PREDICTING THE MEMORY REQUIREMENTSFOR EFFICIENT DEDEPLICATION IN CLOUD ENVIRONMENT USING DEFENSIVE SCHEME IN VULNERABLE PROCEDURE

Mrs. K. Geetha
Guest Lecturer, Department of Computer Applications, Government Arts College (Autonomous), Salem, Tamilnadu , India.
Dr. A. Vijaya
Assistant professor and Head, Department of Computer Applications, Government Arts College (Autonomous), Salem, Tamilnadu , India.

Abstract - Cloud Computing turned into an ideal solution for business clients to maintain and elevate their business needs to clients by means of cloud services like IaaS, SaaS, and PaaS. On Demand and pay-per-use (scale up) procedures pulled in organizations for cloud adoption and migration. Because of the increased demand for cloud services from clients, Efficient Resource Management in cloud computing become an important errand. To accomplish asset multiplexing in cloud computing, recent investigates were introduced dynamic asset allocation through virtual machines. This paper proposes a novel strategy for Predicting the Memory Requirements for Efficient De-duplication in Cloud Environment and Defensive scheme for vulnerable de-duplication Procedure. Experimental outcomes are supporting in this system is more scalable and dependable than existing methodologies.

Keywords: De-Duplication, Multiplexing, Cloud Data, Prediction, Defensive Scheme.

1. Introduction

Cloud Storage provides clients with abundant extra room and make easy to understand for guaranteed data access. In any case, there is an absence of analysis on optimizing cloud stockpiling for successful data access. With the development of capacity and technology, computerized data has consumed increasingly more space. According to measurements, 60% of advanced data is redundant, and the data compression can only eliminate intra¬-record redundancy [1-4]. To tackle these issues, Deduplication has been proposed. Many organizations set up private cloud stockpiling with their unused assets for asset utilization. More recently de-duplication algorithms have advanced into essential stockpiling area where more spatial data exist, in either case increasing the efficiency of usable stockpiling limit. The tradeoffs between computing costs and reduction in storage costs. It considered another type of attacks on the confidentiality property: cross-VM side-channel attacks. It also present DSVD (Defensive Scheme for Vulnerable De-duplication), a framework to detect, and then relieve, and reserve based side-channel attacks in multi-tenant cloud frameworks. DSVD (Defensive Scheme for Vulnerable De-duplication) works by correlating two events: first, it abuses signature-based detection to identify when the ensured virtual machine (VM) executes a cryptographic application; simultaneously, it utilizes anomaly-based detection techniques to monitor the collocated VMs to identify abnormal reserve practices that are normal during store based side-channel attacks. It has been showing that correlation in the occurrence of these two events over strong evidence of side-channel attacks. Upon assault detection, DSVD (Defensive Scheme for Vulnerable De-duplication) can utilize VM migration to alleviate side-channel information spillage. DSVD (Defensive Scheme for Vulnerable De-duplication) is designed as a lightweight fix to existing cloud frameworks, which doesn't need new hardware uphold, or any hypervisor, operating framework, or application modifications. It demonstrates a model implementation of DSVD (Defensive Scheme for Vulnerable De-duplication) in the OpenStack cloud framework. This evaluation recommends DSVD (Defensive Scheme for Vulnerable De-duplication) accomplishes negligible performance overhead with high detection precision.

Mitigating side-channel attacks in the cloud is challenging. Past work on defeating side-channel attacks have some commonsense downsides: they generally require significant changes to the hardware, hypervisors or guest OSes, making them unreasonable to be deployed in current cloud datacenters. Other work has proposed to relieve these attacks in cloud contexts by intermittent VM migrations to diminish the co-location plausibility between victim VMs and potential malicious VMs. These weighty weight approaches cannot viably prevent side-channel spillage unless performed frequently, making them less useful as VM co-location assumes the order of minutes while side-channel attacks can be done on the order of milli seconds. This research proposes to detect side-channel attacks as they happen and prevent information spillage by triggering VM migration upon assault detection. Nonetheless, side-channel attack detection is non-minor. To do as such, it must overcome a few technical challenges in the application of traditional detection techniques, similar to signature-based detection and anomaly-based detection, to side-channel attacks. Signature-based side-channel detection misuses pattern recognition to detect known assault techniques. While low in bogus negatives for existing attacks, it neglects to recognize new attacks; anomaly-based detection banners practices that deviate significantly from the established normal practices as attacks, which can potentially identify new attacks in addition to known ones. In any case, differentiating side-channel attacks from normal applications is troublesome as these attacks simply perform normal memory gets to which resemble some memory intensive applications [5-7].

To overcome these challenges, this exploration designed DSVD (Defensive Scheme for Vulnerable De-duplication), a constant framework to detect the existence of cross-VM side-channel attacks in clouds. There are two key ideas behind DSVD (Defensive Scheme for Vulnerable De-duplication): first, the victim has unique miniature architectural practices when executing cryptographic applications that need protection from side-channel attacks. So, the cloud provider can identify the occurrence of such events using a signature-based detection strategy. Second, the aggressor VM makes an anomalous reserve behavior when it is stealing information from the victim. Such anomaly is inherent in all side-channel attacks due to the intentional reserve contention with the victim to induce side-channel observations. By correlating these two kinds of events, DSVD (Defensive Scheme for Vulnerable De-duplication)can detect the covert store side-channel attacks with high fidelity. This research implements DSVD (Defensive Scheme for Vulnerable De-duplication) as a lightweight extension to virtual machine monitors. Specially, it (1) utilizes the existing host framework offices to collect miniature architectural features from hardware performance counters that are accessible in all modern commodity processors, and (2) non-intrusively interacts with the existing virtualization framework to monitor the VM's reserve exercises while inducing little performance penalty. These evaluations show that it adequately detects side-channel attacks with high true positives and low bogus positives. DSVD (Defensive Scheme for Vulnerable De-duplication) has a few advantages. To begin with, DSVD (Defensive Scheme for Vulnerable De-duplication) focuses on the underlying drivers of reserve based side-channel attacks and hence is difficult to evade using different assault code, while maintaining a low bogus positive rate. This approach can detect different sorts of side-channel attacks and their variants with a simple method. Second, DSVD (Defensive Scheme for Vulnerable De-duplication) is designed as a lightweight fix to existing cloud systems, which doesn't require new hardware support or hypervisor/OS modifications. Therefore DSVD (Defensive Scheme for Vulnerable De-duplication) can be quickly integrated into modern cloud texture without making uncommon changes to the underlying infrastructure [9,10]. Third, DSVD (Defensive Scheme for Vulnerable De-duplication) exploits hardware performance counters to monitor VM exercises, which detects side-channel attacks within the order of milliseconds with negligible performance overhead. Finally, DSVD (Defensive Scheme for Vulnerable De-duplication) requires no changes to the guest VM or the applications running in it, and thus is transparent to cloud customers.

1.1 LiteratureReviews

Chi Yang and Jinjun Chen [1] proposed a novel scalable data compression based on closeness calculation among the partitioned data chunks with Cloud computing. A closeness model was developed to generate the standard data chunks for compressing enormous data sets. Instead of compression over fundamental data units, the compression was conducted over partitioned data chunks. The MapReduce programming model was received for the algorithms implementation to accomplish some additional versatility on Cloud. With the genuine meteorological enormous sensing data experiments on this U-Cloud platform, it was demonstrated this proposed scalable compression based on data chunk comparability significantly improved data compression performance gains with reasonable data accuracy loss. The significant compression ratio brought emotional existence cost savings. With the popularity of Spark and its claim to fame in processing streaming large data set [1].

Youjip Won, Kyeongyeol Lim, and Jaehong Min [2] proposed a novel multicore chunking algorithm, MUCH, which parallelizes the variable size chunking. Until this point in time, the vast majority of the existing takes a shot at deduplication focus on expediting the redundancy detection measure, while less attention has been paid on the most proficient method to make the record chunking quicker. That proposed a multicore chunking algorithm, MUCH, which guarantees Chunking Invariability. They developed a performance model to compute the segment size that boosts the chunking bandwidth while minimizing the memory requirement [2]. Through extensive physical experiments, it demonstrated that the performance of MUCH scales linearly with the number of cores. In quad-core CPUs, MUCH brings a 400 percent performance increase when the capacity device is sufficiently quick. The benefits of MUCH are evident when it chunks huge records, e.g., tar pictures of document framework snapshot, at high performance stockpiling. MUCH successfully increases the chunking performance with the factor being as high as the number of accessible CPU cores without any additional hardware assistance.

Xu Zhang and Yue Cao [3] Xu Zhang and Yue Cao [3] propose a fully distributed ICN-based caching scheme for content items in Radio Access Network (RAN) at eNodeBs. Such caching scheme works in a cooperative manner within neighborhoods, aiming to reduce store redundancy to improve the decent variety of content distribution [3]. The caching decision logic at individual eNodeBs allows for versatile caching, by considering dynamic context information, such as content popularity and accessibility. The efficiency of the proposed distributed caching scheme is evaluated by means of extensive simulations, which show incredible performance gains, regarding a substantial reduction of backhaul content traffic just as extraordinary improvement on the decent variety of content distribution, and so on.

Chuanshuai Yu, Chengwei Zhang, Yiping Mao, Fulu Li [4] presented the jump based CDC algorithm and added a secondary condition to it to reduce the computing overhead and maintain a similar deduplication ratio. This algorithm fulfills both the content defined condition and the equal likelihood condition [4]. The jump based CDC algorithm with or without a secondary condition can significantly reduce the computing overhead while maintaining a similar deduplication ratio. To determine the technique issue of not being ready to use the rolling hash in the new algorithm, they introduced the pseudo-random transformation to supplant the function of rolling hash.

Daniel Posch, Hermann Hellwagner and Peter Schartner [5] proposed a framework for multimedia delivery in VoD use cases. The concepts of CCN, DASH and BE to make dynamic versatile encrypted chunks of data, which can be inherently stored in the network [5]. The evaluation results show that network inherent caching can increase the efficiency of multimedia delivery. Be that as it may, the usage of versatile concepts prompts the question of how to synchronize clients to abuse the advantage of stored data impeccably. Finding a solution to this issue would enhance the framework extraordinarily.

Chi Yang and Jinjun Chen [6] proposed a novel scalable data compression based on likeness calculation among the partitioned data chunks with Cloud computing. A closeness model was developed to generate the standard data chunks for compressing huge data sets. Instead of compression over fundamental data units, the compression was conducted over partitioned data chunks [6]. The MapReduce programming model was received for the algorithms implementation to accomplish some additional versatility on Cloud. With the genuine meteorological large sensing data experiments on this U-Cloud platform, it was demonstrated this proposed scalable compression based on data chunk closeness significantly improved data compression performance gains with reasonable data accuracy loss. The significant compression ratio brought sensational existence cost savings.

C. Goktug Gurler , S. Sedef Savas , and A. Murat Tekalp [7] proposes two modifications to the Torrent protocol, variable chunk size and versatile scheduling window, for efficient, blunder resilient, versatile P2P streaming of scalable video. The proposed modifications yield superior results as far as number of decoded outlines, hence superior quality of experience, in P2P video streaming [7]. The proposed modifications to BitTorrent for video streaming yield superior results both as far as chunks exchanged between leechers (P2P movement) and the number of decoded outlines (superior quality of experience). In the variable size chunk tests show that the number of decodable casings has significantly increased, improving the PSNR and the QoE. In the variable size chunk tests show that the proposed versatile windowing allows better adaptability against increasing number of leechers. In this manner, with the proposed modifications, the companions would get video at a higher quality and the content providers (seeders) have lower cost of bandwidth.

Haiying Shen and Jin Li [8] propose a DHT-aided chunk-driven overlay for P2P live streaming that objectives higher adaptability, better accessibility, and low latency. The design has three main components: a two-layer progressive DHT based infrastructure, a chunk sharing algorithm, and a video provider selection algorithm. The various leveled DHT based infrastructure offers high adaptability. The chunk sharing algorithm provides service for chunk index collection and discovery, which guarantees high accessibility. The provider selection algorithm enables full utilization of framework bandwidth. As a result, the overlay can provide high-quality video streaming. They additionally propose a centralized and rearranged decentralized provider selection algorithm. DCO is superior to tree-based frameworks in dealing with churn and work based frameworks in bandwidth consumption and latency. All the more importantly, it can deftly exploit framework bandwidth by dynamically matching chunk requesters and providers [8]. The experimental results show that DCO improves the performance of the work based frameworks (pull and push) and tree-based frameworks, in term of versatility, accessibility, latency, and overhead. The experimental results likewise confirm the importance of providing incentives to encourage nodes to fill in as coordinators in the DHT-based infrastructure and the importance of selecting chunk providers with enough bandwidth in chunk delivery.

Deepavali Bhagwat, Kave Eshghi, Darrell D. E. Long and Mark Lillibridge [9] introduced a new technique, Extreme Binning, for scalable and equal deduplication, which is particularly suited for outstanding tasks at hand consisting of individual records with low locality. Existing methodologies which require locality to ensure reasonable throughput perform inadequately with such an outstanding task at hand. Extreme Binning misuses record likeness instead of locality to make only one circle access for chunk lookup per document instead of per chunk, thus alleviating the plate bottleneck issue. It parts the chunk index into two tiers resulting in a low RAM footprint that allows the framework to maintain throughput for a bigger data set than a level index scheme. Partitioning the two-tier chunk index and the data chunks is simple and clean. In a distributed setting, with multiple backup nodes, there is no sharing of data or index between nodes. Records are assigned to a single node for deduplication and capacity using a stateless routing algorithm – meaning it isn't necessary to know the contents of the backup nodes while making this decision. Maximum parallelization can be accomplished due to the one document one backup node distribution. Backup nodes can be added to support throughput and the redistribution of indices and chunks is a clean operation because there are no dependencies between the bins or between chunks joined to different bins[9]. The autonomy of backup nodes makes data management errands such as trash collection, integrity checks, and data reestablish requests efficient. The loss of deduplication is little and is effortlessly compensated by the gains in RAM usage and adaptability.

Chu-Hsing Lin, Chen-Yu Lee, Yi-Shiung Yeh, Hung-Sheng Chien and Shih-Pei Chien [10] generalized the SHA family as SHA-mn that takes subjective length message as input to generate a message digest with required length. They adjust every one of the means of SHA-mn as generalized version that contains padding and parsing; setting the initial hash values, constants, Boolean expressions and functions and message schedule; initializing the eight working factors and for-circle operation; and, computing the ith intermediate hash values. Further, the LHV issue that doesn't exist in the original SHA standard is comprehended. Owing to security considerations, SHA-mn is generalized based on the rules of SHA family design [10]. Although many may not concur the technique for calculating complexity according to the birthday oddity as the collision of full SHA-1 has been found in 2005, the design of SHA is improved. Efficient methods of finding collisions of SHA-256 remain the focus of many analysts to date.

2. Predicting the Memory Requirements for Efficient De-duplication in Cloud Environment

With de-duplication there isn't a one size fits all configuration. Depending on the application and resources accessible, certain algorithms may be more powerful than another. One resource consideration is the all out index memory size required to store the index of unique data signatures. For both fixed and variable square definite matching implementations anin-memory index is used for data signature lookups for determining duplicate blocks. There are a few techniques that use closeness signatures that increase the chunk size to control the memory index size, the tradeoff being the increased reliance on the speed and responsiveness of the de-duplication store for data comparisons. Hence, it focuses when utilizing cloud resources is the specific matching techniques where the memory size for the index is a concern. The main target of this stage is to foresee the required cloud instance type based on memory requirements to run popular de-duplication algorithms on a given dataset and Analyze cloud compute requirements for running de-duplication algorithms at varying chunk granularity. In this experimental evaluation of de-duplication in a cloud-based environment it takes a gander at the following components namely the dataset size, the cloud compute instance requirements, and the length at which the data will be retained in the cloud to analyze the potential cost avoidance surrounding performing fixed and variable de-duplication detection on a given dataset. This research performed analysis on the Different online stockpiling platforms for the compute platform and straightforward stockpiling services for the storage infrastructure. The standard little and huge instance types along with the high cpu medium instances were used in this testing. Below is a recap of resources specifications:

The basic memory requirements estimations:

*Memory Size Estimates= (Data Size / Chunk Size) * (1 – De-duplication %) * (Signature Bytes)*

To gauge the index memory size for a given dataset, the following factors must be determined and or assessed: Data size - what is the complete data set size that is focused for de-duplication. Chunk size - for a fixed chunking algorithm what is the size of the chunk used in the de-duplication implementation. De-duplication percentage is based on the percentage seen during an example run on a subset of the dataset. From this testing an example size of 10 to 15% provides a decent example for the various data types likewise tried. The de-duplication percentage gauges are comparable to comparative measurement results in different studies for the given data types. This stage likewise Evaluate cost factors related with running de-duplication in a cloud environment including compute instance types, de-duplication algorithm chunk granularity, and data stockpiling durations.

This exploration can derive the absolute number of chunks based on these calculations. In the most dire outcome imaginable (the best number of chunks), it would assume that 30% of the data is chunked at the minimum square size, and the remaining 80% of the data would chunk just over the normal chunk size. The best-case scenario (least number of chunks), 30% of the data would chunk at the normal chunk size and the remaining 80% at the maximum chunk size.

*Worst Case Total Chunks = ((.30 * DataSize) / Min Block Size) + ((.80 * Datasize) / Average Block Size)*

*Best Case Total Chunks = ((.30 * DataSize) / Average Block Size) + ((.80 * Datasize) / Max Block Size)*

Worst Case Total Chunks = ((.30 107374182400) / 4096) + ((.80 * 107374182400) / 16384)*

*Best Case Total Chunks = ((.30 * 107374182400) / 16384) + ((.80 * 107374182400) / 65536)*

From the worst- and best-case chunk estimates, this research can now utilize thismemory estimation formula presented earlier to estimate the minimum and maximum memoryrequirements for the index when running the CDC16 algorithm against the 100GBdataset.

*Minimum Memory Requirements = 2867200 * (1 - .30) * (20) = 43008000 bytes ~ 42MB*

*Maximum Memory Requirements = 11468800 * (1 - 305) * (20) = 172032000 bytes ~ 165MB*

Therefore, the memory requirements for this 100GB dataset are in the range of42MB to 165MB.

Additional algorithms exist that use a hybrid methodology by incorporating variable and fixed square techniques, just as little chunk merging techniques to reduce the number of little chunks – in this way, reducing the overhead connected with the large number of little chunks. Compression is additionally often used in conjunction with de-duplication to increase the extra room utilization. Data de-duplication algorithms have been extensively investigated. The techniques available today shift depending on the de-duplication placement, timing of the detection process, and the granularity at which duplicates are discovered. Despite the techniques, the general effectiveness of any of the de-duplication algorithms remains data dependent. The fs-c algorithm used in this exploration is based on the TTTD algorithm using the Rabin fingerprinting for generating the natural chunk boundaries. In addition to the inner workings of the de-duplication algorithms, the data characteristics will help in understanding the space savings obtained by the various algorithms. There are a few factors such as data type, scope of the de-duplication, and data capacity period that assume a function in by and large de-duplication savings. De-duplication savings are often expressed as far as ratios, which is in relation to the number of input bytes to the de-duplication measure divided by the number of bytes of output.

3. Defensive scheme for vulnerable de-duplication Procedure

DSVD (Defensive Scheme for Vulnerable De-duplication) is provided by the cloud administrator as a security service to the customers who are eager to pay additional cost for better security, as in the Cloud Monatt cloud framework. It shows the architecture of DSVD (Defensive Scheme for Vulnerable De-duplication), and the workflow of detecting side-channel attacks. This research implements DSVD (Defensive Scheme for Vulnerable De-duplication) into the CloudMonatt framework. Three sorts of workers, the Cloud Controller, the Attestation Server and regular cloud workers, are relevant to this discussion. The Cloud Controller is the cloud manager, responsible for taking VM detection requests and servicing them for every customer. The Attestation Server is a dedicated server to manage the provided security services and coordinate the interaction between the Cloud Controller and the cloud workers.

The Signature Database is used to store signatures of crypto programs. DSVD (Defensive Scheme for Vulnerable De-duplication's) functionality within a cloud worker is firmly integrated with the host OS. DSVD (Defensive Scheme for Vulnerable De-duplication) consists of three modules, with each one running on a dedicated core. The Victim Monitor is responsible for collecting the protected VM's runtime events, which will be taken care of to Signature Detector to detect the cryptographic programs using this signature-based technique; The Attacker Monitor is responsible for collecting store exercises of the different VMs, using anomaly-based detection way to deal with identify side-channel attackers. This examination used the Linux perf event kernel API for the PMU to manage the Hardware Performance Counters, along these lines no change is needed to the hypervisor itself.

DSVD (Defensive Scheme for Vulnerable De-duplication) includes four steps with different paths. Eachstep is described below:

Step 1: generating cryptographic signature. In this progression, the customer who seeks side-channel detection services for his secured VM can indicate to the Cloud Controller what sensitive applications to be ensured, by providing the signatures generated offline using Hardware Performance Counters (not necessarily on the same hardware) or essentially the executables. Then the Cloud Controller will run these crypto programs on a dedicated worker with a similar configuration as the Cloud Server that hosts the secured VM and use Hardware Performance Counters to generate the signatures for the customer. The signatures will be put away in the Signature Database in the Attestation Server for future reference. They will likewise be sent to the cloud server that has this VM.

Step 2: detecting cryptographic applications. This progression happens at runtime. In this progression, the Victim Monitor monitors the secured VM using Hardware Performance Counters. It intermittently records the event counts as a period sequence, while the Signature Detector keepscomparing the latest window of data points in the sequence with the signature. If a signature coordinate is found, the Signature Detector can identify the protected VM is performing a cryptographic application and signal this result to the Attacker Monitor.

Step 3: monitoring store exercises. This progression happens concurrently with Step2. The Attacker Monitor Exploits Hardware Performance Counters to monitor all untrusted VMs simultaneously. One challenge is that not enough performance counters are accessible on the workers to monitor all VMs, if this number is huge: the majority of the Intel and AMD processors support up to six counters, and the number of counters does not scale with the number of cores. Along these lines, when there are a ton of VMs on the server, the Attacker Monitor cannot monitor them concurrently.

To take care of this issue, this examination used a period domain multiplexing strategy: The Attacker Monitor identifies dynamic vCPUs that share LLC with the secured VM as the monitored vCPUs, and then measures every one of them in turn. In particular, in each period, the Attacker Monitor uses a kernel module to check the state and CPU affinity of each vCPU of each VM from its undertaking struct in the kernel. The Attacker Monitor marks the vCPUs in the running state that are sharing a similar LLC with the ensured VM as monitored. Then it sets up Hardware Performance Counters to measure each monitored vCPU's reserve misses and hits in turn. When the Attacker Monitor is noticed that a cryptographic application is happening in the protected VM, it will compare each monitored vCPU's reserve misses and hits previously and during the cryptographic application. In the event that one vCPU has an abrupt increase in the number of store misses or hits during the cryptographic application, the Attacker Monitor will hail a caution.

Step 4: eliminating side channels. Once the Attack Monitor notices that one co-tenant VM has abnormal store conduct precisely when the ensured VM executes cryptographic applications, it will raise alert for side-channel attacks. In addition, the Attestation Server will report this incident to the Cloud Controller for further processing, such as shut down the malicious VM or eventually block the attacker's account.

4. Results and Discussions

Worst Case Total Chunks =

*((.30 * DataSize) / Min Block Size) + ((.80 * Datasize) / Average Block Size)*

Best Case Total Chunks =

*((.30 * DataSize) / Average Block Size) + ((.80 * Datasize) / Max Block Size)*

4.1 The below results depict the simulation results of the memory prediction and the defensive Scheme mechanism

Zero(%)

Trace Methods	Zero(%)
PSC	29.06
EP2C	80.72
ER SCHEME	59.82
ESC	46.94

Table 1: Statistic results of the trace methods Zero (%) values

Table 1 represents statistic results of various trace methods of zero(%) values. The proposed method of PSC is compare with other trace methods then execute good performance of zero(%) process. PSC value is lower than this process. EP2C method is executed highly process values.

Continuous Zeros (%)

Trace Methods	Continuous Zeros(%)
PSC	7.64
EP2C	26.88
ER SCHEME	59.55
ESC	30.3

Table 2: Statistic results of the trace methods Continuous zeros (%) values

Table 2 represents statistic results of various trace methods of continuous zero(%) values. The proposed method of PSC is compare with other trace methods then execute good performance of continuous zero(%) process. PSC value is lower than this process. ERSCHEME method is processed high level values.

Bound (%)

Trace Methods	Bound(%)
PSC	19.71
EP2C	25.65
ER SCHEME	40.83
ESC	20.98

Table 3: Statistic results of the trace methods Bound (%) values

Table 3 represents statistic results of various trace methods of bound(%) values. The proposed method of PSC is compare with other trace methods then execute good performance of bound(%) process. PSC value is lower than this process. ERSCHEME executed high level of bound ratio values.

Low (%)

Trace Methods	Low(%)
PSC	4.89
EP2C	18.2
ER SCHEME	31.1
ESC	24.6

Table 4: Static results of the trace methods Low (%) values

Table 4 represents statistic results of various trace methods of low(%) values. The proposed method of PSC is compare with other trace methods then execute good performance of low(%) process. PSC value is lower than this process. ERSCHEME exposed high level of low ratio values.

Entropy (%)

Trace Methods	Entropy(%)
PSC	2.54
EP2C	7.39
ER SCHEME	3.85
ESC	12.34

Table 5: Static results of the trace methods Entropy (%) values

Table 5 represents statistic results of various trace methods of entropy(%) values. The proposed method of PSC is compare with other trace methods then execute good performance of entropy(%) process. PSC value is lower than this process. ESC represented high level of entropy ratio values.

Compression Ratio (%)

Trace Methods	Compression Ratio(%)
PSC	86.34
EP2C	16.49
ER SCHEME	58.89
ESC	44.65

Table 6: Static results of the trace methods Compression Ratio (%) values

Table 6 represents statistic results of various trace methods of compression ratio (%) values. The proposed method of PSC is compare with other trace methods then execute good performance of compression ratio(%) process. PSC value is higher than this process. EP2C is exposed into low level of compression ratio value.

De-duplication Ratio

Trace Methods	De-duplication Ratio (%)
PSC	25.52
EP2C	13.06
ER SCHEME	6.5
ESC	20.72

Table 7: Static results of the trace methods De-duplication ratio (%) values

Table 7 represents statistic results of various trace methods of de-duplication ratio (%) values. The proposed method of PSC is compare with other trace methods then execute good performance of de-duplication ratio(%) process. PSC value is higher than this process. ERSCHEME represented as low level of de-duplication ratio values.

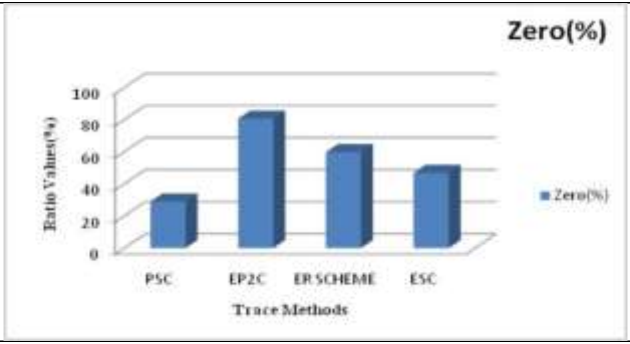

Fig 1: Zero (%) ratio values

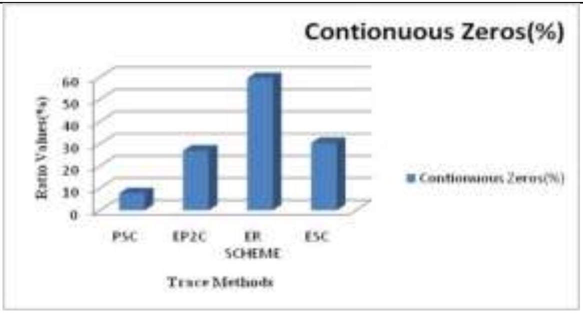

Fig 2: Continuous Zeros (%) ratio values

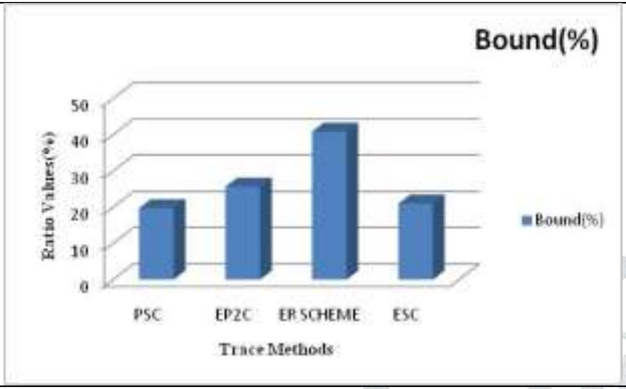

Fig 3: Bound (%) ratio values

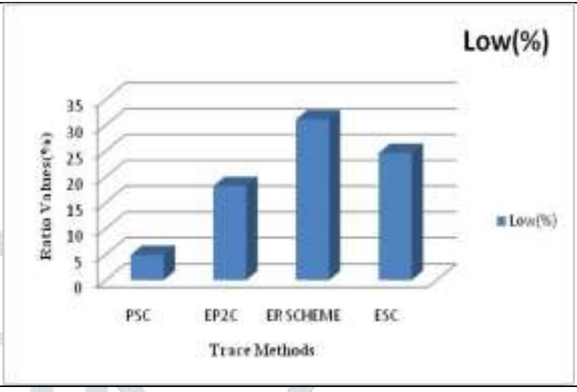

Fig 4: Low (%) ratio values

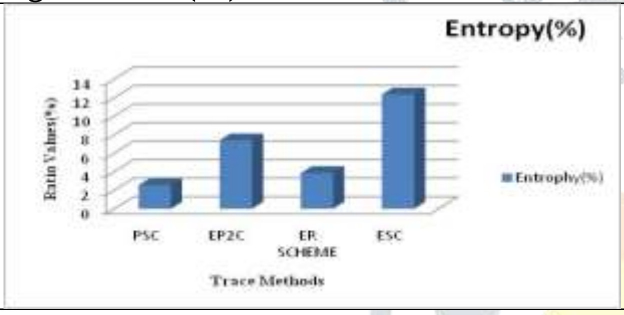

Fig 5: Entropy (%) ratio values

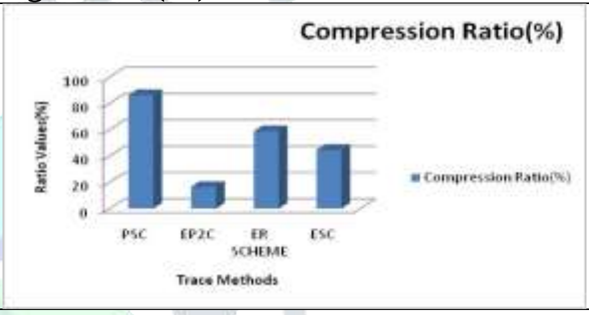

Fig 6: Compression ratio (%) values

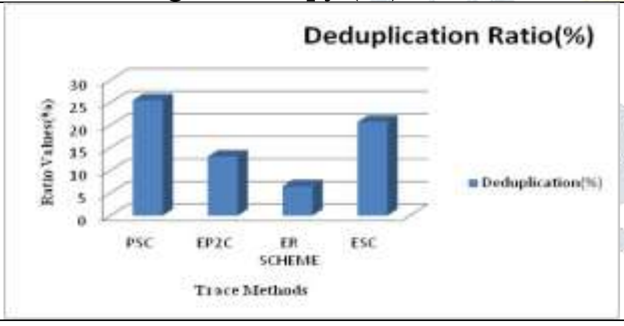

Figure 7: De-duplication ratio values

Figure 1 represents ratio values of follow strategies zero (%) measure. Each follow techniques have executed zero (%) ratio values. Proposed strategy for PSC executed great result of their cycle. Figure 2 represents ratio values of follow strategies continuous zero (%) measure. Each follow techniques have executed continuous zero (%) ratio values. Proposed technique for PSC executed great result of their cycle. Figure 3 represents ratio values of follow techniques continuous bound (%) measure. Each follow techniques have executed bound (%) ratio values. Proposed technique for PSC executed great result of their cycle. Figure 4 represents ratio values of follow techniques continuous low (%) measure. Each follow strategies have executed low (%) ratio values. Proposed strategy for PSC executed great result of their cycle. Figure 5 represents ratio values of follow strategies continuous entropy (%) measure. Each follow strategies have executed entropy (%) ratio values. Proposed technique for PSC executed great result of their cycle. Figure 6 represents ratio values of follow strategies compression ratio (%) measure. Each follow strategies have executed compression ratio (%) ratio values. Proposed technique for PSC executed great result of their cycle. Figure 7 represents ratio values of follow techniques de-duplication ratio (%)

measure. Each follow techniques have executed de-duplication ratio (%) ratio values. Proposed technique for PSC executed great result of their cycle.

4.2 The below results depict the simulation results of the Defensive scheme for vulnerable de-duplication Procedure

Mitigation Method	Virtual Machine Introspection	Content Delivery Network	Proposed Anomaly Detection
20	7	31	37
27	14	39	45
26	20	47	61
44	29	61	80
58	37	70	96

Table 8- Detection Accuracy

Mitigation Method	virtual Machine Introspection	Content Delivery Network	Proposed Anomaly Detection
45	20	31	3
57	27	39	10
70	26	47	18
88	44	61	25
91	58	70	33

Table 9 – Detection Latency

Mitigation Method	virtual Machine Introspection	Content Delivery Network	Proposed Anomaly Detection
20	12	17	38
39	24	30	53
47	35	40	67
60	49	52	80
77	57	64	99

Table -10 Performance Overhead

Mitigation Method	virtual Machine Introspection	Content Delivery Network	Proposed Anomaly Detection
37	22	20	59
40	34	39	73
54	65	67	87
62	79	80	90
74	87	87	99

Table -11 Data Transfer Ratio

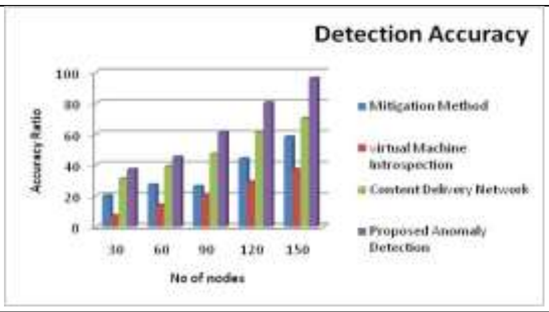

Fig 8: Detection Accuracy

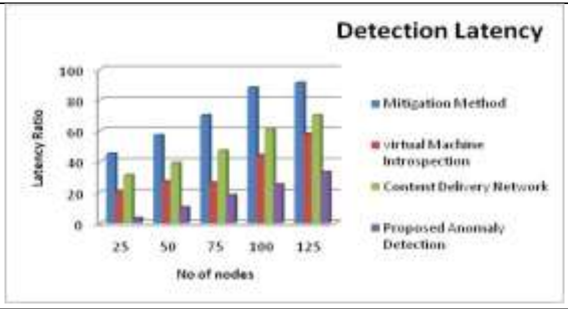

Fig 9: Detection Latency

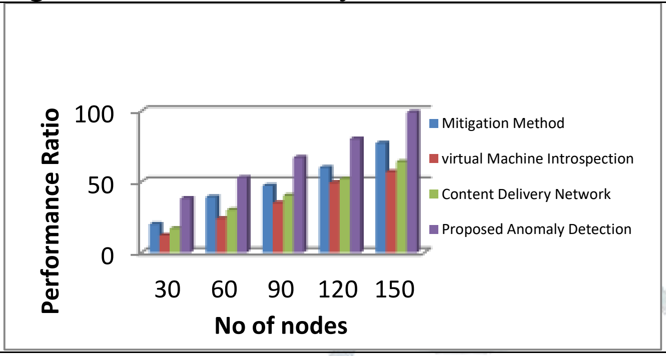

Fig 10: Performance Overhead

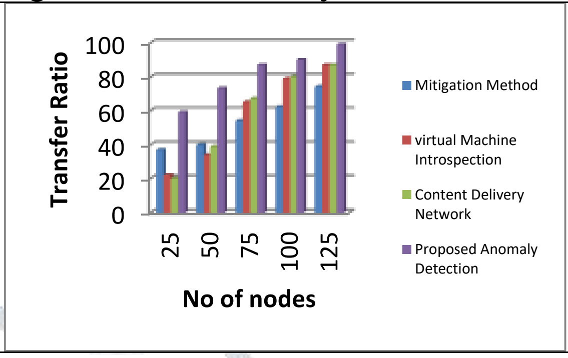

Fig 11: Data Transfer Ratio

Figure 8 represents ratio values of detection accuracy process. Every strategy has executed detection accuracy values. Proposed Anomaly detection strategy executed great result of their process. Figure 9 represents ratio values of detection latency process. Every strategy has executed detection latency values. Proposed Anomaly detection strategy executed great result of their process. Figure 10 represents ratio values of performance overhead process. Every strategy has executed performance overhead values. Proposed Anomaly detection technique executed great result of their process. Figure 11 represents ratio values of data transfer process. Every strategy has executed data transfer values. Proposed Anomaly detection technique executed great result of their process.

CONCLUSION

DSVD (Defensive Scheme for Vulnerable De-duplication) leverages the existing Hardware Performance Counter feature to both monitor a victim VM's cryptographic operations and capture a potential attacker VM's abnormal conduct during this time. DSVD (Defensive Scheme for Vulnerable De-duplication) is designed as a lightweight extension to the cloud system and does not require new hardware, hypervisor/OS or application modifications. The feasibility of DSVD (Defensive Scheme for Vulnerable De-duplication) is approved by this implementation on the OpenStack framework. This evaluation shows DSVD (Defensive Scheme for Vulnerable De-duplication) can detect reserve based side-channel attacks with high fidelity, while introducing minimal overhead to the cloud applications. This research used the Memory Prediction and Defensive Scheme Vulnerabilities to assess the future need adequately. This methodology accomplished high accuracy around predicting the future resource needs.

REFERENCES

[1] Chi Yang and Jinjun Chen, "A Scalable Data Chunk Similarity based Compression Approach for Efficient Big Sensing Data Processing on Cloud", IEEE Transactions on Knowledge and Data Engineering; Print ISSN: 1041-4347; Electronic ISSN: 1558-2191; CD-ROM ISSN: 2326-3865; June 1 2017

[2] Youjip Won, Kyeongyeol Lim, and Jaehong Min, "MUCH: Multithreaded Content-Based File Chunking", IEEE Transactions on Computers (Volume: 64, Issue: 5, May 1 2015), 14 May 2014; 14 May 2014

[3] Xu Zhang and Yue Cao, "A Cooperation-Driven ICN-based Caching Scheme for Mobile Content Chunk Delivery at RAN", 13th International Wireless Communications and Mobile Computing Conference (IWCMC); IEEE; ISSN: 2376-6506,2017

[4] Chuanshuai Yu, Chengwei Zhang, Yiping Mao, Fulu Li, "Leap-based Content Defined Chunking --- Theory and Implementation", IEEE; Print

ISSN: 2160-195X; Electronic ISSN: 2160-1968,31st 2015

[5] Daniel Posch, Hermann Hellwagner and Peter Schartner, "On-Demand Video Streaming based on Dynamic Adaptive Encrypted Content Chunks", IEEE International Conference on Network Protocols (ICNP); ISBN: 978-1-4799-1270-4; ISSN: 1092-1648,21st2013

[6] Chi Yang and Jinjun Chen, "A Scalable Data Chunk Similarity based Compression Approach for Efficient Big Sensing Data Processing on Cloud", IEEE Transactions on Knowledge and Data Engineering (Volume: 29, Issue: 6, June 1 2017);ISSN: 1041-4347; 2017

[7] C. Goktug Gurler1 , S. Sedef Savas2 , and A. Murat Tekalp3, "VARIABLE CHUNK SIZE AND ADAPTIVE SCHEDULING WINDOW FOR P2P STREAMING OF SCALABLE VIDEO", IEEE International Conference on Image Processing; Print ISSN: 1522-4880; Online ISSN: 1522-4880; Electronic ISSN: 2381-8549; 19th2012

[8] Haiying Shen and Jin Li , "A DHT-Aided Chunk-Driven Overlay for Scalable and Efficient Peer-to-Peer Live Streaming", IEEE Transactions on Parallel and Distributed Systems (Volume: 24, Issue: 11, Nov. 2013); Print ISSN: 1045-9219; Electronic ISSN: 1558-2183; CD-ROM ISSN: 2161-9883; 22 October 2012

[9] Deepavali Bhagwat, Kave Eshghi, Darrell D. E. Long and Mark Lillibridge, "Extreme Binning: Scalable, Parallel Deduplication for Chunk-based File Backup", IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems; Print ISBN: 978-1-4244-4927-9; CD-ROM ISBN: 978-1-4244-4928-6; 2009

[10] Chu-Hsing Lin, Chen-Yu Lee, Yi-Shiung Yeh, Hung-Sheng Chien and Shih-Pei Chien, "Generalized Secure Hash Algorithm: SHA-X",IEEE EUROCON - International Conference on Computer as a Tool; Print ISBN: 978-1-4244-7486-8;CD-ROM ISBN: 978-1-4244-7485-1; 2011

[11] Sang-Hyun Lee, Kyung-Wook Shin, "An Efficient Implementation of SHA processor Including Three Hash Algorithms (SHA-512, SHA-512/224, SHA-512/256)", IEEE; Print on Demand (PoD) ISBN: 978-1-5386-4754-7; 2018

[12] IMTIAZ AHMAD AND A. SHOBA DAS, "Analysis and Detection Of Errors In Implementation Of SHA-512 Algorithms On FPGAs", The Computer Journal (Volume: 50, Issue: 6, Nov. 2007); IEEE; Print ISSN: 0010-4620; Electronic ISSN: 1460-2067; NOV 2007

[13] Alavi Kunhu, Hussain Al-Ahmad and Fatma Taher, "Medical Images Protection and Authentication using hybrid DWT-DCT and SHA256-MD5 Hash Functions", IEEE International Conference on Electronics, Circuits and Systems (ICECS); **Publisher:** IEEE; Electronic ISBN: 978-1-5386-1911-7; Print on Demand(PoD) ISBN: 978-1-5386-1912-4; 24th2017

[14] Mochamad Vicky Ghani Aziz, Rifki Wijaya, Ary Setijadi Prihatmanto, Diotra Henriyan, "HASH MD5 Function Implementation at 8-bit Microcontroller", Joint International Conference on Rural Information & Communication Technology and Electric-Vehicle Technology (rICT & ICeV-T); IEEE; Electronic ISBN: 978-1-4799-3365-5; Print ISBN: 978-1-4799-3363-1;CD-ROM ISBN: 978-1-4799-336; 2017

[15] Eko Sediyono, Kartika Imam Santoso and Suhartono, "Secure Login by Using One-time Password Authentication Based on MD5 Hash Encrypted SMS", International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE; Electronic ISBN: 978-1-4673-6217-7; Print ISBN: 978-1-4799-2432-5; CD-ROM ISBN: 978-1-4799-2659-6; 2013