I. J. Computer Network and Information Security, **2023, 4, 61-71**
Published Online on August 8, 2023 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2023.04.06

Distributed Denial of Service Attack Detection Using Hyper Calls Analysis in Cloud

K. Umamaheswari*
Department of Computer Science, Bharathi Women's College, Chennai, India
E-mail: uma.tvr1981@gmail.com
ORCID iD: https://orcid.org/0000-0002-5279-6659
*Corresponding author

Nalini Subramanian
Department of Information Technology, Rajalakshmi Engineering College, Chennai, India
E-mail: mrgn.nalini@gmail.com
ORCID iD: https://orcid.org/0000-0002-8175-3260

Manikandan Subramaniyan
Electrical power Section, Engineering Department, University of Technology and Applied Sciences – IBRI, Sultanate of Oman
E-mail: sg.mai79@ibrict.edu.in

Received: 08 March 2022; Revised: 29 June 2022; Accepted: 12 October 2022; Published: 08 August 2023

Abstract: In the scenario of Distributed Denial of Service (DDoS) attacks are increasing in a significant manner, the attacks should be mitigated in the beginning itself to avoid its devastating consequences for any kind of business. DDoS attack can slow down or completely block online services of business like websites, email or anything that faces internet. The attacks are frequently originating from cloud virtual machines for anonymity and wide network bandwidth. Hyper-Calls Analysis(HCA) enables the tracing of command flow to detect any clues for the occurrence of malicious activity in the system. A DDoS attack detection approach proposed in this paper works in the hypervisor side to perform hyper calls based introspection with machine learning algorithms. The system evaluates system calls in hypervisor for the classification of malicious activities through Support Vector Machine and Stochastic Gradient Descent (SVM & SGD) Algorithms. The attack environment created using XOIC attacker tool and CPU death ping libraries. The system's performance also evaluated on CICDDOS 2019 dataset. The experimental results reveal that more than 99.6% of accuracy in DDoS detection without degrading performance.

Index Terms: Hyper Calls Analysis, Machine Learning, Support Vector Machine, Stochastic Gradient Descent.

1. Introduction

It is now very common that attackers use various automated tools and programs to gain access to user's data especially in cloud. It is a proven concept that to access system's capabilities by boycotting system calls is very difficult. All of the user's processes use system calls for requesting different services from the Operating system's kernel. There will be overwhelmingly large system logs in the case of hyper-calls in cloud hypervisors. This is extremely hard to fetch useful information from the huge communication data of guest virtual machines and hypervisors [1]. The low level requests include every basic interaction between the guest images and hypervisors for allocation and de-allocation of resources, operations with files and process control tasks. It can include any hyper-calls with out-of-bound parameters for performing any malicious activity also. The activities can aim towards exhausting the CPU, memory and resources owned by the hypervisor that can lead to failure in the host machine's performance. The associated legitimate guest machines can be starved from resources. The past researches have provided only passive defense that they cannot be traced back to the attack source. The work proposed in the paper aims at active defense to perform machine learning based hyper-calls analysis to report DDoS nature of attacks as and when they are initiated. The system's performance is analyzed in CICDDOS 2019 dataset. Frequently, multiple Intrusion Detection Systems (IDSs) may be employed to generate alerts in the system, those alerts are usually raised in low level and do not produce any specific information about an attack. The present approach overcomes these problems through hypervisor based system call analysis to find the source of attack and attack strategy. The raw alerts generated from the IDSs are analyzed with multiple machine learning algorithms to generate attack graph. The alerts are aggregated, correlated, clustered and reduced for false positives. Finally, Alerts are verified for prioritization and hyper-alerts generated with attack graphs.

2. Related Works

2.1. Hyper Calls Analysis

Virtual Machine Introspection (VMI) provides options to analyze virtual machine activities [2]. Similar to system calls, hyper calls can be gained by the attackers for attacking the hypervisor through breaking isolation of virtualization and constraints of virtual machines. Hence, hyper-calls injection attacks need to be mitigated in the beginning itself by analyzing for hyper-calls with anomalous values in parameters or in irregular order or from irregular call sites. Attackers often attack hypervisors for preventing its operation similar to a state of crash. These kind of attacks are considered to be a category of DoS that could disrupt the state of hypervisor that is desirable for the attacker. The victim hypervisor may allow the implementation of destructive code or access to high-level operations. It can also lead to accessing unauthorized information. Several incidences are reported for this kind of hyper-call attacks [3-5]. Hyper-call analysis can categorize the calls as follows [6].

- Single hyper-call
- Hyper-call with ordinary list of parameters (benign)
- Hyper-call with modified list of parameters (malicious)
- Sequence of hyper-calls in specific order
- Implementing a particular hyper-call repeatedly
- Implementing sequence of hyper-calls repeatedly

The research is focused on the category of sequence of hyper-calls in a specific order. The concept of VMI is extended here to perform hypervisor introspection (HVI). Hyper-calls are extracted to monitor the execution of guest virtual machines. As the guest virtual machine kernels initiate the hyper-calls, we use the environment of nested hypervisors in our system to detect maliciously crafted hyper-calls as shown in figure 1. Hyper-call detectors are implemented in the lowest level of hypervisor (L0). This type of nested virtualization allows us to detect all the hyper-calls initialized from the guest VMs in the top hypervisor (L1). All the latest CPUs from Intel and AMD support this kind of nested virtualization without requiring extra hardware. It can be used to monitor various versions of level 1 hypervisors. The monitor can be updated easily to adapt with new hypervisor versions as it lies in the layer of level 0.

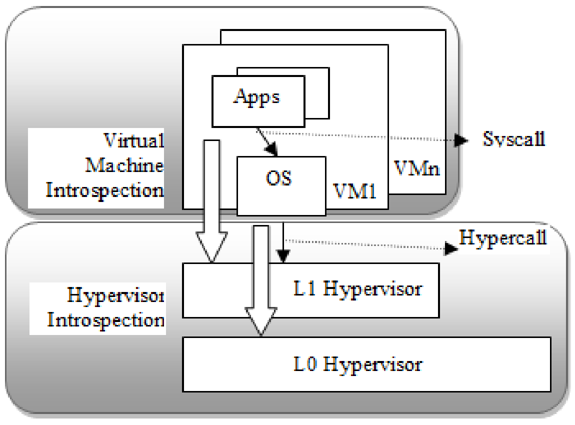

Fig.1. Nested hypervisors and Hypervisor Introspection

2.2. HTTP DDoS Attack Detection

The proposed system aims to improve DDoS attacks detection accuracy and to consider false positive rate and running time. There exists many related work for the enhancement of DDoS attack detection. An App-DDoS detection method is proposed by Sangjae L et al. [7] which is the concept of sequence order, independent network traffic profiling technique. Attributes are extracted from web page request sequences and a PCA based model used for the normal web browser behavior profiling. A defense mechanism is proposed by Dantas Yetal [8] for mitigating HTTP POST Flooding named as SeVen on the basis of Adaptive Selective Verification (ASV). The system was not suitable enough for the detection of Application Layer DDoS attacks as the ASV assumed the communications are stateless- simple client-server syn-ack interactions. SeVen extends ASV but that keeps the resources in busy state even while waiting for receiving total payload. Hence the method is not suitable for dealing with distributed HTTP POST flooding attack in case of large number of reflectors are used to send payloads. A DDoS attack detection method was presented [9] through HTTP packet pattern and rule engine in the environment of cloud computing. The system integrates HTTP GET flooding of DDoS attacks and MapReduce processing for the attack detection in cloud. A DDoS detection system presented by Zecheng He et al. [10] is based on machine learning techniques. Statistical information from the cloud server's hypervisor and the virtual machines are utilized for detection purpose. The system evaluated 9 machine learning algorithms and the most appropriate is selected on the basis of detection performance. A bio-inspired anomaly based application layer DDoS attack detection proposed by Sreeram et al. [11] for earlier and faster detection. The system is evaluated with the CAIDA dataset and achieved high detection rate for the HTTP flooding attacks. Irfan et al. [12] have proposed a DDoS detection system with 4 machine learning classifiers of Naive Bayes, MLP, SVM, and Decision trees for evaluation in a local dataset. Most of the approaches have applied statistical and thresholding techniques for the discrimination of benign and malicious traffic. In order to reduce false positives and improve accuracy, appropriate machine learning technique should be applied in the DDoS system.

2.3. Hyper-Calls Based Attack Detection

The domains of virtual servers are not fully reliable because of emerging new attack methods and the developers may not aware about all existing threats. The Dom0 of hypervisor is targeted in many different attacks. Hyper-call based attack detection is proposed by many researchers like Bharadwaja et al. [13] (Col-labra), Maiero and Miculan [14] (Xenini), Wu et al. [15], (C2Detector), Le [16] (MAC/HAT), Wang et al. [17] (RandHyp). Collabra is based on the concept of collaborative hypercall- based IDS, which gives an anomaly score for each hypercall and then compare it with a threshold for normal and anomaly detection. DoS attacks based on XML and HTTP were investigated by Chonka et al. [18]. Bacon et al. [19] has done a work for information flow control in Platform as a Service (PaaS). The threats in cloud environment are studied and classified by Hashizume et al. [20]. The security aspects of Xen virtualization infrastructure in cloud environment are reported in the work. Certain types of hyper-call attacks in Xen hypervisor are considered and solution proposed. Domain security is insisted many other works of [21] and [22] by using the hypervisor to monitor the domain. But the works did not overcome all the threats and new exploits. A mechanism known as CloudVisor is proposed by Zhang et al. [23] is a monitoring system for the protection of domain resources. But the system is unable to apply changes on the available virtualization system architecture and to update with future hypervisors. The management domain is split up into single purpose components in a solution called Xoar suggested by Colp et al. [24] for hypervisor security. The thesis of Hoang [25] proposed a solution to increase hypervisor security and for the reduction of hyper-call attacks. But the system is not practically applicable to use a MAC method for hyper-call authentication. XenPump is a mechanism for the prevention of time channel attacks proposed by JingZheng et al. [26]. But the system can significantly reduce system efficiency and hypervisor's performance can be lagged. In [27], the Dom0 privileges are split into two parts for stopping the spread of attacks to DomU. But the system did not address DDoS attacks. A secure execution environment designed in the system of [28] for the guest virtual machine through the categorization of hyper-calls into 3 groups: 1. The hyper-calls that should be prohibited as they are harmful to DomU privacy and integrity and not necessary. 2. The hyper-calls that should be unmodified as they are not harmful to DomU. 3. The hyper-calls that should be used with restrictions as they are harmful and necessary for the management of DomU. The system has used encryption and hash functions for the protection of virtual machine's integrity and privacy. A software for hypercall attack simulation is proposed by Milenkoski et al. [29]. The system is installed on a guest VM for the execution of various hyper-call attacks. The system is aimed to evaluate the performance and efficiency of IDS system. Some more systems of [30-32] have proposed models for the security of cloud based systems through machine learning models for the performance comparison of different algorithms.

3. Methodology

The work aims at the detection of DDoS attack through hyper-calls analysis in the nested hypervisors environment. Any changes in hyper-calls traffic can be used as indication for the deviation of normal activities. Machine learning algorithms of Linear Support Vector Machines (SVMs) is applied to classify malicious and benign traffic of hyper-calls issued in the nested hypervisors setup. The hyper-calls are received by passing requests to Virtual Machine Monitor (VMM) services to access memory and file systems. The system's performance is pre-evaluated in the attack environment generated from XOIC and CPU death ping library tools and then exposed to the dataset of CICDDOS 2019. The method comprised of four steps:

- Hyper-calls tracing
- Key fields extraction from voluminous data
- Detection of malicious activity through application of SVM-SGD algorithm.
- Classification of malware sample using Naïve Bayes algorithm (LLRT).

The methodology in 4 steps is shown in figure 2 and results after application of SVM-SGD algorithm are presented in in the subsequent sections.

3.1. Locating Hyper-calls

Arguments for hyper-calls will be delivered through generic registers. The registers can be monitored for getting hyper-call related information. However, the registers produce general and noisy data and may incur high performance costs. Hence another method like syscall hooking is used that set traps at hyper-calls for the VM to be monitored. Hypercall page is initialized through the function *hypercall_page_initialise* in the task of domU and dom0 creation. The hypercall page is split into 128 blocks that would contain the code for initializing hypercall entries. There are about 40 general hypercalls and 8 architecture- specific calls and the virtual address calculated by the addition of 32 times hyper-call number with the hypercall page's base address. The symbol file of dom0 can be used to obtain the hypercall page's virtual address. An additional address translation performed from guest to physical machine through LibVMI library. Here, L1 address of hypercall entries translated into L0 addresses using *xc_map_foreign_range* function from XenControl Library (libxl).

3.2. Key Fields Extraction

The binary files of hyper-call logs need to be converted into the form of numerical vector data for applying malware detection algorithms. The process of redundant deductions can be avoided by selecting appropriate key fields from the resultset. The log data will be made available for subsequent processing through the following steps.

- Information extraction (Canzanese R, 2013)
- Key fields selection through Recursive Feature Elimination method (RFE) (Isabelle Guyon, 2002)
- Scaling for appropriate weights
- Redundancy elimination through Singular Value Decomposition method (SVD) (Christopher D.Manning, 2008)

The step will output a vector of numeric data $\hat{x}$for feeding as input to the detection process and for getting high accuracy in the detection of known and unknown malware. The frequency of system calls is counted in the first step for the extraction of information from binary files. The key- field selection process continues in the next step by fetching consecutive sequence of n-system calls from the numerical vector. Vector encoding technique is used for the conversion of binary file into a vector of word frequencies.

Table 1. Hyper call tracing and information extraction

Call 1	**Call 2**	**Ordered**	**Unordered**
NtAllocateVirtualMemory	NtFreeVirtualMemory	1	2
NtAllocateVirtualMemory	NtQuerySystemInformation	1	2
NtAllocateVirtualMemory	NtQueryVirtualMemory	1	1
NtClose	NtAllocateVirtualMemory	1	1
NtCreateSection	NtMapViewOfSection	1	1
NtFreeVirtualMemory	NtAllocateVirtualMemory	1	-
NtFreeVirtualMemory	NtOpenDirectoryObject	1	1
NtOpenDirectoryObject	NtOpenDirectoryObject	1	1
NtOpenDirectoryObject	NtOpenSymbolicLinkObject	1	1
NtMapViewOfSection	NtQuerySection	1	1
NtOpenFile	NtCreateSection	1	1
NtOpenFile	NtQueryVolumeInformationFile	1	1
NtOpenSymbolicLinkObject	NtQuerySymbolicLinkObject	1	1
NtProtectVirtualMemory	NtProtectVirtualMemory	1	1
NtProtectVirtualMemory	NtQueryInformationProcess	2	3
NtQueryAttributesFile	NtOpenFile	1	1
NtQueryInformationProcess	NtProtectVirtualMemory	1	-
NtQueryInformationProcess	NtQueryInformationProcess	3	3
NtQueryInformationProcess	NtQuerySystemInformation	1	1
NtQueryPerformanceCounter	NtProtectVirtualMemory	1	1
NtQuerySymbolicLinkObject	NtClose	1	1
NtQuerySystemInformation	NtAllocateVirtualMemory	1	-
	Total	25	25

Now the data will be in the form of hyper-call sequences with the ordered and unordered criteria of call frequencies. Ordered criteria lists out the order of issue of hyper-call for every instance, this is not considered in unordered criteria. In the above table 1, the '-' symbol for unordered criteria refers the reordering of another sequence of hyper-calls. Next the unnecessary fields are eliminated from the output vector $\hat{x}$ through RFE method. The method considers all the fields for analyzing relationships for the elimination of the unnecessary fields as a whole. The eliminated data will be able to perform well with the SVM-SGD algorithm. Scaling of selected key field data based on the call frequency weight is the next step. Excess flow of system calls in the event of any abnormal event in the system can be identified by call frequency. The scaled system calls vector $\hat{x}$ is performed with redundancy elimination through SVD method. The features in the call vector is projected into a low dimensional feature space. The method needs training for finding the suitable projections from redundant data in reduction process. The feature vectors xi of the pre-calculated vector $\hat{x}$is framed as a matrix X by SVD method.

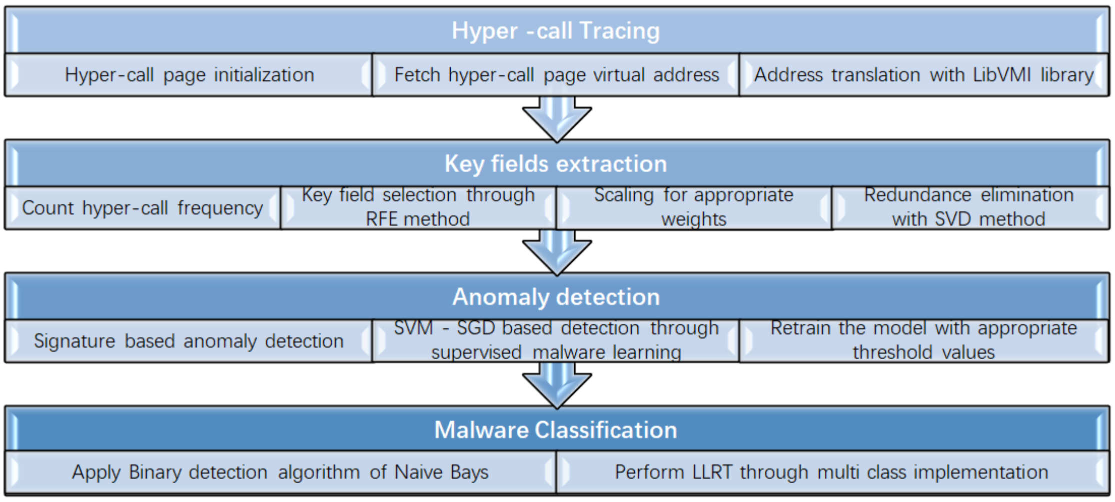
Hyper -call Tracing
Hyper-call page initialization
Fetch hyper-call page virtual address
Address translation with LibVMI library
Key fields extraction
Count hyper-call frequency
Key field selection through RFE method
Scaling for appropriate weights
Redundance elimination with SVD method
Anomaly detection
Signature based anomaly detection
SVM - SGD based detection through supervised malware learning
Retrain the model with appropriate threshold values
Malware Classification
Apply Binary detection algorithm of Naive Bays
Perform LLRT through multi class implementation

Fig.2. Hyper-calls analysis in the nested hypervisor environment

Hence X=[x1,x2,.. xn], xi∈$\hat{x}$. Now matrix factorization techniques are applied for finding the approximation of xi in $\hat{x}$. Here the transformation function applied is

$$x_k = U\Sigma_k V^T \tag{1}$$

Here the rank k approximation of x is computed through the projection of k-item feature vector x_k. U and V are the intermediary matrices and V^T denotes the transpose of matrix V. Here the value of k is taken to eliminate redundancy and to preserve key features from the original call vector.

3.3. *Anomaly Detection*

Malware detection is performed with two methods of signature based detection and SVM-SGD detection for achieving better results through comparative analysis. In the case of Signature based detection n-number of system call traces is compared with malware signature set. Only those calls that are found in the known malware signatures will be reported. The algorithm computes the call frequency and compares it with a threshold λ for reporting it.

$$S^T\hat{x} > \lambda \tag{2}$$

Here S means a binary vector to denote malware signatures. The value of threshold is chosen for fine tuning the detection accuracy. But this method cannot be applied for new malware samples. Hence we move towards supervised learning with linear Support Vector Machines (SVMs) as the model works with a hyper plane $w^T\hat{x}$ to classify data points on benign and malware category. Here, w denotes the hyper plane of the labelled training data. The weighed sum $w^T\hat{x}$is compared with a optimum threshold value λ for the detection process in the feature vector $\hat{x}$through Stochastic Gradient Descent (SGD) algorithm [33]. A process is found to be malicious if

$$w^T\hat{x} > \gamma \tag{3}$$

In the multi-tenant data environment of cloud, SGD is applicable for learning problems with large-scale datasets. Here, an objective function is used with the feature vectors $\hat{x}$i∈X and with the labels $y_i \in \{-1,1\}$. Malware call traces is labelled as 1 and benign calls labelled as -1 with yi. Here, one loss function L is used for penalization of detection errors and regularization constant α is applied for the penalization of high model complexity. Hard margin SVMs are highly sensitive to outliers than that of soft margin SVMs. But use of soft margin SVMs allow mislabelled training data. In this case the level of malicious activity is determined through a hinge loss L and a threshold value γ . The objective function is

$$E(w) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, w^T\hat{x}_i) + \alpha||w||_2 \tag{4}$$

Here L is computed as,

$$L(t, y) = max(0,1 - ty) \tag{5}$$

Each training vector x is treated individually in the SGD algorithm for approximating the gradient of E(w) through random iterations on training data X. The SGD algorithm works by an adaptive learning rate ηt, for updating the weight of every training sample continuously. The learning rate ηtis used to determine estimated gradient which is decaying gradually and contributing to the feature weights. The value of learning rate is

$$\eta t = \frac{1}{\alpha(t_0+t)} \tag{6}$$

Here, t_0 is initial value and t is current training iteration. The training samples in X are ordered randomly between iterations for updating the weights w in the algorithm with approximate gradient and learning rate with i=t mod N.

$$w_{t+1} = w_t - \eta t\left(\frac{\partial L(y_i, w^T\hat{x}_i)}{\partial w} + \alpha\frac{\partial(||w||_2)}{\partial w}\right) \tag{7}$$

The SGD algorithm can handle large volume of training samples from high dimensional and sparse datasets in cloud. The SVM detector treats the data as linearly separable. Retraining of the models is performed for the handling of changes in normal and malicious software. The computational complexity of standard SVM algorithm is O(n^3) for the training of n number of samples. Initially the cubic complexity of SVM based anomaly detection is worse in handling large datasets. In the testing phase with d number of input dimensions, computational complexity becomes O(d^2).

3.4. Malware Classification

The same input hyper call traces are fed into malware classification process immediately after detection. Here, identified malicious activity is classified as existing malware category. Naïve Bayes algorithm (LLRT) is a binary detection algorithm used for the context of malware classification through supervised learning. The algorithm performs Log Likelihood Ratio Test (LLRT) through multi class implementation; here learning process is carried over labelled training data. According to the Bayes' classification the given feature vector $\hat{x}$ is checked to belong to a class C_k. The proportionality of posterior probability is calculated from the product of likelihood of belong to a class C_k and prior probabilities as in the following equation.

$$P(C_k|\hat{x}) \propto P(C_k) \times P_{\hat{X}|C_k}(\hat{x}) \tag{8}$$

The estimations on training data is used for fetching prior probabilities P(C_k). Most likely class label $\hat{C}_k$for the feature vector x̂ from naïve Bayes classifier is computed from the class with the maximum posterior probability.

$$\hat{C}_k = \arg_{C_k} \max P(C_k) \times P_{\hat{X}|C_k}(\hat{x}) \tag{9}$$

This work has used the Gaussian implementation of the naïve Bayes classifier. The posterior probabilities $P_{\hat{X}|C_k}(\hat{x})$are modelled using the Gaussian distribution which is applied in the data of Singular Value Decomposition (SVD) transformed feature vector. The algorithm has the benefit of being simple to train and test the malware sample.

4. Implementation

The work got implemented with Xen virtualization platform with two guest machines loaded into the hypervisor. One of the guest functions as web server as zone 2 and the other takes the role of attacker as zone 1. The version of Xen Server and the client systems are 6.1.0-59235P and 6.1 (build 664) respectively. The guest operating systems are Ubuntu 18.04 and Ubuntu 16.04 and allocated with 4GB of RAM, 1 virtual CPU and 16GB of storage. A Loadable Kernel Module (LKM) is available in the attacker guest which initiates hyper-call sequences to allow for execution from ring 1 of guest kernel space. The DDoS attack scenario is created by the XOIC attacker tool and CPU death ping library tools. The system observes network traffic through Snort sensor devices from appropriate locations. The following figure 3 depicts the attack scenario with the tool installed in VM3 of zone 1 with death ping library agents set in VM1 and VM2. Each agent targets attack on a VM in Zone2. The XOIC tool floods the VMs in Zone2 by TCP and UDP packets and the CPU death ping libraries flood the VMs with ICMP and HTTP requests. The XOIC tool provides an easy-to-interact console that one can directly launch the attack and observe the system status accordingly. The host response is observed with different parameters of series of hyper-calls. Eventually this will trigger overutilization of CPU in the host through an unreported hyper-call or parameter combination. Based on the methodology this attack comes under domain2 of host resource exhaustion.

In the second phase the system is trained against the real traffic dataset namely CIC-DDoS2019 dataset [34]. The dataset is chosen because of its implementation on TCP/IP communication stack that contains DDoS attacks. The dataset contains 56,863 rows of benign traffic mixed with 50,006,249 rows of DDoS attacks with a total of 50,063,112 rows of traffic with 86 features. There are 12 types of DDoS attacks enclosed in training data and 7 types of DDoS attacks in testing data.

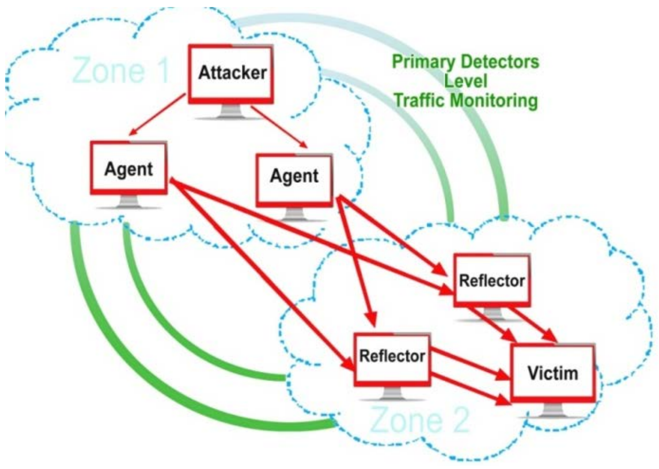

Fig.3. DDoS attack generation in Guest 1 with XOIC tool

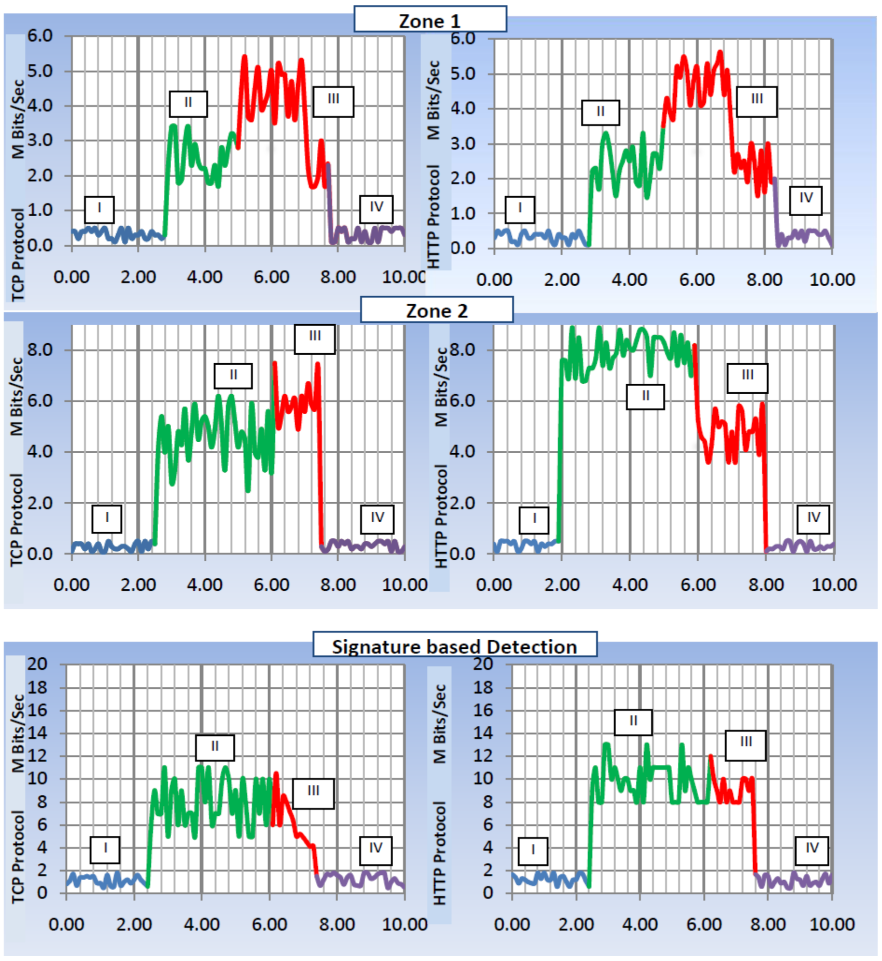

Fig.4. TCP traffic caused by XOIC and the HTTP traffic from CPU death PING libraries

5. Results and Discussion

In the first phase, traffic level of Snort sensor devices is monitored in the VMs of Zone2 for the evaluation of detection accuracy. The above figure 4 shows the TCP traffic rise caused by the DDoS attacks from the XOIC tool and the HTTP traffic of the CPU death PING libraries. The next figure 5 shows the UDP traffic caused by XOIC and the ICMP traffic from CPU death PING libraries. The attack lasts for about 10 minutes. The graph shows the traffic in different levels of attack progress and detection. The first level shows the system's response in normal traffic conditions. The next level II, attack begins and the traffic rate increases. At this stage, the system observed to be in hanging state and unable to respond to normal user requests. In level III, the system reports the attack and the malicious traffic get blocked. In the final level, one can again observe the system with normal traffic.

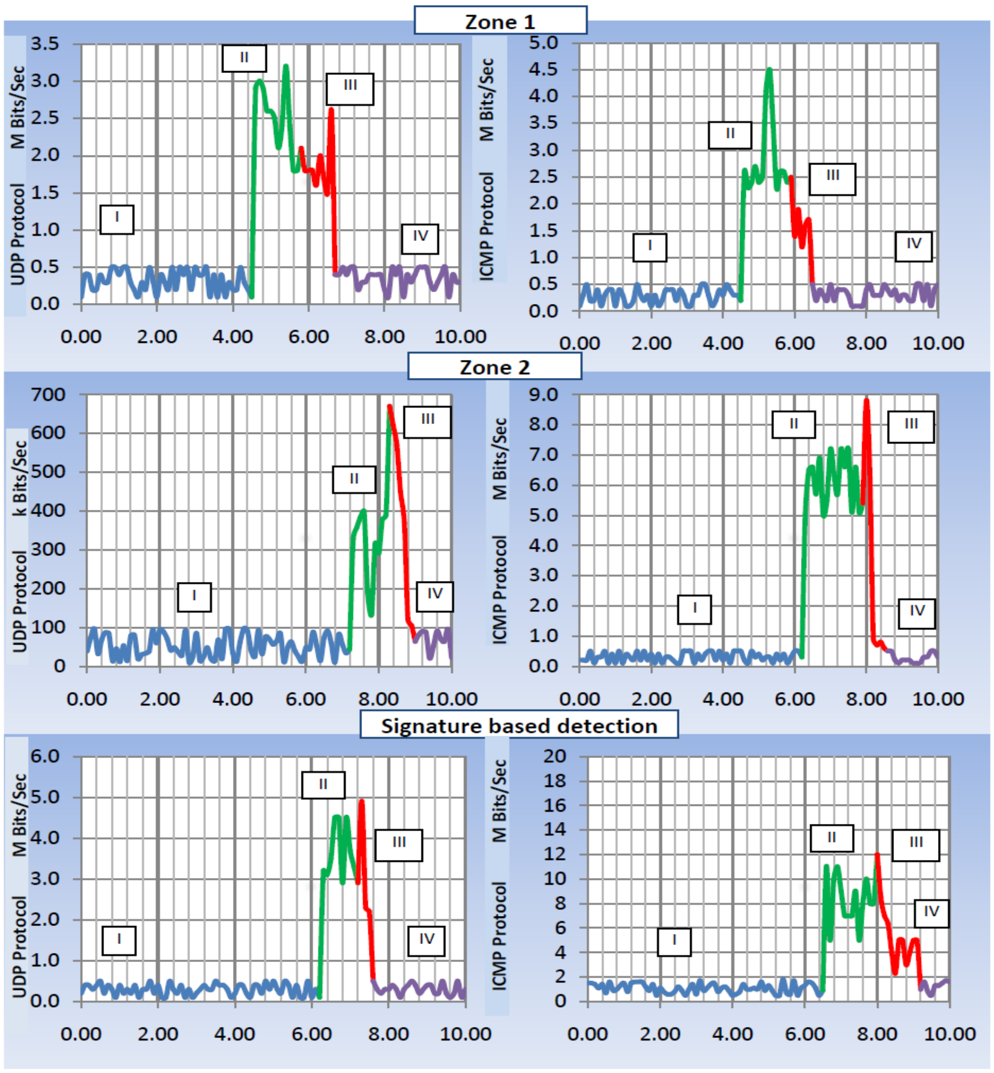

Fig.5. UDP traffic caused by XOIC and the ICMP traffic from CPU death PING libraries

It can be observed from Tables 2 and 3 that the system's performance with hyper-calls analysis is found to produce better statistic results. Comparison of detection statistics generates more detailed evaluation results for the two algorithms as shown in the tables. It can be seen that the SVM-SGD detection method generates high precision results with low false alarm rate than that of signature based detection approach for hyper-calls analysis. It can be seen that the time complexities of the two approaches are based on the data size. In the first day of training data, the time elapsed is relatively smaller than that of flow records having heavy traffic. For the evaluation purpose, the confusion matrix, the detection rates and false alarm rates are taken as basic performance criteria than any other parameters. This is the choice generally used for assessing any kind of intrusion detection system in realizing the difference in cost of false positives with that of false negatives. For example, in the case of an unknown flow, which is not labelled in a heavy traffic, it is of lower cost than that of a wrongly labelled attack or normal flow for a time window. For instance, there is no labelled attacks on day 1, the performance is measured by analysing the detection rate of the normal flow or confusion matrix values even if the misclassification of unknown flows as normal. On day 2, with a sample of the infiltration attack on distinction ports used for training the model, the detection rate of attack and normal data is high for the methods. UDP-lag attack observed similarly with the peak detection rate on the day 2. Since, there is no attack on the day 1, the attack data values seem to be nil on the day.

For the DDoS case study, we used separate tools for launching the attacks in zone 2 from zone 1. The system is observed to handle the DDoS attacks efficiently by identifying and blocking the malicious traffic in relatively less time.

Finally, while comparing the two deployment models for alert detection rate, the model with Hyper-calls analysis reporting 36.8% large number of alerts than that of signature based analysis. On the other side, the basic signature based model generates alerts with low delay of 32.3% in comparison with the Hyper-calls analysis model because of the reduction in the number of alerts to be processed. Thus the proposed system is proved to be the best model in detection of known and unknown malware instances in both data from the CIC-DDoS2019 dataset and that of tools generated attack data.

Table 2. Day wise comparison of Detection methods

	Detection Rate of Normal Traffic		**Detection Rate of Unknown Traffic**	
Detection Method	Day 1	Day 2	Day 1	Day 2
SVM-SGD	0.94	0.94	0.92	0.94
Signature Based Detection	0.91	0.91	0.99	0.97

	Detection Rate of Attack Traffic		**False Alarms Rate of Normal Traffic**	
Detection Method	Day 1	Day2	Day1	Day 2
SVM-SGD	0.99	0.82	0.00	0.21
Signature Based Detection	0.85	0.24	0.03	0.58

	False Positive Rate of Attack Traffic		**Time Complexity (in Epochs)**	
Detection Method	Day 1	Day 2	Day 1	Day 2
SVM-SGD	0.01	0.01	2.20	2.31
Signature Based Detection	0.06	0.23	0.52	0.41

Table 3. The Performance observed from experimental results of detection methods relative to benign and various types of attacks in CIC-DDoS2019 dataset

Detection Method / Attack	SVM SGD %	Signature Based Detection %
TNR (BENIGN)	100	100
DrDoS_DNS	75	47
DrDoS_LDAP	65	49
DrDoS_SNMP	73	55
DrDoS_SSDP	63	58
DrDoS_UDP	53	48
DrDoS_NetBIOS	95	78
DrDoS_MSSQL	64	56
Syn	72	65
TFTP	100	98
DrDoS_NTP	96	89
WebDDoS	32	20
UDP-lag	99	89

6. Conclusions

In this paper a malware detection and classification system is proposed for the DDoS type of attacks in the cloud environment. The system makes use of hyper-calls analysis through hypervisor based introspection (HVI) for the detection of sophisticated DDoS attacks in both the tools generated real-time attack environment and in CIC-DDoS2019 dataset. The detection algorithm of SVM-SGD is compared with signature based detection to report DDoS attacks with high accuracy and low false positive rate in the heavy network traffic occurring cloud environment. The system has achieved relatively high accuracy rate of 99.6% compared with the signature-based detection method which is only 67.22% of accuracy on the average. Also the system is proved to report 36.8% large number of alerts than that of signature based analysis. The system's performance is found to be better by achieving high degree of accuracy than the state-of-the-art detection approaches as it has finely applied both the detection and classification algorithms after performing hyper-calls analysis.

References

[1] Aditya Singh, Thesis "System Call Analysis and Visualization".

[2] ZechengHe,Ruby B. Lee, "Machine Learning Based DDoS Attack Detection From Source Side in Cloud", in *IEEE 4th International Conference on Cyber Security and Cloud Computing*, 2017.

[3] National Vulnerability Database (NVD). CVE-2017- 8903. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-8903, 2017

[4] Amir F. Mukeri, Dwarkoba P. Gaikwad, " Adversarial Machine Learning Attacks and Defenses in Network Intrusion Detection Systems", *International Journal of Wireless and Microwave Technologies*, Vol.12, No.1, pp. 12-21, 2022.

[5] Naila Samad Shaikh, Affan Yasin, Rubia Fatima, "Ontologies as Building Blocks of Cloud Security", *International Journal of Information Technology and Computer Science*, Vol.14, No.3, pp.52-61, 2022.

[6] Milenkoski, B. D. Payne, N. Antunes, M. Vieira, and S. Kounev, "Experience Report: An Analysis of Hypercall Handler Vulnerabilities," *in IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)* 2014, 2014, pp. 100-111.

[7] S. Lee, G. Kim, and S. Kim, "Sequence-order-independent network profiling for detecting application layer DDoS attacks," in *EURASIP Journal on Wireless Communications and Networking*, vol.2011, no.1, article no.50, 2011.

[8] Y.G.Dantas, V.Nigam, and I.E.Fonseca, "A Selective Defense for Application Layer DDoS Attacks," in *Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference (JISIC)*, pp.75–82,TheHague, Netherlands, September2014.

[9] J.Choi, C.Choi, B.Ko, and P.Kim,"A method of DDoS attack detection using HTTP packet pattern and rule engine in cloud computing environment," *Soft Computing*, vol. 18, no. 9, pp. 1697–1703,2014.

[10] Z.He, T.Zhang, and R.B.Lee, "Machine Learning Based DDoS Attack Detection from Source Side in Cloud," in *Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pp. 114–120, New York, NY, USA,June 2017.

[11] I. Sreeram and V. P. Vuppala, "HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm," *Applied Computing and Informatics*, 2017.

[12] I. Sofi, A. Mahajan, and V. Mansotra, "Machine learning techniques used for the detection and analysis of modern types of ddos attacks", *learning*, vol.4, no.06,2017.

[13] Bharadwaja S, Sun W, Niamat M, Shen F, "Collabra: a xen hypervisor based collaborative intrusion detection system", in *2011 eighth international conference on information technology: new generations (ITNG)*, IEEE, New York, 2011, pp 695–700.

[14] Maiero C, Miculan M, "Unobservable intrusion detection based on call traces in paravirtualized systems", in *2011 proceedings of the international conference on security and cryptography (SECRYPT)*, IEEE, New York, 2011, pp 300–306.

[15] Wu J, Ding L, Wu Y, Min-Allah N, Khan SU, Wang Y, "C2detector: a covert channel detection framework in cloud computing", *Secur Commun Netw* 7(3), 2014, pp 544–557.

[16] Le CHH, "Protecting xen hypercalls", *PhD thesis, Universiti of British Columbia*, Vancouver, 2009.

[17] Wang F, Chen P, Mao B, Xie L, "Randhyp: preventing attacks via xen hypercall interface", in *Information security and privacy research, Springer*, Berlin, 2012, pp 138–149.

[18] Chonka, Y. Xiang, W. Zhou, and A. Bonti, "Cloud security defence to protect cloud computing against HTTP-DoS and XML-DoS attacks", *Journal of Network and Computer Applications*, vol. 34, 2011, pp. 1097-1107.

[19] J. Bacon, D. Eyers, T. F. J. M. Pasquier, J. Singh, I. Papagiannis, and P. Pietzuch, "Information Flow Control for Secure Cloud Computing", in *IEEE Transactions on Network and Service Management*, vol. 11, 2014, pp. 76-89.

[20] K. Hashizume, D. Rosado, E. Fernández-Medina, and E. Fernandez, "An analysis of security issues for cloud computing", *Journal of Internet Services and Applications*, vol. 4, 2013/02/27, 2013, pp. 1-13.

[21] Z. Wang, X. Jiang, W. Cui, and P. Ning, "Countering kernel rootkits with lightweight hook protection", *in Proceedings of the 16th ACM conference on Computer and communications security*, Chicago, Illinois, USA, 2009.

[22] Chen, T. Garfinkel, E. C. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, et al., "Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems", *ACM SIGARCH Computer Architecture News*, vol. 36, 2008, pp. 2-13.

[23] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization", *in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, Cascais, Portugal, 2011.

[24] P. Colp, M. Nanavati, J. Zhu, W. Aiello, G. Coker, T. Deegan, et al., "Breaking up is hard to do: security and functionality in a commodity hypervisor", *in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, Cascais, Portugal, 2011.

[25] C. Hoang, "Protecting Xen Hypercalls: Intrusion Detection/Prevention in a Virtualized Environment", *MSc, Computer Science, University of British Columbia*, 2009.

[26] W. Jingzheng, D. Liping, L. Yuqi, N. Min-Allah, and Y. Wang, "XenPump: A New Method to Mitigate Timing Channel in Cloud Computing," *in IEEE 5th International Conference on Cloud Computing (CLOUD)* 2012, pp. 678-685.

[27] C. Yu, L. X. Li, K. Wang, and W. T. Yu, "Protecting the Security and Privacy of the Virtual Machine through Privilege Separation," *Applied Mechanics and Materials*, vol. 347-350, pp. 2488-2494, August 2013.

[28] C. Li, A. Raghunathan, and N. K. Jha, "Secure Virtual Machine Execution under an Untrusted Management OS," in *IEEE 3rd International Conference on Cloud Computing*, 2010, pp. 172-179.

[29] Milenkoski, B. D. Payne, N. Antunes, M. Vieira, and S. Kounev, "HInjector: Injecting Hypercall Attacks for Evaluating VMI-based Intrusion Detection Systems," *in Annual Computer Security Applications Conference (ACSAC)*, 2013.

[30] Isaac Kofi Nti, Owusu Nyarko-Boateng, Justice Aning, "Performance of Machine Learning Algorithms with Different K Values in K-fold Cross-Validation", *International Journal of Information Technology and Computer Science*, Vol.13, No.6, pp.61-71, 2021.

[31] Pushpam Kumar Sinha, "Modifying one of the Machine Learning Algorithms kNN to Make it Independent of the Parameter k by Re-defining Neighbor" *I. J. Mathematical Sciences and Computing*, vol.4, pp.12-25, August 2020.

[32] Samuel Ndichu, Sylvester McOyowo, Henry Okoyo, Cyrus Wekesa, "A Remote Access Security Model based on Vulnerability Management", *International Journal of Information Technology and Computer Science,* Vol.12, No.5, pp.38-51, 2020.

[33] Tong Zhang, "Solving Large Scale Linear Predition Problems Using Stohasti Gradient Descent Algorithms", in *ICML 2004: Proceedings of the Twenty-First International Conference on Machine Learning.* Omnipress, 919-926, 2004.

[34] Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A., "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy" in *Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST),* Chennai, India, 1–3 October 2019; pp. 1–8.

Authors' Profiles

Ms. K. Uma Maheswari is an Assistant Professor in the Department of Computer Science in Bharathi Women's College, Chennai, India. She received her Ph.D. from Bharathiar University, Coimbatore in 2019. She obtained her Master's degree from Bharathidasan University, Tiruchirappalli, India in 2004. She has around 12 years of teaching experience. Her areas of research interest are Digital Forensics and Security in Cloud Environment.

Ms. Nalini Subramanian is an Assistant Professor in the Department of Information Technology in Rajalakshmi Engineering College, India. She received her Ph.D. from Sathyabama Institute of Science and Technology in 2020. She obtained her M.E degree in Computer Science & Engineering from Sathyabama University, Chennai, India in 2006. She has 17 years of teaching experience. Her area of research interest includes cloud computing, network security and machine learning.

Mr. Manikandan Subramaniyan was born in Tamil Nadu, India, in 1979. He received Ph.D (2019),ME (2002) and BE (2001) degrees from the Sathyabama Institute of Science and Technology, Annamalai University and Madras University, respectively. Currently he is working as lecturer in Electrical Power section, Engineering Department, University of technology and Applied Science - IBRI, Sultanate of OMAN. His research interest includes network security and cloud security.

How to cite this paper: K. Umamaheswari, Nalini Subramanian, Manikandan Subramaniyan, "Distributed Denial of Service Attack Detection Using Hyper Calls Analysis in Cloud", International Journal of Computer Network and Information Security(IJCNIS), Vol.15, No.4, pp.61-71, 2023. DOI:10.5815/ijcnis.2023.04.06