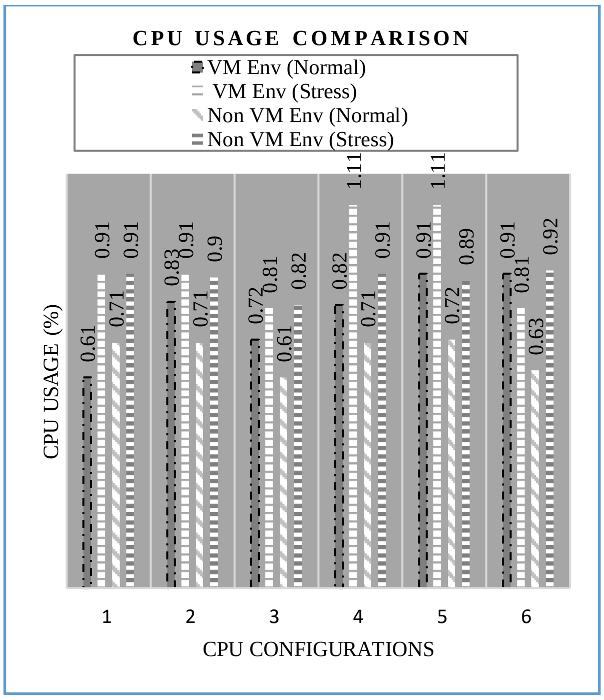

Fig 6: CPU usage comparison for virtualized and non-virtualized environments using normal and stress modes

The performance of the proposed attack detection method is compared with many state-of-the-art methods found in the literature. The existing models include the Unsupervised Deep Learning (UDL) method [33], Intel Cache Monitoring Technology (ICM) and Hardware Performance Counters (HPC) tools [34], the Method of AES encryption [35], and the HPC method [36]. The comparison is made in terms of the tool used to generate required data, used performance counter for the experimental study, cache side-channel attacks detected by the models, and the ability to detect stealth attacks.

Table VII: Performance comparison among different side-channel attack detection models (the proposed model is highlighted)

Attack Detection Method	The tool used to Generate the Dataset	Used Performance Counter for Empirical Study	Cache Side-Channel Attacks Detected	Ability to Detect Stealth Attack?
UDL method in [35]	PCM tool from Intel	L1_INST_MISS L1_INST_HIT LLC_MISS	Category 1 Category 3	N
ICM and HPC method in [36]	CMT tool from Intel	L1_MISS LLC_MISS	Category 1 Category 3	N
Method on AES Encryption in [37]	CMT tool from Intel	L1_MISS LLC_MISS	Category 1 Category 3	N
HPC method in [38]	perf tool in Linux	LLC_MISS	Category 1 Category 3	N
SFML-SCAM (Proposed Detection Model)	**PAPI Library in Python**	**Instruction Per Cycle (IPC), L1_MISS, L2_MISS, LLC_MISS and RETIRED_BRANCH**	**Category 1 Category 2 Category3**	**Y**

As presented in Table 7, it is understood that different methods used different approaches in tool usage and performance counters usage. UDL method in [33] used the PCM tool from Intel to create the dataset and used L1 cache hits, L1 cache misses and LLC cache misses as performance counters. It could detect category 1 and category 3 attacks. ICM and HPC methods in [34] used the CMT tool from Intel to create the dataset and used L1 cache misses and LLC cache misses as performance counters. It could detect category 1 and category 3 attacks. Method on AES encryption in [35] used the CMT tool from Intel to create the dataset and used L1 cache misses and LLC cache misses as performance counters. It could detect category 1 and category 3 attacks. HPC method in [36] used the perf tool that comes in Linux OS to create the dataset and used only LLC cache misses as performance counters. It could detect category 1 and category 3 attacks. The proposed method named SFML-SCAM is capable of detecting all three kinds of attacks. Besides, it has the capability of detecting stealth attacks. It makes use of the PAPI library for making datasets at runtime. It uses many performance counters such as Instruction Per Cycle (IPC), L1_MISS, L2_MISS, LLC_MISS, and RETIRED_BRANCH for the detection of attacks using a machine learning approach. The proposed method thus outperforms the methods found in the state of the art.

VII. DISCUSSION

Detecting and mitigating cache side-channel attacks is the main focus of this paper. Six different configurations of VMs are used in an empirical study. The rationale behind this is that it ensures a diversity of experiments.

Table VIII: Performance difference between stress and normal modes in terms of detection rate

Configurations	Detection Rate Difference		
	VM Env (Normal)	VM Env (Stress)	Difference in Detection Rate
1	0.984984	0.925925	0.059059
2	0.971971	0.919919	0.052052
3	0.962962	0.929929	0.033033
4	0.987987	0.923923	0.064064
5	0.945945	0.929929	0.016016
6	0.97097	0.928928	0.042042

Besides each configuration is considered with normal mode and stress mode. In the normal mode, the cache does not experience stress while in the stress mode, there is cache stress which will have more resource utilization. The proposed methodology for detecting cache side-channel attacks showed that there is a difference in the detection rate for normal and stress modes for all configurations.

As presented in Table 8, there is a performance difference between the normal and stress models for all configurations. In the case of stress mode, the detection rate is slightly

Table IX: Performance difference between stress and normal modes in terms of CPU usage

Configurations	Difference in CPU Usage		
	VM Env (Normal)	VM Env (Stress)	Difference in CPU Usage
1	0.61	0.91	0.3
2	0.83	0.91	0.08
3	0.72	0.81	0.09
4	0.82	1.11	0.29
5	0.91	1.11	0.2
6	0.91	0.81	0.1

Table X: Information leakage against stealth attack

Attack Detection Method	Information Leakage due to one Stealth Attack (bits)
UDL method in [35]	189
ICM and HPC method in [36]	246
Method on AES Encryption in [37]	345
HPC method in [38]	195
SFML-SCAM (Proposed Detection Model)	0

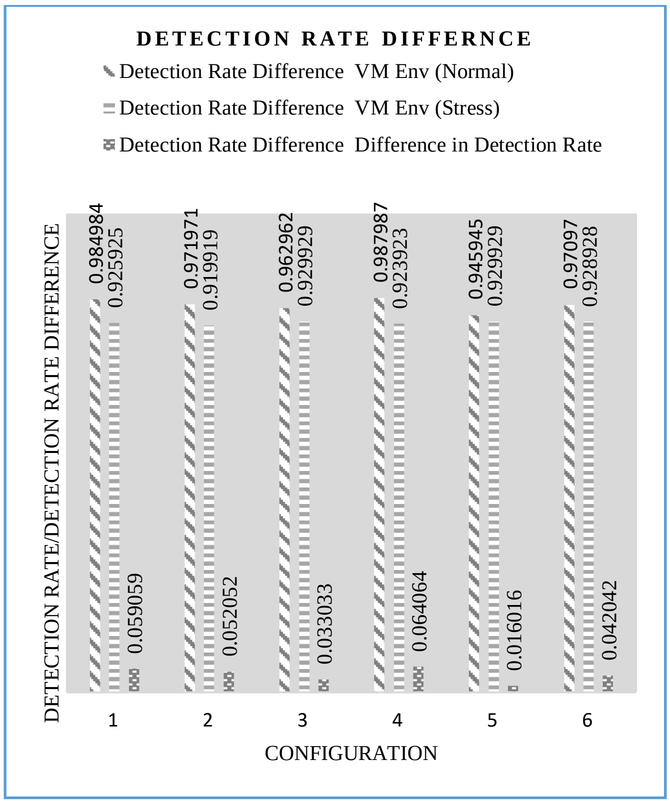

Fig 7: Attack detection rate comparison

reduced. As presented in Table 9, there is a performance difference between the normal and stress models for all configurations in terms of CPU usage. In the case of stress mode, the CPU usage is observed to be more except in the last configuration. As presented in Fig. 7, the attack detection rate is presented for VM normal and VM stress models with stealth attacks due to its detection mechanism. Other existing methods cause information leakage. for different configurations. As presented in Table 10, the proposed detection model does not cause information leakage As presented in Fig. 9, there is no information leakage with the proposed detection method as it can handle it. Other methods caused information leakage in one experiment.

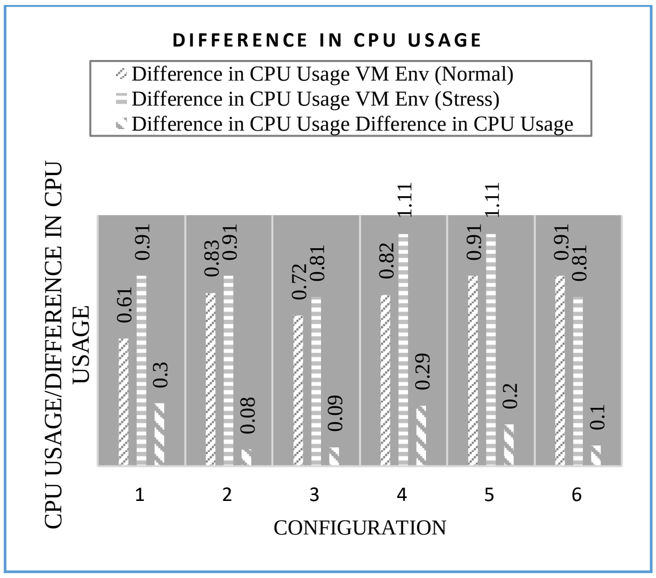

Fig 8: Difference in CPU usage

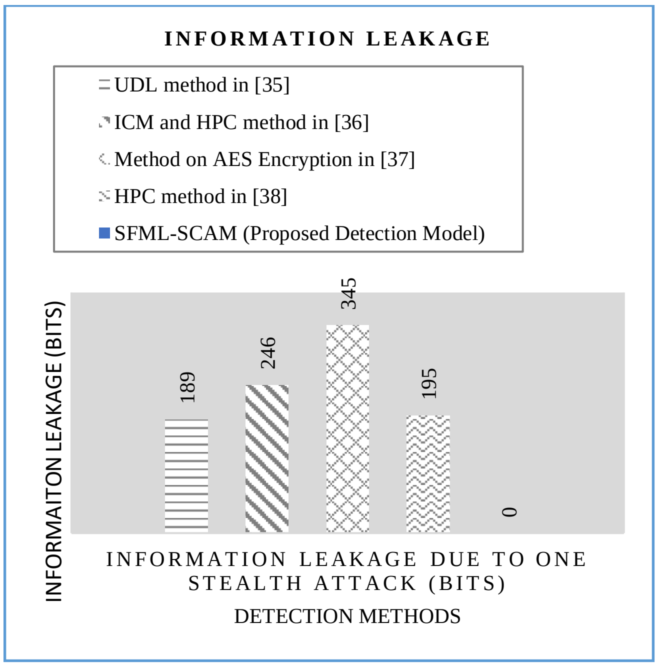

Fig 9: Information leakage against stealth attack

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a security framework for improving Quality of Service (QoS) by detecting and mitigating cache side-channel attacks in virtualized environments. Two algorithms are proposed to realize the framework. They are known as Softmax Function-Based Machine Learning for Side-Channel Attack Monitoring (SFML-SCAM) and Intelligent Noise Addition for Attack Mitigation (INA-AM). The first algorithm takes care of monitoring and detecting cache side-channel attacks and notifying administrators while the second one is responsible to mitigate the effects of attacks. A machine learning technique assists the SFML-SCAM in detecting attacks based on the abnormality in the CPU counters in real-time. The INA-AM algorithm, on the other hand, has a strategy to add noise to the cache data to deceive the attacker and mitigate the effects of the attack. The usage of the machine learning technique enhances the functionality of the SFML-SCAM to detect attacks in real-time. Experiments are made with open-source tools along with a prototype that implements the proposed algorithms. Information leakage is evaluated for all detection methods against stealth attacks. Experimental results revealed that the proposed framework outperforms the state-of-the-art. This paper focused on VM side-channel attacks. However, the defence fails if the hypervisor is compromised to use a tool to launch attacks by adversaries. Therefore, in the future, we focus on the VM protection approach against compromised hypervisors.

References

[1] Mahfouz, A. M., Rahman, M. L., & Shiva, S. G. (2017). Secure live virtual machine migration through runtime monitors. 2017 Tenth International Conference on Contemporary Computing (IC3). P1-5.

[2] Li, Z., Yan, C., Yu, L., & Yu, X. (2018). Energy-aware and multi-resource overload probability constraint-based virtual machine dynamic consolidation method. Future Generation Computer Systems, 80, 139–156.

[3] Tao, Z., Xia, Q., Hao, Z., Li, C., Ma, L., Yi, S., & Li, Q. (2019). A Survey of Virtual Machine Management in Edge Computing. Proceedings of the IEEE, 1–18.

[4] Saeed, A., Garraghan, P., Craggs, B., Linden, D. van der, Rashid, A., & Hussain, S. A. (2018). A Cross-Virtual Machine Network Channel Attack via Mirroring and TAP Impersonation. 2018 IEEE 11th International Conference on Cloud Computing (CLOUD). P1-8.

[5] Mohammad-Mahdi Bazm, Thibaut Sautereau, Marc Lacoste, Mario Südholt, Jean-Marc Menaud. Cache-Based Side-Channel Attacks Detection through Intel Cache Monitoring Technology and Hardware Performance Counters. FMEC 2018 - Third IEEE International Conference on Fog and Mobile Edge Computing, Apr 2018, Barcelona, Spain. P1-6.

[6] Yan, M., Sprabery, R., Gopireddy, B., Fletcher, C., Campbell, R., &Torrellas, J. (2019). Attack Directories, Not Caches: Side Channel Attacks in a Non-Inclusive World. 2019 IEEE Symposium on Security and Privacy (SP).P1-17.

[7] Liu, F., Wu, H., Mai, K., & Lee, R. B. (2016). Newcache: Secure Cache Architecture Thwarting Cache Side-Channel Attacks. IEEE Micro, 36(5), 8–16.

[8] Liu, F., Ge, Q., Yarom, Y., Mckeen, F., Rozas, C., Heiser, G., & Lee, R. B. (2016). CATalyst: Defeating last-level cache side channel attacks in cloud computing. 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). P1-13.

[9] Anwar, S., Inayat, Z., Zolkipli, M. F., Zain, J. M., Gani, A., Anuar, N. B., … Chang, V. (2017). Cross-VM cache-based side channel attacks and proposed prevention mechanisms: A survey. Journal of Network and Computer Applications, 93, 259–279.

[10] Fiore, U., Florea, A., Gellert, A., Vintan, L., & Zanetti, P. (2018). *Optimal Partitioning of LLC in CAT-enabled CPUs to Prevent Side-Channel Attacks. Lecture Notes in Computer Science, 115–123.*

[11] Jia, H., Liu, X., Di, X., Qi, H., Cong, L., Li, J., & Yang, H. (2019). *Security Strategy for Virtual Machine Allocation in Cloud Computing. Procedia Computer Science, 147, 140–144.*

[12] Mushtaq, M., Bricq, J., Bhatti, M. K., Akram, A., Lapotre, V., Gogniat, G., & Benoit, P. (2020). *WHISPER A Tool for Run-time Detection of Side-Channel Attacks. IEEE Access, 1–30.*

[13] Qiu, Y., Shen, Q., Luo, Y., Li, C., & Wu, Z. (2017). A Secure Virtual Machine Deployment Strategy to Reduce Co-residency in Cloud. 2017 IEEE Trustcom/BigDataSE/ICESS. P1-8.

[14] Liu, S., Wei, Y., Chi, J., Shezan, F. H., & Tian, Y. (2019). Side Channel Attacks in Computation Offloading Systems with GPU Virtualization. 2019 IEEE Security and Privacy Workshops (SPW). P1-6.

[15] Yan, M., Gopireddy, B., Shull, T., &Torrellas, J. (2017). Secure Hierarchy-Aware Cache Replacement Policy (SHARP). Proceedings of the 44th Annual International Symposium on Computer Architecture - ISCA '17. P1-14.

[16]] Natu, V., & Duong, T. N. B. (2017). Secure Virtual Machine Placement in Infrastructure Cloud Services. 2017 IEEE 10th Conference on Service-Oriented Computing and Applications (SOCA). P1-8.

[17] Sayakkara, A., Le-Khac, N.-A., & Scanlon, M. (2019). *A survey of electromagnetic side-channel attacks and discussion on their case-progressing potential for digital forensics. Digital Investigation.* P1-12.

[18] MariaMushtaq, AyazAkram, Muhammad Khurram Bhatti, Maham Chaudhry, MuneebYousaf, et al.. Machine Learning For Security: The Case of Side-Channel Attack Detection at Run-time. ICECS2018, Dec 2018, Bordeaux, France. P1-5.

[19] Yang, C., Guo, Y., Hu, H., Wang, Y., Tong, Q., & Li, L. (2019). Driftor: mitigating cloud-based side-channel attacks by switching and migrating multi-executor virtual machines. Frontiers of Information Technology & Electronic Engineering, 20(5), 731–748.

[20] Agarwal, A., & Duong, T. N. B. (2019). Secure virtual machine placement in cloud data centres. Future Generation Computer Systems, 100, 210–222.

[21] Van Cleemput, J., De Sutter, B., & De Bosschere, K. (2017). *Adaptive Compiler Strategies for Mitigating Timing Side Channel Attacks. IEEE Transactions on Dependable and Secure Computing, 1–14.*

[22] Shin, Y., Koo, D., &Hur, J. (2020). *Inferring Firewall Rules by Cache Side-channel Analysis in Network Function Virtualization. IEEE INFOCOM 2020 - IEEE Conference on Computer Communications.* P1-10.

[23] MAsim Mukhtar, Maria Mushtaq, M Khurram Bhatti, VianneyLapotre, Guy Gogniat. FLUSH + PREFETCH: A Countermeasure Against Access-driven Cache-based Side-Channel Attacks. Journal of Systems Architecture, Elsevier, 2020, 104, P1-19.

[24] Sangeetha, G., & Sumathi, G. (2020). An optimistic technique to detect Cache based Side Channel attacks in Cloud. Peer-to-Peer Networking and Applications. P1-14.

[25] Chiappetta, M., Savas, E., & Yilmaz, C. (2016). Real time detection of cache-based side-channel attacks using hardware performance counters. Applied Soft Computing, 49, 1162–1174.

[26] Yuan Xu, Gelei Deng, Tianwei Zhang, Han Qiu, YungangBao. (2021). Novel denial-of-service attacks against cloud-based multi-robot systems. Information Sciences. 576, p329-344.

[27] EnioMarku, GergelyBiczok, and Colin Boyd. (2021). SafeLib: a practical library for outsourcing stateful network functions securely. IEEE, p1-9.

[28] MUSTAFA GAMSIZ AND ALİ HAYDAR ÖZER . (2021). An Energy-Aware Combinatorial Virtual Machine Allocation and Placement Model for Green Cloud Computing. IEEE. 9, p18625-18648.

[29] K. Balaji, P. Sai Kiran, M. Sunil Kumar. (2021). An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. Elsevier, p1-5.

[30] DEAFALLAH ALSADIE. (2021). A Metaheuristic Framework for Dynamic Virtual Machine Allocation with Optimized Task Scheduling in Cloud Data Centres. IEEE. 9, p74218-74233.

[31] Yarom, Y.; Falkner, K. Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In Proceedings of the USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014.

[32] Yarom, Y. Mastik: A Micro-Architectural Side-Channel Toolkit. Available online: https://cs.adelaide.edu.au/~yval/Mastik/Mastik.pdf (accessed on 16 July 2021).

[33] Gulmezoglu, B.; Moghimi, A.; Eisenbarth, T.; Sunar, B. FortuneTeller: Predicting Microarchitectural Attacks via Unsupervised Deep Learning. arXiv 2019, arXiv:1907.03651

[34] Mohammad-Mahdi, B.; Thibaut, S.; Marc, L.; Sudholt, M.; Menaud, J. Cache-based side channel attacks detection through Intel Cache Monitoring Technology and Hardware Performance Counters. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing, Barcelona, Spain, 23–26 April 2018.

[35] Mushtaq, M.; Akram, A.; Muhammad, K.B.; Rao, N.B.R.; Lapotre, V.; Gogniat, G. Run-time Detection of Prime+Probe Side-Channel Attack on AES Encryption Algorithm. In Proceedings of the 2018 Global Information Infrastructure and Networking Symposium, Thessaloniki, Greece, 23–25 October 2018.Appl. Sci. 2020, 10, 984 14 of 14.

[36] Chiappetta, M.; Savas, E.; Yilmaz, C. Real time detection of cache-based side channel attacks using hardware performance counters. Appl. Soft Comput. 2016, 49, 1162–1174.

[37] Mahipal, S., and Sharmila, V. C. (2021). Virtual Machine Security Problems and Countermeasures for Improving Quality of Service in Cloud Computing. 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). P1-6

[38] Dhavlle, A., Rafatirad, S., Khasawneh, K., Homayoun, H., and Dinakarrao, S. M. P. (2021). Imitating Functional Operations for Mitigating Side-Channel Leakage. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, P1-14.

[39] Dutta, S. B., Naghibijouybari, H., Abu-Ghazaleh, N., Marquez, A., and Barker, K. (2021). Leaky Buddies: Cross-Component Covert Channels on Integrated CPU-GPU Systems. 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). P1-13.

[40] Eliyan, L. F., and Di Pietro, R. (2021). DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges. Future Generation Computer Systems, 122, p149–171.

[41] Ranaweera, P., Jurcut, A. D., and Liyanage, M. (2021). Survey on Multi-Access Edge Computing Security and Privacy. IEEE Communications Surveys & Tutorials, 23(2), p1078–1124.