A Collaborative Approach for a Source based Detection of Botclouds

Hammi Badis
Troyes University of technology
Autonomous Network Environment Team
badis.hammi@utt.fr

Guillaume Doyen
Troyes University of technology
Autonomous Network Environment Team
guillaume.doyen@utt.fr

Rida Khatoun
Telecom ParisTech
rida.khatoun@telecom-paristech.fr

***Abstract*—Since the last years, cloud computing is playing an important role in providing high quality of IT services. However, beyond a legitimate usage, the numerous advantages it presents are now exploited by attackers, and botnets supporting DDoS attacks are among the greatest beneficiaries of this malicious use. In this paper, we present an original approach that enables a collaborative egress detection of DDoS attacks leveraged by a botcloud. We provide an early evaluation of our approach using simulations that rely on real workload traces, showing our detection system effectiveness and low overhead, as well as its support for incremental deployment in real cloud infrastructures.**

I. Introduction

For the last few years, cloud computing has rapidly gained momentum. The reason mainly lies in its potential to offer high service performance while reducing infrastructure and operational costs to its users. However, beyond their legitimate usage, these features are also exploited by malicious users, who use the cloud as a support for their attacks toward any third party connected to the Internet. The greatest beneficiaries of this cloud conversion into an attack support are botnets called Botclouds, that perpetrate Distributed Denial of Service (DDoS). Beyond the numerous exploit cases that are currently reported, the successful deployment of Botclouds is stated in the experimental study [7] that showed how five of the most famous CSP can be at the source of many different attacks over a mid-term period. Such a facility of attacks deployment is partly due to the simplistic or even lack of egress filtering mechanisms. Showing that, for instance, recommandations provided in RFC 3013 [5] against IP spoofing, are now not sufficient to efficiently deal with this phenomenon.

In this context, our goal[1] is to develop an egress detection system to protect the cloud infrastructure from being a support for DDoS attacks. Such a goal is highly challenging since it induces the detection of distributed and weak footprint malicious operations at their source in a highly heterogeneous and dynamic environment. As a consequence, we propose to design and implement a distributed approach composed of monitoring probes located at the hyperviser level and that monitor the system activity of virtual machines.

Numerous studies have tackled the problem of intrusion and DDoS attacks. In [3], the authors presented an Unsupervised Behavior Learning (UBL) system for predicting performance anomalies in virtualized cloud systems. UBL is a host-based IDS, implemented at the hypervisor level and that leverages Self Organizing Map (SOM) to predict anomalies system behaviors. However, the limit of UBL lies in monitoring Thus, it is not effective in detecting distributed abnormal behaviors and attacks such as DDoS ones. Unlike UBL, our approach relies on a signature-based approach implemented at the source hosts which do not need a learning phase. Indeed, due to cloud's workload volume and heterogeneity, it is difficult to define normal activity, and thus to rely on it for detection. However, relying on signature-based approach allows a best detection of DDoS attacks. Since DDoS attacks represent large-scale coordinated attacks, in order to detect them efficiently, one needs to combine the evidences of suspicious network or host activity from multiple distributed hosts and networks. To overcome the problem of IDS isolation, Collaborative IDS (CIDS) have been proposed. However, if CIDS solutions provide promising results they are intended to be integrated in intrusion detection systems close to, or even at, the target location which leads to unavoidable side effect damages. Source-based detection approaches represent a very promising solution to detect large scale attacks and to avoid their side-effect damages. *D-WARD* [6] represents a DDoS defense mechanism that autonomously detects and stops attacks originating from the networks they monitor, thus avoiding them from being involved in these attacks. However, the limitation of this solution is the large number of independent network administrative domains that must implement it in order to be efficient, which makes is hardly deployable.

The following of this paper is organized as follows. In section II we present the collaborative approach we propose. In section III, we provide the early validation results of our approach that we obtained by using on real workload traces obtained through *in situ* experimentations. Finally, section IV draws the first conclusion of this work and highlights some direction of future work.

II. A collaborative source-based approach

A. System architecture

The proposed hierarchical detection system relies on a mesh architecture resulting from the overlap of several trees of overlay networks. Each tenant is treated separately and

[1]This work is supported by the CASCC (*Contrôle Autonome et Sécurité dans le Cloud Computing*) research project, which is founded by the Champagne-Ardenne region.

one tree is built for each. The goal of using a tree-based hierarchical architecture resides in reducing the detection algorithm's time and execution cycles complexity. Indeed, the proposed approach relies on the idea of (1) applying PCA-based detection algorithm, on small sets of VMs after normalizing their data. A set of VM having the required number is called a *Clique* (Q) and its size n is fixed as a pre-execution parameter; and (2) aggregating the set of results in order to take a decision about the tenant's state (attacker or not). To reach such a goal, each node (IDS) of the architecture has a specific task whose description is given below.

According to its position in the tree, a node could be one of three types. The $Root$ node, which represents the node having the highest level of a given tree is responsible for final detection decisions. *Transition Nodes* (TN), represent nodes assuring data transition between *Leaf* nodes and the $Root$. $Leaf$ nodes, represent the nodes having the lowest level and which interacts directly with VMs. Due to the fully distributed approach built in a peer-to-peer fashion of our system, the type of a given node changes from one tenant to another. In other words, a node could be a $Leaf$ for a given tenant, a $Root$ for another tenant and a TN for a third one. Each node, except the $Root$ communicates only with its father. Figure 1 describes the detection system architecture and scenario of one tenant tree using $n = 5$. The trees structures are maintained through a P2P Distributed Hash Table (DHT) such as those based on the Plaxton algorithm [8]. The use of a P2P substrat in a cloud environment is motivated by (1) the need for a scalable infrastructure able to deal with the potentially hundreds of thousands VM a CSP can host; (2) the need to address the churn of VMs that can be launched, stopped and moved from one hypervisor to another in a dynamic manner and (3) the absence of any central point that could act as a single point of failure, thus ensuring the resilience of the detection system.

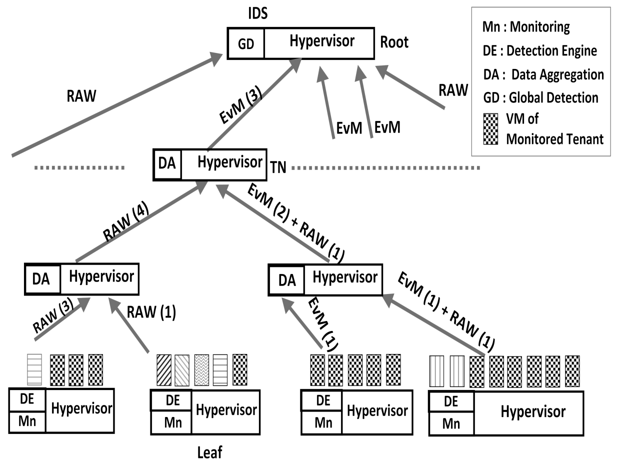


Figure 1: System architecture and detection scenario for $n = 5$

1) Detection engine : The heart of our proposed detection approach relies on the dissimilarity calculation of the factorial space describing a tenant's activity with the factorial space representing DDoS flooding attack activity. Due to space constraints, we invite the reader to refer to [1] for a detailed description. In the rest of this paper, we reference this phase by Algorithm PCA.

2) Leaf nodes: $Leaf$ nodes are situated at the bottom of the structure and represent the sole nodes having direct interaction with the VMs via a monitoring probe. Monitored data is stored in a matrix we call TRD. According to the size of TRD, which represents the number of monitored VMs the hypervisor hosts for a given tenant, the respective IDS executes one of the three following actions : (1) If the number of VMs is less than n, the $Leaf$ sends 1 message containing a table of the monitored data (raw data) of all VMs to its father. (2) If the number of VMs equals to n, the $Leaf$ applies Algorithm PCA on data of monitored $Clique$. Then, it sends 1 message containing the eigenvector matrix, to its father. (3) If the number of VMs is greater than n, the $Leaf$ applies Algorithm PCA on each one of the *(number of monitored VMs / n)* $Cliques$. Afterwards, it sends 2 messages to its father. The first message contains the list of the different eigenvector matrices (LEM) resulting from the different applications of Algorithm PCA. The second message contains the table of the remaining *(number of VMs modulo n)* raw data.

3) Transition Nodes (TN)*:* TN are intermediate nodes between a $Leaf$ and the $Root$. They are called TN because they assure data transition and aggregation operations. More precisely, a TN acts exactly like a $Leaf$ node. The sole difference resides in the data reception. Indeed, a $Leaf$ node gets only raw data through monitoring, while a TN receives raw data as well as eigenvector matrices. A TN stores the different messages it receives from all its children (*Leaf nodes* or other *TN nodes*). Thus, it owns one matrix which contains the raw data, called TRD and a list that contains the eigenvector matrices called LEM. According to the size of TRD, three scenarios are considered such as for *Leaf* nodes.

4) Root node: The $Root$ node is responsible of the detection decisions. Indeed, like a TN, the $Root$ stores the different messages it receives from all its children. Thus, it also builds TRD and LEM. Then, it applies Algorithm PCA on each one of the *(size of TRD /n)* $Cliques$ and on the set of *(size of TRD modulo n)* remaining raw data. Each result of eigenvector matrix is added to LEM.

The next step consists in computing the factorial space that represents the global tenant's activity. Since each element of LEM represents a factorial space that represents the activity of a $Clique$, the center of gravity of all these factorial spaces represents a median factorial space that describes the global activity of the tenant [4]. Once this global activity is computed, it is compared with an attack reference R which represents the factorial space that stands for the signature of an attack activity [4]. To that aim, we compute the distance between these factorial spaces calculated through the dissimilarity between eigenvector matrices. The dissimilarity measure is calculated by the Frobenius norm.

However, due to noise and limits in the system monitoring, which especialy occurs for the almost imperceptible activity of weak attacks, monitored metrics can suffer from abrupt changes. For instance, the CPU activity measurement of attacking VMs, which exhibits some breaks, suffers from it. This phenomenon strongly impacts the metrics' correlations, which in turn impacts the dissimilarity computation and causes a high rate of false negatives. To remedy this problem, and since attacks last for a significant time to be effective, we apply an Exponentially Weighted Moving Average (EWMA) as a post filter. That way, we link activity data and limit abrupt changes and thus outlier detections. The final step of the detection process consists in the comparison of the dissimilarity value with a fixed threshold, to decide whether or not the dissimilarity value stands for a DDoS attack.

III. Evaluation and discussion

A. Evaluation scenarios and workload traces

In order to evaluate the performance of the proposed collaborative approach, we rely on simulations using real workload traces provided by an intensive measurement campaign we conducted [2]. We used real traces (VM system logs) by injecting them into a simulator built on the R tool[2]. More precisely, we consider the workloads of five experimentations. For each one, a botcloud realizes an UDP flood attack at a stable rate. The latter goes gradually from a weak attack rate of 8 MB/s up to a very sharp one of 80 MB/s per source, reaching aggregated attack rates of 328 MB/s to 3.12 GB/s. For each experiment, we present the early results we obtained through facing the botcloud's activity to 24 other legitimate tenants picked randomly from a total workload of more than 1250 VMs, deployed on an architecture of 8 physical servers having one hypervisor each.

As the approach rely on $Cliques$, we are interested, in studying the impact, on the detection quality, of the n parameter, which represents the number of VMs in a $Clique$. To that aim, we realized simulations by varying n from 5 to 10 and we considered the case of weak and sharp attack rates described above. The set of n values was chosen to always rely on small $Cliques$ for detection, while covering a time complexity of one decade for Algorithm PCA

B. Evaluation results

In this section we present the different results we obtained. Each presented result represents the arithmetic average of the 50 iterations of the described experimentation. The confidence intervals we got are very narrow and we omitted them for visibility purposes.

1) Sharp attacks : Figures I.a, I.b and I.c represent the results describing experimentations where the botcloud realizes a straightforward and sharp attack for all n values. Figure I.a represents the ROC curves of the detection results. We can note that the system exhibits good detection performances since for all n, curves are close from the upper-left corner. However, if ROC curves considers true positive and false positive rates, it does not consider true negative and false negative ones. In order to better understand the detection performance, as well as the relationship between the threshold value and detection quality, Figure I.b describes the evolution of detection Matthews Correlation Coefficient (MCC) over threshold values. The best MCC values of the described experimentations are going increasingly from 0.7259 to 0.7804. Finally, Figure I.c describes the obtained Accuracy ACC values over the experimentations. ACC values corresponding to highlighted MCC values are going increasingly from 0.9906 to 0.9924. All best detection performance were obtained by using a threshold value between 1.18 and 1.20. It is also the case of all other experimentations with a sharp attack rate. We can observe that our system, detects sharp attacks with a great efficiency and accuracy.

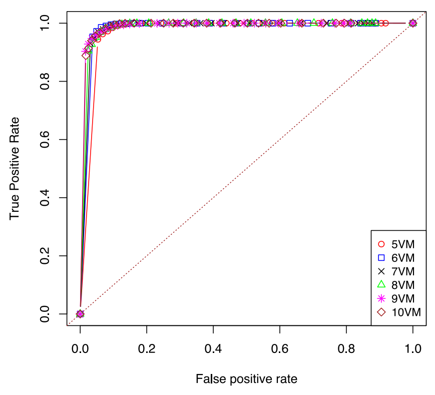
2) Weak attacks : Figure I.d, I.e and I.f represent the results describing experimentations where the botcloud realizes, for all n values, a weak attack. Figure I.d represent the ROC curves of the detection results. We can clearly note that the system has a lower detection performance than in case of a sharp attack. This decline of detection performance is explained by system's weak footprint. Indeed, when the attack rate is low, the system activity is almost imperceptible are and monitored metrics are less correlated than in a sharp attack case. As for the previous case, we use MCC as another indicator of detection performance, as well as to observe the relationship between the threshold value and detection quality. Figure I.e describes the evolution of MCC over threshold values while Figure I.f describes the obtained ACC values of the same experimentation. The best MCC values are going randomly from 0.1971 to 0.3073 and obtained through a threshold value between 1.44 and 1.50. ACC values corresponding to these results vary randomly between 0.8608 and 0.9434. We can observe that n parameter has no influence in detection quality. However, the detection is less efficient.

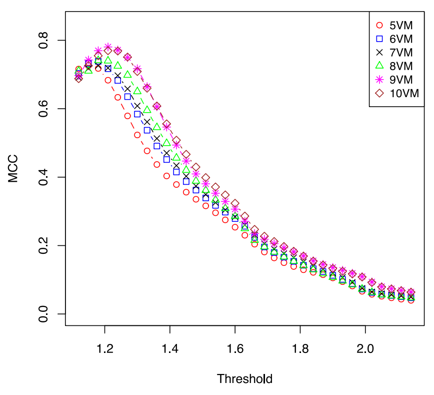
From the validation results, we can observe that the proposed system exhibits a better performance in detecting sharp attacks than in detecting weak ones. In addition, for sharp attacks, from the one hand, increasing the value of n enhance the detection; however, it increases the global computatation complexity. From the other hand, the worst case $n = 5$ still realizes good detection results. This demonstrates the reliability of the proposed system that still exhibits a high performance even if distributed over a set of distributed IDS. Finally, if the presented results are very promising regarding the design of such a distributed detection system, they have to be mitigated with real-world constraints. For instance, in a real infrastructure, the monitoring of nodes' activity will be asynchronous which is not the case of a simulation environment. The latter would lead to taking decisions on solely a part of the data, which would cause an efficiency deterioration as compared to the results presented above.

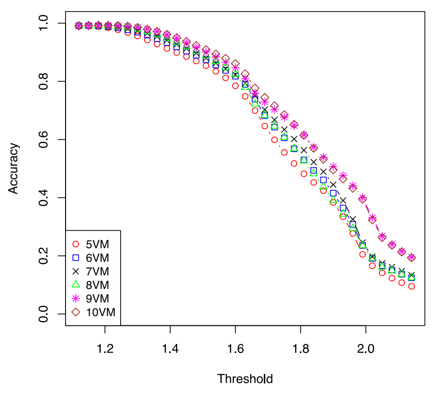
IV. Conclusion and future works

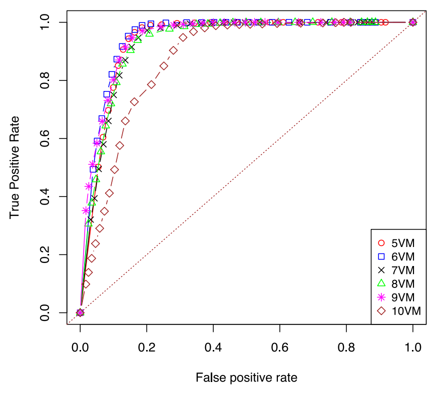
In this paper, we have tackled the problem of cloud's conversion into an attack support. We proposed a collaborative approach for a source-based detection of botclouds.

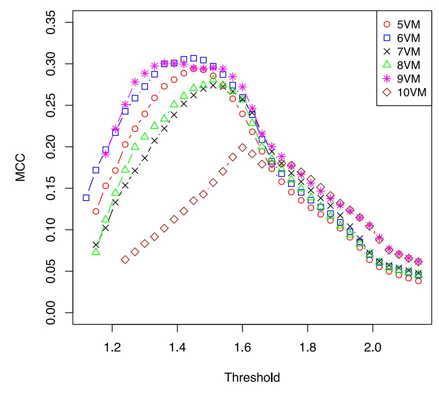
[2]http://www.r-project.org


(a)


(b)


(c)


(d)


(e)

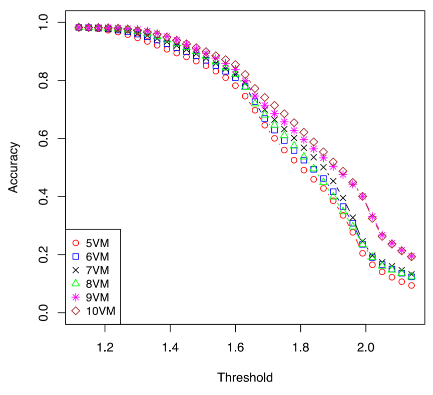

(f)

Table I: Results for sharp (a, b and c) and week attacks (d, e and f)

Our approach relies on an overlay network of hypervisor-level implemented IDS. The latter adopts a hierarchical architecture where the detection algorithm is applied on small sets of VMs. Such an architecture presents numerous advantages such as (1) a low complexity of detection algorithm, (2) a fast reaction time, (3) a scalability support. We validated and demonstrated the efficiency and reliability of our detection approach through early simulation results that relies on real workload traces.

Our future work will focus on, implementing the proposed approach in a real testbed, (2) realizing performant detection in a huge workload of data to reach our final goal which consists in (3) the development and implementation of an autonomous self-protection system for CSPs against attacks leveraged by a cloud infrastructure.

References

[1] Hammi Badis, Guillaume Doyen, and Rida Khatoun. Toward a source detection of botclouds: a pca-based approach. In *8th International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, 2014.

[2] Hammi Badis, Guillaume Doyen, and Rida Khatoun. Understanding botclouds from a system perspective: a principal component analysis. In *Network Operations and Management Symposium (NOMS 2014)*, pages 1–8. IFIP/IEEE, may 2014.

[3] Daniel Joseph Dean, Hiep Nguyen, and Xiaohui Gu. Ubl: unsupervised behavior learning for predicting performance anomalies in virtualized cloud systems. In *Proceedings of the 9th international conference on Autonomic computing*, ICAC '12, pages 191–200. ACM, 2012.

[4] Badis Hammi, Rida Khatoun, and Guillaume Doyen. A factorial space for a system-based detection of botcloud activity. In *6th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IFIP/IEEE, 2014.

[5] Tom Killalea. Rfc 3013, recommended internet service provider security services and procedures. 2000.

[6] J. Mirkovic and P. Reiher. D-ward: a source-end defense against flooding denial-of-service attacks. *Dependable and Secure Computing, IEEE Transactions on*, 2(3):216 – 232, july-sept. 2005.

[7] Hayati Pedram, Jindou Jia, and Rvacheva Daria. botcloud an emerging platform for cyber-attacks, October 2012. http://baesystemsdetica.blogspot.fr.

[8] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(3):241–280, 1999.