International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 4 | ISSN : 2456-3307

Two-Player Security Game Approach Based Co-Resident Dos Attack Defence Mechanism for Cloud Computing

***Rethishkumar S.*[1], *Dr. R. Vijayakumar* [2]**

[1]Research Scholar, School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India
[2]Professor, School of Computer Sciences, Mahatma Gandhi University, Kottayam, Kerala, India

ABSTRACT

Virtual Machines (VM) are considered as the fundamental components to cloud computing systems. Though VMs provide efficient computing resources, they are also exposed to several security threats. While some threats are easy to block, some attacks such as co-resident attacks are much harder even to detect. This paper proposes two-player game approach based defense mechanism for minimizing the co-resistance DOS attacks by making it difficult for attackers to initiate attacks. The proposed defense mechanism first analyzes the attacker behavior difference between attacker and normal users under PSSF VM allocation policy. Then the clustering analysis is performed by EDBSCAN (Enhanced Density-based Spatial Clustering of Applications with Noise). The partial labeling is done based on the clustering algorithm to partially distinguish the users as legal or malicious. Then the semi-supervised learning using Deterministic Annealing Semi-supervised SVM (DAS3VM) optimized by branch and bounds method is done to classify the nodes. Once the user accounts are classified, the two-player security game approach is utilized to increase the cost of launching new VMs thus minimizing the probability of initiating co-resident DOS attack. Thus the security threats can be averted efficiently using the proposed defense mechanism. Experimental results prove that the proposed co-resident DOS attack defense mechanism makes a significant contribution preventing security threats.

Keywords: Co-resident DOS attack, PSSF, EDBSCAN, DAS3VM, branch and bound method.

I. INTRODUCTION

Cloud computing has paved the way for "the long-held dream of computing as a utility" [1]. Commercial third-party clouds allow businesses to avoid over provisioning their own resources and to pay for the precise amount of computing that they require. Virtualization is a key to this model. By placing many virtual hosts on a single physical machine, cloud providers are able to profitably leverage economies of scale and statistical multiplexing of computing resources. While many models of cloud computing exist, the Infrastructure-as-a-Service (IaaS) model used by providers such as Amazon's Elastic Compute Cloud (EC2) service offers a set of virtualized hardware configurations for customers [2]. The sharing of a common physical platform among multiple virtual hosts, however, introduces new challenges to security, as a customer's virtual machine (VM) may be co-located with unknown and untrusted parties. Placement on a common platform entails the sharing of physical resources and leaves sensitive data processed in a cloud potentially vulnerable to the actions of malicious co-residents sharing the physical machine. Researchers have already demonstrated the methods of bypassing co-resident isolation in virtualization middleware, particularly through cache-based side channels [3, 4, 5]. Their results confirm that hypervisors present a new attack surface through which privacy and isolation guarantees can be compromised. However, defenses against such vulnerabilities are already being proposed in the academic literature [6, 7].

Virtual machines (VM) area commonly used resource in cloud computing environments. For cloud providers, VMs help increase the utilization rate of the underlying hardware platforms. For cloud customers, it enables on-demand resource scaling, and outsources the

maintenance of computing resources. However, apart from all these benefits, it also brings a new security threat [8]. In theory, VMs running on the same physical server (i.e., co-resident VMs) are logically isolated from each other [9]. In practice, nevertheless, malicious users can build various side channels to circumvent the logical isolation, and obtain sensitive information from co-resident VMs, ranging from the coarse-grained, e.g., workloads and web traffic rates, to the fine-grained, e.g., cryptographic keys. For clever attackers, even seemingly innocuous information like workload statistics can be useful. For example, such data can be used to identify when the system is most vulnerable, i.e., the time to launch further attacks, such as Denial-of-Service attacks.

Hence in this paper, a two-player security game approach has been proposed to defend the VMs against co-resident DOS attacks. Game based security approaches have extensively employed as an efficient scheme for defense mechanisms. In [10] a game theoretical approach has been developed to defend against co-resident attacks. However, that approach does not consider the impact of the datacenter size in attack defense. Likewise the semi-supervised learning based classification approach does not provide optimal solutions. Inorder to overcome these limitations, the proposed two-player security game approach based defense mechanism considers the size of data center. As the DBSCAN clustering approach incurs time complexity, the Enhanced DBSCAN (EDBSCAN) is utilized which is followed by DAS3VM with branch and bound methods for find globally optimal solutions. Finally the two-player game model is used to increase cost and difficulty of launching attacks, thus forcing the attackers to behave as normal users. The remainder of the paper is organized as follows: section 2 describes various past researches related to this research work. Section 3 explains the proposed defense mechanism. Section 4 presents the evaluation results of the proposed model while the section 5 makes a conclusion about this research work.

II. RELATED WORKS

Yinqian Zhang et al [11] described about Home Alone, a system that lets a tenant verify its VMs' exclusive use of a physical machine. The key idea in Home Alone is to invert the usual application of side channels. Rather than exploiting a side channel as a vector of attack, Home Alone uses a side-channel (in the L2 memory cache) as novel, defensive detection tool. By analyzing cache usage during periods in which "friendly" VMs coordinate to avoid portions of the cache, a tenant using Home Alone can detect the activity of a co-resident "foe" VM. Key technical contributions of Home Alone include classification techniques to analyze cache usage and guest operating system kernel modifications that minimize the performance impact of friendly VMs sidestepping monitored cache portions. Their implementation of Home Alone on Xen-PVM requires no modification of existing hypervisors and any special action or cooperation by a cloud provider.
Adam Bates et al [12] explain about Co-resident watermarking, a traffic analysis attack that allows a malicious co-resident VM to inject a watermark signature into the network flow of a target instance. This watermark can be used to exfiltrate and broadcast co-residency data from the physical machine, compromising isolation without reliance on internal side channels. As a result, this approach is difficult to defend without costly underutilization of the physical machine. It evaluates co-resident watermarking under a large variety of conditions, system loads and hardware configurations, from a local lab environment to production cloud environments (Future grid and the University of Oregon's ACISS). The ability to initiate a covert channel of 4 bits per second is demonstrated, and it can confirm co residency with a target VM instance in less than 10 seconds. It also shows that passive load measurement of the target and subsequent behavior profiling is possible with this attack. The final investigation demonstrates the need for the careful design of hardware to be used in the cloud.

Yi Han et al [13] concentrates on the co-resident attack, where malicious users aim to co-locate their virtual machines (VMs) with target VMs on the same physical server, and then exploit side channels to extract private information from the victim. Most of the previous work has discussed how to eliminate or mitigate the threat of side channels. However, the presented solutions are impractical for the current commercial cloud platforms. This method approaches the problem from a different perspective, and study how to minimize the attacker's possibility of co-locating their VMs with the targets, while maintaining a satisfactory workload balance and low power consumption for the system. Specifically, it introduces a security game model to compare different VM allocation policies.

Similarly, there has been many techniques developed for the security of cloud computing. However, most approaches focused on eliminating the side channels for avoiding attacks. But the clever attackers overcome these hurdles and initiate the attacks. Hence more advanced defense systems are needed to be included in this research.

III. PROPOSED METHODOLOGY

In the initial stages of the proposed methodology, the analysis of the attacker behavior is done and compared with that of the legitimate user. The attacker behavior is analyzed in three different scenarios namely under no security mechanism, with presence of PSSF VM allocation policy and the proposed defense mechanism. Under no defense mechanism, the behavior of the normal users can be found. From the attacker's perspective, no matter how they start VMs, it is clearly preferable that they can spread their VMs across a wider range of servers, in order to increase the probability of co-locating with their targets. Under PSSF policy, the new VMs will be allocated to lightly loaded servers. In addition, all servers are managed in groups in order to decrease the power consumption, and a new VM will not be assigned to one server in a new group until all servers in the currently active groups are fully utilized. PSSF does not differentiate between users, and treats all VM requests equally. Consequently, if the attacker keeps creating new accounts, each of which starts one and only one VM, PSSF becomes less effective although even in this worst case scenario. The only option for an attacker is by creating new accounts. But as this option can be utilized without any constraints, the attacker continuously initiates new VMs by creating new accounts. Hence the proposed defense mechanism focuses on making life difficult for the attacker by increasing the cost for creation of new accounts. This makes the possibility of initializing new VMs minimum and hence forces the attacker to behave as normal user. Figure 1 shows the overall architecture of the proposed defense mechanism

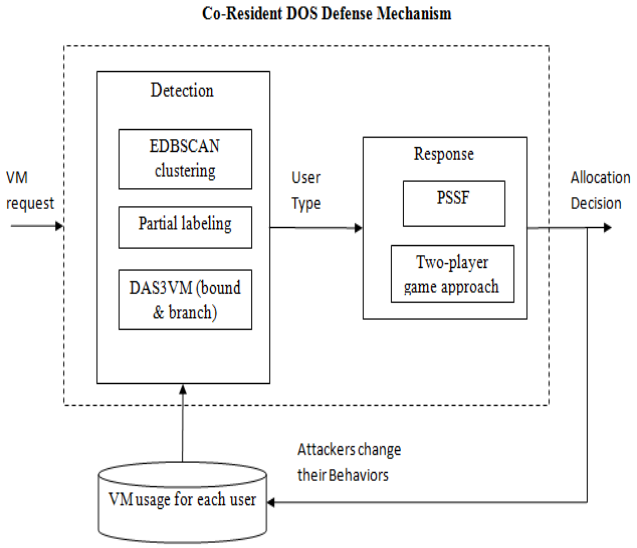

Figure.1. Proposed Defense Mechanism Architecture

The defense mechanism consists of two main parts: the detection module and the response module. When a user requests to start a new VM, the detection module loads historical data from the database, classifies the user into one of the three types: low/medium/high risk, and sends the result to the response module. The latter limits the available servers according to the result, so that VMs of any user only co-locate with VMs started by users of the same type. In addition, the response module sets the parameter in the VM allocation policy PSSF (also based on the classification result), which finally selects a server within the limited set. It is likely that attackers will adapt the way they start VMs according to the allocation decisions. Meanwhile, the defense system will also update the database of VM usage regularly, which in turn may change the allocation decision. Therefore, in order to analyze how to optimize this adversarial learning problem, we model the problem as a two-player security game (*G)* between the attacker (*A)* and the defender (*D)*. Before studying about the security model, let us know about co-resident and DOS attacks.

Co-resident attacks

In compute clouds, the co-resident threat considers a malicious and motivated adversary that is not affiliated with the cloud provider. Victims are legitimate cloud customers that are launching Internet-facing instances of virtual servers to do work for their business. The adversary, who is perhaps a business competitor, wishes to use the novel abilities granted to him by

cloud co-residency to discover valuable information about his target's business. This may include reading private data or compromising a victim machine. It could also include more subtle attacks such as performing load measurements on the victim's server or launching a denial-of-service attack. Masquerading as another legitimate cloud customer, the adversary is free to launch and control an arbitrary number of cloud instances. As is necessary for the general use of any third-party cloud, the cloud infrastructure is a trusted component. Co-residency detection through virtualization side channels is a danger that was first exposed by Ristenpart et al. [3]. This work lays out strategies for exploiting the instance placement routines of the Amazon EC2 cloud infrastructure in order to probabilistically achieve co-location with a target instance. From there, co-residency can be detected using a cross-VM covert channel as a ground truth. While more advanced methods of successful placement are outlined, such as abusing temporal locality of instance launching, it is shown that a brute force approach is also modestly successful. Masquerading as a legitimate customer, an attacker is able to launch many instances, perform the co-residency check, terminate, and repeat until the desired placement is obtained. Several cross-VM information leakage attacks are also outlined, such as the load profiling and keystroke timing attacks. However, independent results confirmed that many of the approaches in previous work, such as the use of naive network probes, are no longer applicable on the EC2.

DOS attacks

DOS attacks try to render the service unavailable to its users. The attack consumes large amounts of system resources such as processing power, memory, and bandwidth. This consumption will leave the service inaccessible to the users or intolerably slow. DOS attacks, and their variant Distributed Denial of Service (DDOS), grab high media attention mostly because of their magnitude. In 1988, only six DDOS attacks took place, as the reports show. DDOS attacks targeted large websites like CNN, Yahoo, and Amazon in the year 2000 with an attack rate of about 1GBps. DDOS attacks reached the rates of 70GBps in the year 2007. In 2013, a huge attack took place on Spamhaus spam detection service that reached highest rate at that time of 300 GBps [14]. Recently, in February 2014, the largest DDOS attack known until now with the rate of 400GBps took place. This attack targeted a public cloud service provider called CloudFlare. Attacks of such magnitude affect not only their targets, but affect the overall Internet in the area. As mentioned in [15], regular Internet users experienced noticeable slowness in their Internet services. DOS attacks can be operated on multiple layers in the network. An attacker can operate a DOS attack at the network level to render the whole server unreachable. This is done by getting the Network Interface Card (NIC) of the server completely occupied with useless packets in such a way that no more bandwidth is available for legitimate users. A DOS attack can be launched in the transport layer using the very old, but still effective, SYN Flood technique. In a SYN flood attack the attacker sends a flood of TCP SYN requests that gets the server busy without actually completing the three-way handshake procedure used in the setup of TCP sessions. DOS attacks can also be launched at the application level by sending fake requests to the application layer protocol to consume the servers' memory and processing power. Sending a flood of fake SMTP requests to an electronic-mail server is a clear example. Many security appliances are currently capable of detecting simple DOS attacks that come from a single attacking node. Thus, DOS attacks have evolved into a more complicated attack called Distributed DOS (DDOS). In DDOS, the DOS attack is performed from multiple sources around the Internet such that it would be harder to trace and block the attacker. More information about DDOS in cloud computing can be found in [16]. Although DDOS take major attention in the media, it is not the only threatening type of DOS attacks. Another type of DOS is Asymmetric application-level DOS attacks that exploit the vulnerabilities in web-servers, databases, as well as other cloud resources. This type of attack allows a malicious attacker to bring down an application using very small attack payload, sometimes as small as 100 bytes. In [17], a method that depends on covariance-matrix was used to detect DOS attacks. This method was proven to be highly effective in detecting DOS attacks that are based on flooding. A new DOS attack along with its counter-measure was introduced in [18]. This attack operates on the application level to detect a network bottle-neck in one of the links and focus on flooding this link.

Co-resident DOS attacks

A special type of DOS attacks called Co-Resident DOS attacks is the combination of co-resident and DOS attack. One research direction has employed Game Theory defense mechanisms in defending the cloud infrastructure against Co-Resident DOS attack. In co-resident DOS attack, the attacker rents a VM inside the public cloud and conducts the DOS from the rented VM onto another VM within the same node. The attacker uses simple tools (like nmap and hping) to deduce the exact location of the VM in the cloud and conduct a DOS attack on the bottle-neck network channel shared among the VMs. In [19] a detection method that is based on game theory defense mechanisms was introduced. A model was suggested to prevent flooding attacks was introduced in [20]. This model can provide the foundation of further study in the topic of DOS flooding attacks prevention.

Defense Action Set

From the attacker action set, it can be found that the attacker is capable of triggering their targets to launch new VMs, capable of compromising a low risk account, and only starts same type of VMs. Hence based on these attacker behaviors, the defense process is defined. The defense process begins with determination of attacker behavior. Then the user accounts are clustered using EDBSCAN and then labeled partially as low, medium or high risks. Finally the accounts are classified using semi-supervised learning- DAS3VM with branch and bound method. Based on the classified results, the cost for the initialization of new VMs through new accounts is maximized making it difficult for the attacker to co-locate the target VMs.

EDBSCAN clustering

DBSCAN clustering has been previously used for clustering the user accounts. There are two parameters in the DBSCAN algorithm, ε and *MinPts*, where ε is the maximum distance between two neighbors, and *MinPts* is the minimum number of points in a cluster. However, once *MinPts* is set, ε can be determined by drawing a *k*-distance graph (*k = MinPts)* as introduced in [21]. In other words, ε can be considered as a function of *MinPts*. *MinPts* should not be too small; otherwise noise in the data will result in spurious clusters. Based on these conditions, the clustering process is carried out. However the issue with the DBSCAN clustering process is the time complexity for the whole process. Hence the Enhanced DBSCAN is proposed to overcome the time complexity.

In EDBSCAN, the parameters Eps and MinPts are determined differently from DBSCAN. The procedure is given in the following algorithm:

Algorithm 1: EDBSCAN

Input: List of points pointList and depth
Output: KD Tree
Function kdtree(pointList, depth)
Step 1: Select axis based on depth so that axis cycles through all valid values (axis=depth mod k)
Step 2: Sort point list and choose median as pivot element
Step 3: Create node and construct sub-trees
node location := median;
leftChild := kdtree(points in pointList before median, depth+1);
rightChild := kdtree(points in pointList after median, depth+1);
Step 4: Repeat Steps 1 - 3 till pointList is empty.

The above algorithm is used in the EDBSCAN algorithm to reduce the time complexity. Let ‘d’ be the distance of a point ‘p’ to its kth nearest neighbor, then the d-neighborhood of ‘p’ contains exactly k+1 points for almost all points ‘p’. The d-neighborhood of ‘p’ contains more than k+1 points only if several points have exactly the same distance ‘d’ from ‘p’ which is quite unlikely. Furthermore, changing ‘k’ for a point in a cluster does not result in large changes of ‘d’. This only happens if the kth nearest neighbors of p for k = 1, 2, 3, …, are located approximately on a straight line which in general is not true for a point in a cluster. For a given k, a function k-dist from the database D is defined by mapping each point to the distance from its kth nearest neighbor. When sorting the points of the database in descending order of their k-dist values, the graph of this function gives some hints concerning the density distribution in the database. This graph is called the sorted k-dist graph. If an arbitrary point ‘p’ is chosen, set the parameter Eps to k-dist(p) and set the parameter MinPts to k, all points with an equal or smaller kdist value will be core points. However, as indicated the k-dist graphs for k > 4 do not significantly

differ from the 4-dist graph and they need considerably more computations. The applicability of value 4 to MinPts was further proved by several proposals [22]. Therefore, the parameter MinPts is set to 4 during experimentations. The 4-dist value of the threshold point is used as the Eps value for DBSCAN. These estimated values were given as input to the DBSCAN algorithm. The time requirement of DBScan algorithm is O(n log n) where n is the size of the dataset and because of this it is not a suitable one to work with large datasets. This when combined with k-distance graph to automatically select MinPts and Eps values, increases to O(n2 log n). The present research work uses a KD Tree (space partitioning tree) to reduce the time complexity to O(log n) time. While using KD-Tree finding k nearest neighbors for each n data point the complexity is O(kn log n). The k value is very negligible and therefore does not make much different and hence the time complexity becomes O(n log n). As now the MinPts and Eps parameters are determined, they are utilized in the clustering of the user accounts.

Partial Labeling: Once the list of clusters is obtained, the next step is to compare them with the attacker's potential behaviors, and mark those clusters that are highly likely to be malicious or legal. After the labeling process is completed, the semi-supervised learning method is employed for user account classification.

DAS3VM

The last step of the classification is to apply semi-supervised learning techniques on the partially labeled dataset obtained from the previous step. The algorithm has three parameters. The regularization parameters λ controls the tradeoff between maximizing the hyper plane margin, and minimizing the misclassification rate. The second parameter λ' controls the influence of unlabelled data, and reflects the confidence in the cluster assumption. The third parameter *r* is estimated based on the clustering result. In order to classify each node into one of the three types (low, medium or high risk), the "one-vs-all" approach is adopted:

- For each of the three types of labels − *H1*, *H2* and *L*, we build the corresponding SVM, and then use it to test all the nodes.
- Each node is given three scores −*SH*1, *SH*2 and *SL*. A node is finally labeled as *H1* (*H2*, *L)* if and only if *SH*1 (*SH*2, *SL)* is positive while the other two scores are negative. Otherwise, the node is labeled as *M* (medium risk).
- Nodes labeled as *H1* and *H2* are combined to *H*.

This process of account classification is very efficient as the account classification is highly accurate. But it cannot be termed as 100% accurate, as the clever attackers will adapt their behaviors accordingly. However, it should be noted that the objective of the proposed classification approach is not accurately labeling high risk accounts, but rather carefully choosing the hyper plane corresponding to *L*, so that it is difficult or expensive for attackers to be classified as low risk. Once this objective is achieved, the two-player security game approach is evaluated. However, the classification scheme provides sub-optimal solutions which are needed to be enhanced inorder to avoid maximum activities of clever attackers. Hence the semi-supervised learning method is applied with branch and bound method [23] for obtaining globally optimal solutions.

Branch and Bound for DAS3VM

The classification function f corresponding to L over the space χ, where χ is usually discrete has to minimized. A branch and bound algorithm has two main ingredients:

Branching: the region χ is recursively split into smaller sub-regions. This yields a tree structure where each node corresponds to a sub-region.

Bounding: consider two (disjoint) sub-regions (i.e. nodes) A and $B \subset \chi$. Suppose that an upper bound on the best value of f over A is known and a lower bound (say b) on the best value of f over B is known and that a < b. Then, there is an element in the subset A that is better than all elements of B. So, when searching for the global minimize, safely discard the elements of B from the search: the sub-tree corresponding to B is pruned.

The aim is to minimize the classification function over all 2^u possible choices for the vector y_u, which constitute the set χ introduced above. The binary search tree has the following structure. Any node corresponds to a partial labeling of the data set and its two children correspond to the labeling of some unlabeled point. One can thus associate with any node

a labeled set L containing both the original labeled examples and a subset S of unlabeled examples $\{(xj, yj)\}_{j \in S \subset [l+1 \ldots n]}$ to which the labels yj have been assigned. One can also associate an unlabeled set U = [l + 1 . . . n] \ S corresponding to the subset of unlabeled points which have not been assigned a label yet. The size of the sub-tree rooted at this node is thus $2^{|U|}$. The root of the tree has only the original set of labeled examples associated with it, i.e S is empty. The leaves in the tree correspond to a complete labeling of the dataset, i.e. U is empty. All other nodes correspond to partial labeling.

Concerning the upper bound, it is decided to have the following simple strategy: for a leaf node, the upper bound is simply the value of the function; for a non leaf node, there is no upper bound. In other words, the upper bound is the best objective function found so far. The set A is the leaf corresponding to the best solution found so far and the set B is the sub-tree that is considering to be explored. Because of this choice for the upper bound, a natural way to explore the tree is a depth first search. Indeed it is important to go to the leaves as often as possible in order to have a tight upper bound and thus perform aggressive pruning. Let $D(\alpha, yU)$ be the dual objective function, where y_U corresponds to the labels of the unlabeled points which have not been assigned a label yet,

$$D(\alpha, yU) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(K(x_i, x_j) + \frac{\delta_{ij}}{2C} \right)$$

The dual feasibility is $\alpha_i \geq 0$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$
The vector $\alpha(yU)$ must be found that satisfies since the dual is maximized.

$$D(\alpha(yU), yU) \leq \max D(\alpha, yU) = J)yU)$$

The goal is thus to find a choice for $\alpha(yU)$ such that a lower bound on Q can be computed efficiently. The choice corresponding to the lower bound presented above is the following. Train an SVM on the labeled points, obtain the vector α and complete it with zeros for the unlabeled points. Then Q(yU) is the same for all the possible labeling of the unlabeled points and the lower bound is the SVM objective function on the labeled points.

Then the branching process is performed. Let s(L) be the SVM objective function trained on the labeled set.

$$s(L) = \min \frac{1}{2} w^2 + C \sum_{(x_i, y_i) \in L} \max(0,1 - y_i(w.x_i + b))^2$$

The lower bound is s(L) is employed for the branching strategy consists in selecting the following point in U,

$$\arg \max_{x \in U, y \in \pm 1} s(L \cup \{x, y\})$$

The algorithm is implemented recursively. At the beginning, the upper bound can either be set to +1 or to a solution found by another algorithm. The bound and branching is given as follows:

Algorithm 2: Branch and bound for DAS3VM

Function: (Y *, v) DAS3VM(Y, ub)
Input: Y: a partly labeled vector (0 for unlabeled)
ub: an upper bound on the optimal objective value.
Output: Y *: optimal fully labeled vector
v: corresponding objective function.
If $\sum \max(0, Y_i) > ur$ then
return
end if
$v \leftarrow SVM(Y)$ // Compute the SVM objective function on the labeled points
if v > ub then
return // lower bound is higher than the upper bound
end if
if Y is fully labeled then
Y*← Y
Return
end if
Find index i and label y // Find next unlabeled point to label
$Y_i \leftarrow -y$ // Start first by the most likely label
(Y*, v) ←DAS3VM(Y, ub) //Find (recursively) the best solution
$Y_i \leftarrow -Y_i$ // Switch the label
(Y *, v) ←DAS3VM (Y, min(ub, v)) // Explore other branch with updated upper-bound

if $v_2 < v$ then
$Y^* \leftarrow Y^*_2$ and $v \leftarrow v_2$ //Keep the best solution
End if

Thus the optimal solutions for the DAS3VM classification are obtained and hence the user accounts are classified accurately.

Two-player security game approach

As stated [10], once the user accounts are classified, the game approach is employed for enhancing the defense mechanism. The attacker's behavior, as found earlier, when the attacker starts their first VM (there is no incentive for attackers to start more than one VM at first, as none of them will co-locate with the targets), they are labeled as medium risk. In order to be reclassified as low risk, the attacker has to keep the first VM running before starting more VMs. This is called the initial cost. After being labeled as low risk, the attacker can create as many VMs as they want. However, they have to carefully control the pace, so that they will not be reclassified as medium or even high risk. When it becomes more expensive to keep the current account being considered as low risk than to create a new account (i.e., pay the initial cost again), the attacker will discard the current account. Table 1 shows the comparison of the VM allocation policies with defense mechanisms in terms of attacker's overall cost.

Under most of the existing VM allocation policies, it is relatively easy for malicious users to co-locate with their targets, and the overall cost is also quite low. If the PSSF policy is applied, it will be much more difficult to achieve co-residence. However, because maximizing the attacker's cost is not one of the objectives when the design of the policy, there is still room for improvement regarding this aspect. Under the defense mechanism [10], all accounts are classified into different categories, and attackers are forced to act as normal users. Hence, their overall cost is increased significantly. In addition, the integration of the PSSF policy also guarantees the low probability of co-location. The two player game based defense mechanism proposed in this paper, further tries to increases the overall cost with optimal solutions for the larger size of the data center. Therefore, this defense mechanism is an effective and practical countermeasure against the co-resident DOS attack.

IV. PERFORMANCE EVALUATION

The performance of the proposed two-player game approach based defense mechanism (referred as Two-player approach in graphs) is evaluated in CloudSim. The results are compared with the co-location resistant (CLR) algorithm proposed in [24], PSSF based game theoretical approach (referred as PSSF in graphs) [10] in terms of account classification accuracy, precision, recall and attackers overall cost.

Classification accuracy

Accuracy is the percentage corresponding to the correctly done classification of user accounts as legal and malicious users.

$$\text{Accuracy} = \frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{FN} + \text{FP} + \text{TN})} * 100$$

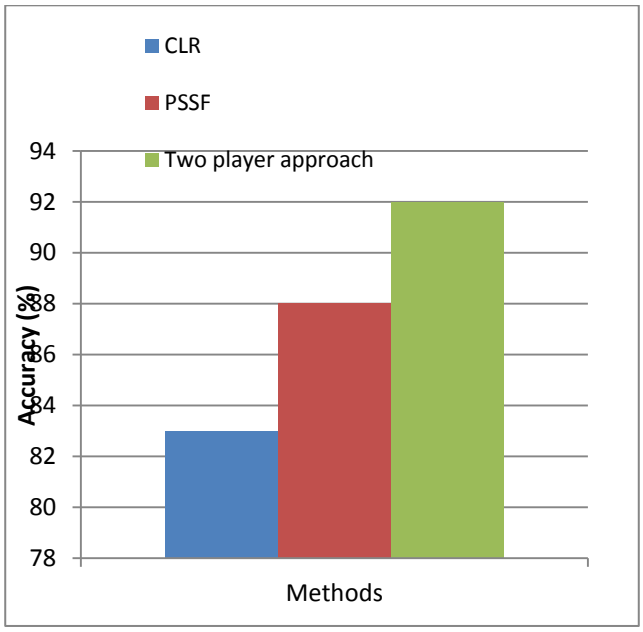

Figure 2. Accuracy Comparison

Figure 2 shows the comparison of the defense mechanisms in terms of accuracy. From the graph, it can be found that the proposed Two-player approach provides highly accurate classification. This is because of the fact that the optimal solutions for DAS3VM are obtained by bound and branch method increases the accuracy of classification.

Precision

Precision is the correctness of the classification of the user accounts.

$$\text{Precision} = \frac{\text{TN}}{(\text{TP} + \text{FP})} * 100$$

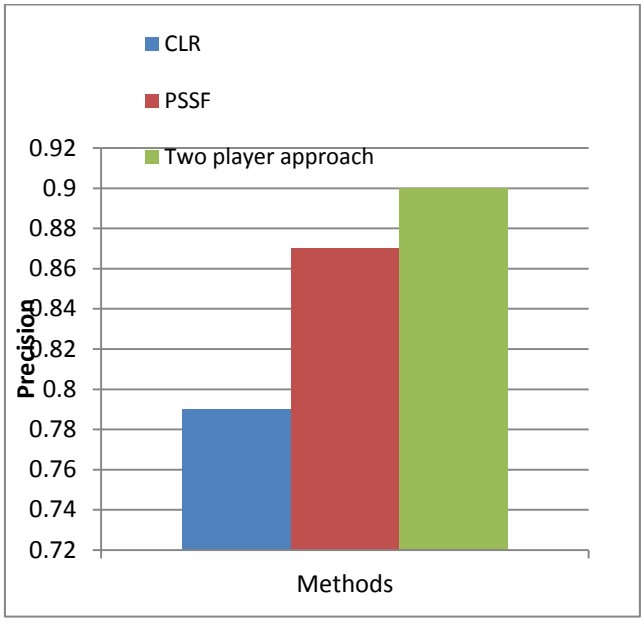

Figure 3. Precision Comparison

Figure 3 shows the comparison of the defense mechanisms in terms of precision. From the graph, it can be found that the proposed Two-player approach provides précised classification thanks to the optimal solutions for the large size data center. It is evident that the proposed defense mechanism can avoid the possibility of attacks with greater probability.

Recall

Recall is the completeness of the classification done in the cloud user accounts.

$$\text{Recall} = \frac{\text{TN}}{(\text{TP} + \text{FN})} * 100$$

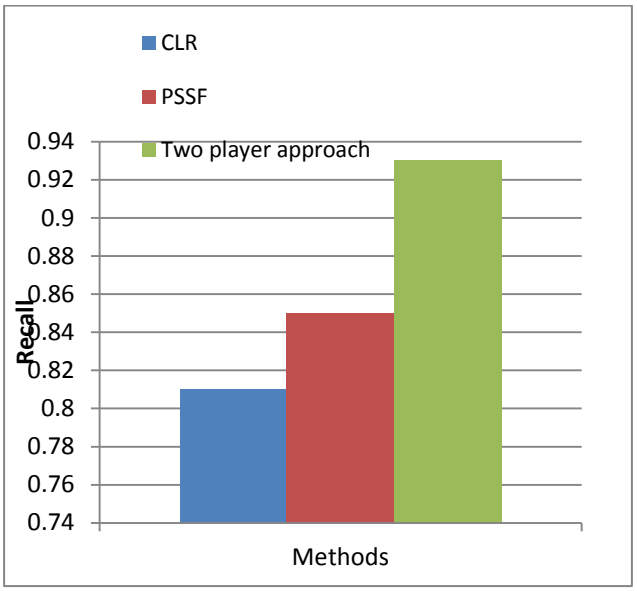

Figure.4. Recall Comparison

Figure 4 shows the comparison of the defense mechanisms in terms of recall. From the graph, it can be found that the proposed Two-player approach provides classification with high recall. The proposed approach increases the defense against co-resident DOS attacks with higher accuracy.

Attacker's overall cost

This parameter helps in evaluating the cost incurred for initiating an attack i.e. creating a new account for initiating a new VM. The cost is represented in US dollars ($) for common cost evaluation.

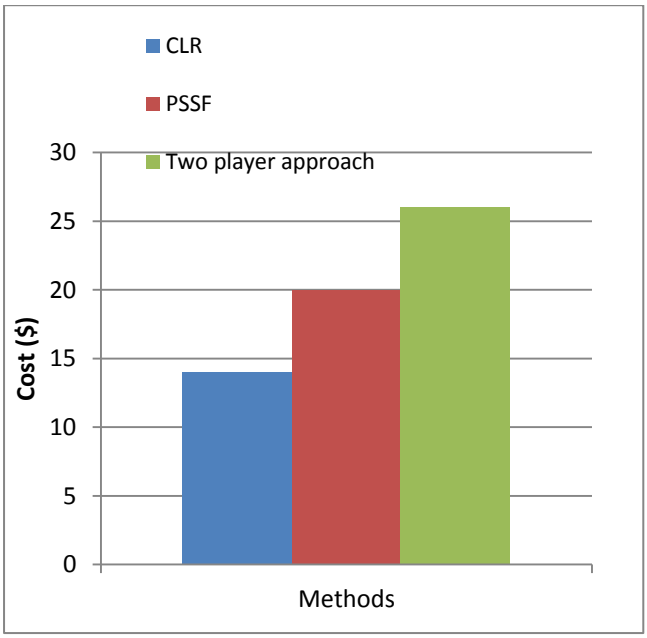

Figure 5. Attacker's overall cost comparison

Figure 4 shows the comparison of the attacker's overall cost for initiating a new VM for beginning a co-

resident DOS attack in the presence of evaluated defense mechanisms. From the graph, the cost incurred by the attacker to initiate a co-resident DOS attack is higher in proposed two-player approach than the other methods. It is evident that the proposed defense mechanism makes it difficult for an attacker by making an attack process highly expensive. Thus it forces the attacker to reduce risks and behave as a normal user.

V. CONCLUSION

This paper presents an efficient two-player game based defense mechanism to minimize the attackers from launching co-resident DOS attacks. Though many game based methods have been developed previously, most of them focus on eliminating the side channels to avoid attacks. But they do not defend effectively against the clever attackers. Hence, in the proposed method, two-player security game concept has been employed in which the user accounts' clustering done by EDBSCAN while optimized classification is achieved DAS3VM with bound and branch method. This approach considers the larger size of data centers while allocating the VMs using PSSF policy. This proposed approach effectively minimizes the probability of malicious users from initializing new accounts for co-resident DOS attack launch and the users are forced to use only one account at a time as the cost is maximized.

This is proved from the performance evaluation results in terms of accuracy, precision, recall and attacker's overall cost. Thus the proposed defense mechanism increases the security in cloud computing environment.

VI. REFERENCES

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., et al.: Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, University of California, Berkeley (2009)

[2] Amazon. Amazon Elastic Compute Cloud (EC2). http://aws. amazon.com/ec2/

[3] Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, You, Get off of my cloud: exploring information leakage in third-party compute clouds. In: CCS'09: Proceedings of 16th ACM Conference on Computer and Communications Security, Chicago (2009)

[4] Xu, Y., Bailey, M., Jahanian, F., Joshi, K., Hiltunen, M., Schlichting R.: An exploration of L2 cache covert channels in virtualized environments. In: Proceedings of 3rd ACM Workshop on Cloud Computing, Security (CCSW'11) (2011)

[5] Zhang, Y., Juels, A., Oprea, A., Reiter, M.K.: HomeAlone: Co-Residency Detection in the Cloud via Side-Channel Analysis. In: Proceedings of 2011 IEEE Symposium on Security and Privacy, Berkeley (2011)

[6] Keller, E., Szefer, J., Rexford, J., Lee, R.B.: Eliminating the hypervisor attack surface for a more secure cloud. In: Proceedings of ACM Conference on Computer and Communications, Security (CCS'11) (2011)

[7] Raj, H., Nathuji, R., Singh, A., England, P.: Resource management for isolation enhanced cloud services. In: Proceedings of 2009 ACM Workshop on Cloud Computing Security, CCSW '09, Chicago (2009)

[8] Bier, V. M., & Azaiez, M. N. (Eds.). (2008). *Game theoretic risk analysis of security threats* (Vol. 128). Springer Science & Business Media.

[9] Han, Y., Chan, J., Alpcan, T., & Leckie, C. (2014, June). Virtual machine allocation policies against co-resident attacks in cloud computing. In *2014 IEEE International Conference on Communications (ICC)* (pp. 786-792). IEEE.

[10] Han, Y., Alpcan, T., Chan, J., Leckie, C., & Rubinstein, B. I. (2016). A game theoretical approach to defend against co-resident attacks in cloud computing: Preventing co-residence using semi-supervised learning. *IEEE Transactions on Information Forensics and Security*, *11*(3), 556-570.

[11] Yinqian Zhang, Ari Juels, Alina Oprea (2011) "Home Alone: Co-Residency Detection in the Cloud via Side-Channel Analysis" 2011 IEEE Symposium on Security and Privacy.

[12] Adam Bates, Benjamin Mood, Joe Pletcher, Hannah Pruse, Masoud Valafar (2010) "Detecting Co-Residency with Active Traffic Analysis Techniques".

[13] Han Y, Tansu Alpcan, Jeffrey Chan, Christopher Leckie, (2011) "Security Games for Virtual Machine Allocation in Cloud Computing".

[14] Yu, S.: Distributed Denial of Service Attack and Defense. Springer, 2014.

[15] Lenon, M.: Cloudare infrastructure hit with 400gbs ntp-based ddos attack, 2014. http://www.securityweek.com/cloudflare-infrastructure-hit-400gbs-ntp-based-ddos-attack

[16] Kumar, N., Sharma, S.: Study of intrusion detection system for ddos attacks in cloud computing. In: Wireless and Optical Communications Networks (WOCN), 2013 Tenth International Conference on, pp. 1{5. IEEE, 2013.

[17] Ismail, M.N., Aborujilah, A., Musa, S., Shahzad, A.: Detecting ooding based dos attack in cloud computing environment using covariance matrix approach. In: Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, p. 36. ACM, 2013.

[18] Liu, H.: A new form of dos attack in a cloud and its avoidance mechanism. In: Proceedings of the 2010 ACM workshop on Cloud computing security workshop, pp. 65{76. ACM, 2010.

[19] Bedi, H.S., Shiva, S.: Securing cloud infrastructure against co-resident dos attacks using game theoretic defense mechanisms. In: Proceedings of the International Conference on Advances in Computing, Communications and Informatics, pp. 463{469. ACM, 2012.

[20] Zunnurhain, K.: Fapa: a model to prevent ooding attacks in clouds. In: Proceedings of the 50th Annual Southeast Regional Conference, pp. 395{396. ACM, 2012.

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining (KDD)*, 1996, pp. 226–231.

[22] Phung, D., Adams, B., Tran, K., Venkatesh, S. and Kumar, M. (2009) High Accuracy Context Recovery using Clustering Mechanisms, In proceedings of the seventh international conference on Pervasive Computing and Communications, PerCom Galveston, USA, Pp. 122-130

[23] Chapelle, O., Sindhwani, V., & Keerthi, S. S. (2006). Branch and bound for semi-supervised support vector machines. In *Advances in neural information processing systems* (pp. 217-224).

[24] Y. Azar, S. Kamara, I. Menache, M. Raykova, and B. Shepard, "Co-location-resistant clouds," in *Proc. 6th ACM Workshop Cloud Comput. Secur.*, 2014, pp. 9–20.