Covert Channel Detection framework for cloud using distributed machine learning

Mansaf Alam, Shuchi Sethi
Department of Computer science, Jamia Millia Islamia (a central university)
Covert Channel Detection framework for cloud using distributed machine learning

Abstract— Recent research shows that colluded malware in different VMs sharing physical host may use access latency of a resource as a covert channel to leak critical information. Covert channels employ time characteristics to transmit confidential information to attackers. In this manuscript we have made two important contributions and to the best of our knowledge they are novel. One is to propose a framework for detecting covert channel attack based anomalies in cloud using machine learning. This framework is based on application criticality where the degree of error tolerance will be based on the sensitivity of application. This is a useful feature for cloud based applications as SLAs are trade off between service and cost. There are two modules of the framework, a cluster monitor and another module becomes part of hypervisor for monitoring purposes. Second contribution is we propose an approach to distribute the machine learning technique in such a manner to be able to handle imbalance learning as well as enabling SVM to handle large datasets.

Index Terms—SVM, Covert channel, detection framework, anomaly detection, virtual machine, cloud computing, security

I. INTRODUCTION

Security is a major barrier to mass adoption of cloud. The need for cloud to be secure cannot be over emphasized. Co-residency with other tenants poses a threat as virtualization is a key enabling technology in cloud computing. Recent research has shown that covert channels in shared hardware enable attackers to exfiltrate sensitive data. Covert Channel attacks tend to break the secure isolation provided in a virtual environment. This study has gained momentum after cloud computing is becoming popular as attacks of these types particularly pose threat to cloud environments which base their security on isolation in virtual setup.

Small data leakages can be detected by existing firewalls, intrusion detection mechanisms, traffic logging etc. This makes covert channels ideal choice to leak confidential data in cloud for attackers with compelling reasons. Impossible elimination and difficult detection add fuel to the fire in this situation. There are various categories of covert channels for leaking information like paging rate, temperature based covert channels[11]. But most channels are noisy with high error rates. Certain channels like billing of used resources may conceal secret information, but such channels are overt and its traffic which hides secret information. With extensive data analysis it won't be very difficult to detect hidden data patterns.

So far covert channel attacks have not been exploited in real life but they are treated as a major potential risk. Besides with demonstration of risk in practical scenarios, it can safely be assumed more hazardous as it would require high expertise by attacker to use such channels and with time attacks only get better. This gives inspiring conviction that such threats are taken seriously. A few researchers have demonstrated such attacks on Amazon EC2 [3, 1].

Cache based covert channels were introduced by percival and its feasibility in cloud was proved first by Ritenpart et al.[3] and [7]. Researchers proposed techniques like RPcache, Homealone[5] for such channels. But research has later shown that cache based covert channel attacks are noisy due addressing, scheduling and physical limitations. Zhenyu Wu et al. demonstrated a pure time based channel based on memory bus contention.

Two main techniques have been proposed to counter time channel attacks. One is fuzzy time where the accuracy of the clock is disrupted so that common time reference of sender(a trojan) and receiver(a spy) is also disrupted. Another way is to use lattice scheduling for processor scheduling to avoid sender and receiver processes be able to run at the same time. But using self clocking techniques popular in network transmissions such obstacles can be overcome by the attacker.

This paper incorporates attacks like load based time channels that are caused due to virtualization in cloud; in cloud VMs are co-located and resources are shared [10]. Due to parallelism and dynamic resource allocation in cloud, covert channels are hard to control and only provider can report such attacks; but its not in their interest to do so [8]. Besides these attacks leave no trail so these are likely not to be noticed. Continuous monitoring of flow of information between VMs might help but it may result in legal issues. The main reason for this threat attributes to the fact that the resources are not properly isolated in cloud but resource partitioning techniques limit the efficient multiplexing of resources and affecting the performance of cloud.

The major challenge in detecting a covert channel based on access latency is to be able to train the detector to differentiate between access latency due to attacker and due to noise(core migration, scheduling etc) in virtualized environment so that classification is accurate.

Authors believe there is not much effort being put in detection and mitigation of covert channel despite the fact that research has been extended in this field of study to show the presence of new attacks in cloud, a few of which are incorporated in this manuscript, which aims to fortify this research. Still most companies rely on traditional intrusion detection mechanism for covert channel detection. It must be mentioned here that majority of threats in cloud emerge from multitenant architecture and not necessarily a product of external network. Though in related work some relevant efforts in the area are mentioned but in cloud performance of detection algorithm is an issue and practicality of some efforts are questionable. The contribution of this paper is injecting a context awareness in SVM as a flexi-parameter for covert channel detection framework for classification of a VM as secure or insecure.
The current generation of cloud computing infrastructures do not provide any security against untrusted cloud operators making them unsuitable for storing sensitive information such as medical records, financial records or high impact business data. To solve this issue various research projects are going on like proofs of storage whose target is to be able to identify if a tenant's data has been tempered. Besides research is going on to provide secure cloud storage systems that will provide all 3 aspects of data security as confidentiality, verifiability and integrity. Confidentiality has always been a focus of researchers and so has been encryption. Researchers have developed a virtual vault using homomorphic encryption where data will be processed in encrypted form and no leakage would happen.

But this does not completely remove the possibility of data leakage. If both sender and receiver cooperate in a manner as to transfer the data which is not allowed by access control policies, leakage may happen. Here the sender is an innocuous VM that has unknowingly downloaded a malicious program like trojan horse and receiver is a spy program. This is done using a means where this communication is undetected, bypasses access control policies and exploits shared resources to communicate and hence called covert channels. There has been a lot of research in this area which is posing a threat to cloud computing[20-23,26,27].

The major challenge faced for detection is that in cloud dynamic changes of the system make error rates go high and it will require new threshold. Time trait is a lucrative distinctive measure to study pattern-change behavior. But dynamism of cloud may yield higher similarity than expected.

II. RELATED WORK

A few previously known detection techniques available and applicable to such channels in "non-cloud" environment are shape and regularity tests and measuring entropy of the data to determine the data is legitimate or not [10,12,13]. Introduce Minimal Requisite Fidelity that can result in disruption of communication in structure carriers and decrease the capacity/speed of the channel. Then "the wardens" can detect such communication. In some approaches data is tested based on certain parameters to see if it carries a secret message. In cloud scenario limited historic data is available for comparison due to environment being dynamic. For those interested in further reading can refer to[14-16,24]. This research paper is focused on covert channels where the channel or medium of communication itself is hidden rather than just the hidden traffic. Such channels are hard to detect conceived from the fact that there is no visible flow of data itself so detection techniques applied to overt channels with hidden messages are not applicable. This paper distinguishes the covert channels from covert communication.

Researchers have proposed variety of techniques in covert channel detection. Most are based on building a model based on secure historic traffic study and comparing the secure model with current and looking for deviations that will enable detection based on verification of suspicion. This is simple and idealist approach though previous approaches result in high false positives.

In [17] a solution to such attack is proposed by capturing all atomic instructions and replacing them as Trap to hypervisor. When the hypervisor intercepts this trap, it pauses the guest's vCPUs and executes the instruction without asserting the memory bus lock. Since the guest's vCPUs are paused, no other memory operations can be performed by this guest. But tackling situation this way has a limitation of slowing the system down. Thus it is not generalizable.

In [18] researchers have implemented an observer that uses a secure VM to compare traffic between a vulnerable VM and secure VM. This method can be used in real time. But crux here is maintaining a secure copy of every VM which itself becomes its limitation.

In [4] authors have proposed a flexible framework for time based covert channel detection by classifying their distributions as markov when attack is being launched and bayesian otherwise. The Bayesian detector is trained and then target value is calculated. This method is based on prerequisite of strong independence, which makes it suitable for covert channel but not versatile enough to be adapted for other attacks.

Cabuk et al [16] have designed robust time based covert channel and proposed regularity(correlation in data) tests for detection of covert channels. With study of shape(mean, variance, distribution) applying tests like Kolmogorov-Smirnov its possible to detect most covert channels. In [13] researchers proposed conditional entropy based approach for covert channel detection. But its applicability in cloud is still a question.

In [5] authors propose a scheme of Homealone that can detect any unusual cache activity when friendly VMs are silenced for a certain period. Though the original approach can be modified to enable detection, but this approach has a high overhead as suggested by researchers in [1].

III. THREAT MODEL

A. Scope of threats covered in paper

Threats that break the isolation layer of VMs by leaking information between VMs which are otherwise prohibited from communicating directly and want to share information illegitimately are part of our model. The two popular categories of inter VM covert channel are: if the VMs are on same platform and the other one is when the VMs are on separate platform. The latter category consists of network based covert channels using TCP/IP header information like ISNs which hide confidential information and are randomly generated by the sender. Though such attacks are hard to detect but novel approaches using chaos theory have been successfully applied to detect such covert channels and such attacks are not new to cloud environment. But the inter VM attacks on same platform are specific to cloud. The threat model includes the attackers which have access to same VM as the victim. The malicious intentions of the attackers can be bypassing access control and leak confidential data or increase use of shared resource like cpu or memory to deny essential services to co resident VMs.
In this figure, one VM is bypassing any security control and communicating using legitimate means transparent to existing detection methods.

IV. MACHINE LEARNING

We mention certain notations that are required for discussing our framework. SVM stands for support vector machines which is popular high accuracy classification technique. Support Vector machines use a hyper-linear separating plane to create a classifier. It is a non-probabilistic binary classifier. For problems that cannot be linearly separated, SVM offers a probability to find a solution by making a non-linear transformation of the input space into a high dimensional feature space. Those separating planes are optimal, which means that a maximal margin classifier with respect to the training data set can be obtained.

Soft margins are introduced by employing a vector of slack variables €i, which measure the degree of constraints violation. Soft margin SVM can be defined as:

$$\text{min } \frac{1}{2} ||W||^2 + C \sum \xi_i \text{ s.t. } y_i (w^T x_i + b) \geq 1 - \xi_i; \xi_i \geq 0$$

This can be rewritten as:

$$\text{min } \frac{1}{2} ||W||^2 + C \sum \max(0,1-y_i(w^T x_i + b))$$

The data point x' falls in three categories. On the margin, it acts as a support vector and within the margin, where it adds to cost of the function by violating the constraint. Now C in the rewritten equation to determine the cost of allowing errors. If C is close to 0 cost of margins is low while if C is close to infinity, cost of violation goes high.

SVM by default is not suitable for large scale distributed systems where processing is required in real time as it converges only for a small dataset. This makes it nonscalable in both computation time as well as memory use.

V. FRAMEWORK FOR COVERT CHANNEL DETECTION

Its a dual sided coin, pattern recognition and anomaly detection and these are the two epistemic poles between which all the work wanders. Definition of anomaly germinates from the ground of pattern equilibrium. As this manuscripts discusses its approach to bifurcate the anomaly from mass metadata in data centers, it is presumed the best place to look for is when resource request originates and flows to hypervisor. This limits the set of variables to look for. Our framework works on two main parameters one is resource access latency and other one is resource request.

The framework has two modules. Module one captures access latency parameter of the VMs that are part of its cluster. Then its two layers, one being the offline layer preprocesses the data based on access latency of a resource and some data which is not obtained ,typical feature of big data, will be taken care with preprocessing using hive where data is stored using Hadoop distributed file system allowing missing data items. This behaviour attributes to the fact that some of the applications in VMs may not be requiring certain resources during the time of monitoring. In this section our focus would be on actual processing, so we
simply assume any big data tool may be used along with layer 2 of our framework. The second (processing) layer is active processing layer. In this layer data is fed into the LCBCCD (Linear Cloud Based Covert Channel Detection Algorithm) which in turn returns if a process signifies covert channel attack.

Data can be represented as \( X = \{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\} \) where \( x(i) \) is a real number; \( y(i) \) belongs to \{-1,1\}

If the two classes are \( c(i)=+c \) and \( -c \) which are linearly separable where given the training data set with input \( x(i) \) where \( x(i)>0 \) is \(+c\) if its a covert channel attack and \( x(i)<0 \) is \(-c\) if not.

**PROPOSED ALGORITHM I**

```
LCBCCD (DATAFORSVM[], SVMSTRUCT)
```

Begin
BeginFor \( i=1 \) to \( N \)
\( \backslash \) Import data for \( N \) VMs and average the \( \backslash \) access latency
Data\[i\] = dataforsvm\[i\]
EndFor
BeginIf
If application is “CNCA”
\( C<1 \)
Else if application is “NCNCA”
\( C>1 \)
SVMSTRUCT = SVMTRAIN(Data, label, C)
\( \backslash \) we train the data to obtain hypothesis stored \( \backslash \) in the form of SVMSTRUCT.
EndIf
SVMOUTPUT = SVMCLASSIFY(SVMSTRUCT, Data[\( i \)])

To distinguish between the security requirements of applications, it is proposed that a VM be associated with the most critical application it runs and be put under one of 4 categories (C4) as and when the VM is instantiated. This can be achieved by user input or as part of sanity check prior to its active functioning.

**Step 1:**
- **Category One:** Commercial Non-Critical applications (CNCA)
- **Category Two:** Commercial Critical applications (CCA)
- **Category Three:** Non-Commercial Critical Applications (NCCA)
- **Category Four:** Non-Commercial Non-Critical Applications (NCNCA)

**Step 2:**
- If the application is either category One or Four then we apply Linear Cloud Based Covert Channel Detection Algorithm (LCBCCD). Being linear classification with single parameter this is simpler and has faster performance.

**Step 3:**
- If the application is either category Two or Three then we apply Kernel Cloud Based Covert Channel Detection Algorithm (KCBCCD).

**PROPOSED ALGORITHM II**

```
KCBCCD (DATAFORSVM[], SVMSTRUCT)
```
Begin
BeginFor i=1 to N
\ capture resource request
Data[i] = datasvm[i]+ContextData[i]
EndFor
BeginIf
If application is "CNCA"
C<1
Else if application is "NCNCA"
C>1
\ Context is added for training to enable high
\ accuracy in covert channel detection
svmstruct=svmtrain(Data,label,C,kernel,'rbf')
\ we train the data to obtain hypothesis stored in
\ the form of svmstruct.
EndIf
svmoutput=svmclassify(svmstruct,Data[])
End

KCBCCD algorithm differs from LCBCCD in following ways:
  a) It uses non linear transform to fit the data. Radial
      basis function is powerful classifier.
  b) To enable generalization and avoid overfitting
      without valid reason slack variable is used.
  c) It uses context variables to allow context based
detection in cloud.

The major issue with the above algorithms is that in present form they will only work on small dataset and also suffer from imbalanced learning.
In the algorithm below we assume that 5 VMs are available as in our cloud testbed, we had only 5 active VMs that were participating to obtain the result. But in real world this can be scaled up.
To handle the big data scalability issue we partition the dataset such that non anomalous set and anomalies in the training set are equal in number on each VM in the cluster considering all anomalies on one VM each. Thus training is done in such a manner so that only non attack data differs on each VM. Also size of dataset on each VM is determined by anomalies in the given set. This provides for balanced learning and DCAD(Distributed Cloud based Anomaly Detection) algorithm also permits big data processing at the same time. The validity of DCAD algorithm does not need to be discussed as we have not altered the mercer’s condition.

VI. EXPERIMENTS AND DISCUSSION
There are minimum requirements to set up a covert channel. In [7] authors state a theorem that If the sender is able to invoke change(s) in the visible space or execution trace of the receiver, a covert channel may exist. Also to deny services to a VM there is no need to have a visible space. By consuming enough cpu etc resources , timely services to other VM can be disturbed causing access latency.

A. CPU load based attack
We seek to find answer to a couple of questions while replicating this attack. Can this attack lead to access latency? In a simple CPU-load based attack [4], attacker VM launches an attack by making sudden bursts of requests for CPU resource. For a covert channel attack sender sends more than 100 user requests in burst to transmit a bit 1 and no request is made for a bit 0. With time synchronization
after a time gap of every 10 secs request for the resource is made. One of the colluding VM with sensitive information intending to leak it, has Apache JMeter tool, usually used to load test a website and spy VM has a static website running on it. We can use JMeter to specify the user load and control cpu usage of the system which gives spy 1 bit if load is high and 0 bit if load is low. Also the attack can be used to launch denial of service against a VM with web server by sending request bursts with 500 or 1000 users(threads) as it takes CPU usage upto 90%. A simplified version will exploit CPU as shared resource and on a windows(Hyper-V) based system. Monitoring tool can be used by the receiver to interpret the bits. In covert channel attacks care has to be taken not to make time difference very significant using load as covertness is inversely proportional to accuracy and bandwidth of the channel making it easily detectable, so based on the system we can decide ideal number of threads as shown in figure 1 and 2 below. Thus as a convincing reply to questions above, such attacks are detectable and access latency is one of the parameter that can reliably be used for detection.

**C. Bus contention based covert channel**

Whenever memory bus is put in contended state as its being shared by VMs and there is a delay in accessing the bus, it will be noticed by the receiver VM. This attack as explained in [1] is carried out by a sender sending the packet by causing latency in accessing the memory bus by the receiver. This is interpreted as high and low signal. Wu et al have demonstrated that bus contention based attack can achieve a high bandwidth of over 100-bits per second which can steal a thousand byte private key file in less than 3 minutes. This attack is hard to detect because of its negligible impact on performance of cache causing transparency for any cache based detection techniques. Thus our detection framework does not only rely on access latency of bus and cache but also time interval as a parameter for detection.

Minimum latency occurs when transmitting a 0. Latency is calculated as $\text{Mean(Access Time)} - \text{Base time}$. In bus contention based channel, contention is created on front side bus using atomic instructions like xchg 8086 instruction for exchanging memory with register. There are many instructions in assembly that can be used to deliberately create contention. A predecided threshold time is agreed between two programs. Then if at receiver end $\text{mean(access time)} > \text{threshold}$ bit is interpreted as 1 else 0. If too many consecutive 1s or 0s, receiver can assume that sender process has been descheduled by the hypervisor.

If the latency is not random then this can be an attack and we try to find a function using SVM to map the latency pattern.

The datasets used are google cluster trace [25] may 2011 from where we obtained normal training data and then inject anomalous data. In this preprocessing was done to scale the cluster data down so that learning is uniform.
We compare our LCBCCD and KCBCMCD detection framework results to an approach proposed in [4]. We applied k-cross fold validation and average of all specificity and sensitivity are taken to simplify result comparison.

To test our DCAD algorithm we used dataset svmguide1 with 7089*4 points year 2010 from mldataset from machine learning repository. This dataset is taken for comparison with existing literature[2].

A. Accuracy

The table given below shows the results and comparison.

B. Robustness

In absence of real life cloud data which is a noisy environment ,authors check the robustness of method proposed and how fast it would deteriorate in performance if we added some noise to the data. In fact in practical scenarios such situations will be common.5% gaussian noise is added to vector latency as shown in the graph below and then performance is evaluated for RBF with context. The error rate was 0.8% for the method which means around 0.7 increase in error due to 5 percent increase in noise. This shows that our approach is practically usable in real life scenarios. Authors believe that use of context variables has increased reliability.

C. Alternative Approach Comparison

To compare our approach to NN in our tests we applied Neural network for training and testing on data using 10-fold cross validation. The results are shown in another graph. We tested NN performance and error rate was higher at around 8% in first training. It improved as we retrained the network but as our model is required for real time scenarios, time required for training should be minimum. Initially we trained NN for three features on 6 neurons(taking ideal case of 2N+1). Thus comparative analysis suggests our approach is more practical. The authors considered this alternative approach due to the fact that SVM cannot really handle large volume of data but less satisfactory results gave authors compelling reason for distributing SVM over cloud for processing.

VII. SUMMARY AND CONCLUSION

This approach of classification using learning has important applications in the area of medicine by detecting abnormalities in human beings; unexpected engineering issues like jet engine vibrations; Most importantly in Intrusion detection. In this paper we have used performance and high activity as parameters for detection of the bus contention time channel but as we try to generalize this model, we might come across time channels that are clever enough to leave no footprints, keep activity levels lower than

| Algorithm          | Specificity | Sensitivity | Error rate          |
|--------------------|-------------|-------------|---------------------|
| LCBCCD             | .91         | .95         | The error rate      |
| KCBCMCD            | .9771       | .9935       | specified here are  |
| C2 detector        | 1           | .72         | of those obtained   |
| DCAD               | Accuracy-.97|             | using the framework |
| Distributed SVM    | .91         |             | (1)CPU load : .04   |
|                    |             |             | (2)Cache: .1        |
|                    |             |             | (3)Bus contention:  |
|                    |             |             | .06                 |

REFERENCES

[1] Wu, Zhenyu, Zhang Xu, and Haining Wang. "Whispers in the Hyper-space: High-speed Covert Channel Attacks in the Cloud." USENIX Security symposium. 2012.
[2] Zhu, Kaihua, et al. "Parallelizing support vector machines on distributed computers." Advances in Neural Information Processing Systems. 2008.
[3] Ristenpart, Thomas, et al. "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds." Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009.
[4] Wu, Jingzheng, et al. "C2detector: a covert channel detection framework in cloud computing." Security and Communication Networks 7.3 (2014): 544-557.
[5] Zhang, Yinqian, et al. "Homealone: Co-residency detection in the cloud via side-channel analysis." Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011.
[6] Gray III, James W. "On introducing noise into the bus-contention channel." Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on. IEEE, 1993.
[7] Wang, Zhenghong, and Ruby B. Lee. "New constructive approach to covert channel modeling and channel capacity estimation." Information Security. Springer Berlin Heidelberg, 2005. 498-505.
[8] Aviram, Amitai, et al. "Determining timing channels in compute clouds." Proceedings of the 2010 ACM workshop on Cloud computing security workshop. ACM, 2010.
[9] Potlapally, Nachiketh R., et al. "Aiding side-channel attacks on cryptographic software with satisfiability-based analysis." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 15.4 (2007): 465-470.

[10] Gianvecchio, Steven, et al. "Model-based covert timing channels: Automated modeling and evasion." Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2008.

[11] Okhravi, Hamed, Stanley Bak, and Samuel T. King. "Design, implementation and evaluation of covert channel attacks." Technologies for Homeland Security (HST), 2010 IEEE International Conference on. IEEE, 2010.

[12] Fisk, Gina, et al. "Eliminating steganography in Internet traffic with active wardens." Information Hiding. Springer Berlin Heidelberg, 2003.

[13] Gianvecchio, Steven, and Haining Wang. "Detecting covert timing channels: an entropy-based approach." Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007.

[14] Cybenko, George, and Vincent Berk. "Process detection in homeland security and defense applications." Defense and Security Symposium. International Society for Optics and Photonics, 2006.

[15] Berk, Vincent, et al. "Detection of covert channel encoding in network packet delays." Rapport technique TR536, de l'Université de Dartmouth (2005).

[16] Cabuk, Serdar, Carla E. Brodley, and Clay Shields. "IP covert timing channels: design and detection." Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004.

[17] Sultafomaggio, Brendan, Dongyan Xu, and Xiangyu Zhang. "Busmonitor: A hypervisor-based solution for memory bus covert channels." Proceedings of EuroSec (2013).

[18] Liu, Anyi, Jim Chen, and Li Yang. "Real-time detection of covert channels in highly virtualized environments." Critical Infrastructure Protection V. Springer Berlin Heidelberg, 2011. 151-164.

[19] Zhao, Hong, and Yun-Qing Shi. "Detecting covert channels in computer networks based on chaos theory." Information Forensics and Security, IEEE Transactions on 8.2 (2013): 273-282.

[20] Wu, JingZheng, et al. "Identification and evaluation of sharing memory covert timing channel in Xen virtual machines." Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011.

[21] Zhang, Yinqian, et al. "Cross-VM side channels and their use to extract private keys." Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012.

[22] Hélouët, Loïc, and Aline Roumy. "Covert channel detection using information theory." arXiv preprint arXiv:1102.5586 (2011).

[23] Xu, Yunjing, et al. "An exploration of L2 cache covert channels in virtualized environments." Proceedings of the 3rd ACM workshop on Cloud computing security workshop. ACM, 2011.

[24] Borders, Kevin, and Atul Prakash. "Web tap: detecting covert web traffic." Proceedings of the 11th ACM conference on Computer and communications security. ACM, 2004.

[25] Chen, Yanpei, et al. "Analysis and lessons from a publicly available google cluster trace." EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95 (2010): 94.

[26] M. Alam 1 ; S. Sethi 1 ; S. Sethi, Covert Channel Detection Techniques in Cloud Confluence 2013: The Next Generation Information Technology Summit (4th International Conference), 2013 page 3.02

[27] Shuchi Sethi, Kashish Ara Shakil, Mansaf Alam, Seeking Black Lining In Cloud, arXiv:1501.04473 [cs.CR]