Toward a Source Detection of Botclouds: a PCA-based Approach

Hammi Badis[1], Guillaume Doyen[1], and Rida Khatoun[2]

[1] Université de Technologie de Troyes, ERA ICD UMR CNRS 6281
[2] Telecom ParisTech
{badis.hammi, guillaume.doyen}@utt.fr, rida.khatoun@telecom-paristech.fr

Abstract. Cloud computing security is often focused on data and users security and protection against external intrusions. However, it exists an area of cloud security that is often overlooked and that can have disastrous consequences: the conversion of cloud computing into an attack vector. Beyond a legitimate usage, the numerous advantages of cloud computing are exploited by attackers. Botnets supporting Distributed Denial of Service (DDoS) attacks are among the greatest beneficiaries of this malicious use. In this paper, we propose a novel source-based detection approach that aims at detecting the abnormal virtual machines behavior. The originality of our approach resides in (1) relying only on the system's metrics of virtual machines and (2) considering a source-based detection. Our approach is based on Principal Component Analysis to detect anomalies that can be signs of botcloud's behavior supporting DDoS flooding attacks. We also present the results of the evaluation of our detection algorithm.

1 Introduction

For the last few years, cloud computing has gained and is still gaining momentum. The reason lies in the numerous benefits it offers to its users, such as a fast deployment of services, a substantial reduction of both infrastructure and operation costs, a fair pay-per-use system, and all of this, while ensuring large scalability.

However, beyond the legitimate usage of these advantages, the latter are also exploited by malicious users, in order to use the cloud as a support for their attacks toward any third party connected to the Internet. Such a phenomenon represents a major issue since it strongly increases the power of distributed massive attacks while involving the responsibility of cloud service providers. The greatest beneficiaries of this cloud conversion into an attack support are botnets, which are called in this case botclouds. Indeed, a botcloud can be setup on demand and at very large scale without requiring a long dissemination phase nor expensive deployment costs. Botnets are primarily used to launch Distributed Denial of Service (DDoS) attacks which are considered among of the most dangerous ones. For instance, an experimental study [1] has shown how the cloud could be at the source of many attacks. To that aim, for five of the most famous

cloud service providers (CSP), the authors rent some virtual machines (VM), deployed and executed different attacks (e.g., DDoS, shellcode, malware traffic, malformed traffic, etc.) during a 21-day period. However, they did not encounter any reaction nor countermeasure from any of the cloud service providers. In the same issue, a group of researchers [2] have investigated how the cloud could be used to build a large botcloud, where they realized large DDoS flooding and click fraud attacks.

In this context, our goal is to develop a source-based 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. The originality of our work resides in (1) the consideration of system metrics in the detection of flooding denial of service attacks by considering (2) a source-based detection. Indeed, due to the impossibility to master the infected personal computers that arrange botnets, current DDoS and botnets detection solutions are solely based on a network approach and located at the target side [3]. However, the complete control of the attack support by the CSP, enables the consideration of system metrics that can facilitate botclouds detection and the consideration of a source-based detection approach which, to the best of our knowledge, have never been studied to date.

In this paper, we propose a novel method of using Principal Component Analysis (PCA) for Botclouds' supporting DDoS detection. We present the results of the validation of our approach through a simulation tool that relies on real traces that we obtained through *in situ* experimentations.

This paper is organized as follows: Section 2 gives an overview of the related works. Then, Section 3 describes the approach we propose for the detection of botclouds leveraging DDoS flooding attacks. Section 4 discusses the evaluation results. Conclusion and future work are given in Section 5.

2 Related works

2.1 Host based IDS

In [4], 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 which uses a set of continuous VM behavior learning modules to capture the patterns of normal operations of VMs relying on system metrics (CPU, MEM, TX, RX). To that aim, it leverages Self Organizing Map (SOM), an unsupervised learning method, to predict anomalies by looking at early deviations from the normal system behaviors. This work is related to ours since, it considers the same metrics as our system. However, the limit of UBL relies in monitoring computer activities on a single host. Thus, it does not enable the building of a global view of intrusions and is not effective in detecting fast-spreading attacks such as DDoS ones. Unlike [4], our approach relies on a signature-based approach implemented at the source hosts which do not need a learning phases such as in UBL.

2.2 Collaborative IDS

DDoS attacks represent large-scale coordinated attacks. Thus, in order to detect them efficiently, we need to combine the evidences of suspicious network or host activity from multiple distributed hosts and networks. To overcome the problem of IDS isolation, Collaborative Intrusion Detection Systems (CIDS) have been proposed to correlate suspicious evidence between different IDSs, thus improving the efficiency of intrusion detection. Several CIDSs have been proposed in the last few years. [5] represents a collaborative system that detects DDoS flooding attacks as far as possible from the victim host. It relies on a distributed architecture composed of multiple Intrusion Protection Systems forming overlay networks of protection rings around subscribed customers. [6] is a proposed gossip-based collaborative system of host based IDSs, which use distributed probabilistic inference to detect network intrusions. The system relies on a fully distributed architecture. [7] proposes a CIDS that uses the Chord DHT (Distributed Hash Table) system to organize IDSs into a P2P network. Each IDS shares its blacklist with others through a fully distributed P2P overlay. If a suspicious IP is reported more than a threshold N, then all the IDSs which reported it will be notified. The system relies only on IP addresses in the identification of potential intruders. Thus, it is not effective against worms having a low spreading degree (less than N). In [8], the authors proposed a hierarchical CIDS based on dependency. Participating hosts are clustered into cooperating regions and a Markov model is used to aggregate the alerts collected from the local hosts within the region. Then, sequential hypothesis testing is applied globally to correlate findings across regions.

2.3 Source-based IDS

Source-based detection approaches represent a very promising solution to detect large scale attacks and to avoid their side-effect damages. To the best of our knowledge, the sole attempt to design and implement a source-based DDoS detection system is presented in *D-WARD* [9]. In the latter, the authors proposed 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. Attacks are detected by the constant monitoring of two-way traffic flows between the network and Internet and a periodic comparison with normal flow models. However, the limitation of this solution is the large number of independent network administrative domains that must deploy it in order to be efficient.

2.4 Our previous work

In [10] and [11], we presented the results of an intensive measurement campaign we conducted in the aim of featuring and understanding a botcloud in its execution environment. We considered the case of a public CSP, providing an Infrastructure as a Service (IaaS), such as Amazon EC2. We reproduced

the case where a malicious user rents several virtual machines to host a botcloud intended to support DDoS attacks. The botcloud was implemented over Hybrid_V1.0 botnet[3]. We have realized transport layer DDoS flooding attacks (TCP SYN and UDP flooding). Indeed, the popularity of these attacks is due to their high effectiveness against any kind of service since there is no need to identify and exploit any particular flaws of victims' services. From a timeline perspective, all the experiments we conducted are composed of three phases, each lasting one hour. These are: (1) a first phase of normal state, where the botcloud is deployed, active but does not attack; (2) a second phase of attack toward a third party and finally (3) a third phase where the attack is stopped and the system comes back to normal. We have performed our experimentation over Planet-Lab [12] which relies on the LXC[4] project for virtualization. We used PlanetLab to face the need of an execution environment in which several tenants execute legitimate services while we are able to deploy and control a modified safe version of a botcloud. In order to maintain the privacy of the tenant's activity [13], we have limited the measurements to the sole metrics that are commonly available at the hypervisor level, thus operating in a black box way. As a result, the metrics we collect are: CPU (%), memory (MEM (KB/s)), bandwidth sent (TX (Kb/s)) and bandwidth received (RX (Kb/s)). The monitoring of tenants (slices) was performed every minute, through the Slicestat[5] service. Over the all measurement campaign, and beyond the results we present here, we have collected about 18 GBytes of log files.

In [10], we have highlighted, over a PCA [14], the correlations between the different collected system metrics of a botcloud. Indeed, in the case of UDP flood attack, CPU and TX metrics are positively and strongly correlated and both are negatively and strongly correlated with RX metric. Concerning the case of TCP SYN flood attack, there is a strong and positive correlation between the TX and RX metrics and both are strongly and negatively correlated with the CPU metric. In the second part of the work, we have detected and separated the attack phase from the idle one, for the two study-cases, namely, botclouds supporting UDP flood and TCP SYN flood attacks. We have also confirmed the hypothesis of the strong similarity of bots' behaviors.

In [11], we have shown, using PCA, that from a system perspective, whatever the attack rate, the contribution of metrics in modeling the botcloud's behavior is almost constant. In addition, the factorial space defined by the eigenvectors' matrix and which defines the botcloud's activity is also, almost constant. These results have led us to define the generic factorial space that represents the activity of a botcloud supporting a DDoS flooding attack. In this paper, we use this factorial space for the detection of botcloud's activity against legitimate workload.

---

[3] http://security-sh3ll.blogspot.com/2010/01/hybrid-botnet-system-v10-released.html

[4] https://linuxcontainers.org

[5] http://codeen.cs.princeton.edu/slicestat/

3 A source approach based on a PCA

In this section, we describe the approach we propose for a source-based detection of botclouds leveraging DDoS flooding attacks. First, we give an overview on PCA, the statistical method we used in our detection process. Then, we detail the steps of our detection approach. Table 1 describes the different notations used in the next sections.

X^k	The k^{th} $[n \times p]$ Data matrix	x_{ij}^k	The j^{th} variable of the k^{th} $[n \times p]$ Data matrix at time i
e_{it}^k	The i^{th} eigenvector of tenant k calculated at time t	w	Size of time monitoring window
M_t^k	$[p \times p]$ Eigenvectors matrix of tenant k calculated at time t	W_t^k	$[w \times p]$ Data matrix of last activity belonging to tenant k at t time
λ_{it}^k	The i^{th} eigenvalue of tenant k calculated at time t	W_{ij}^k	The j^{th} variable of W^k at time i
p	Number of variables of a matrix	S_t^k	normalized matrix of W_t^k
n	Number of rows of a matrix	S_{ij}^k	The j^{th} variable of S^k at time i
t	Time index	C_t^k	Covariance matrix of tenant k calculated at time t
m	Number of chosen Principal Components	vm_{ij}^{vk}	The j^{th} variable of v^{th} virtual machine belonging to tenant k at time i
k	Tenant's number	V^k	Number of VMs belonging to a tenant k
d_t^k	Decision made for tenant k at time t	v	VM's number
D_t^k	Dissimilarity value of M_t^k and R	R	Reference factorial space for a DDoS attack
H	Threshold	σ	Standard deviation

Table 1. Notations table

3.1 Principal Component Analysis

PCA [14] is a descriptive statistical method belonging to the factorial category. It is aimed at easing the exploration and analysis of high-dimensional vectors of input data by reducing their dimensions and enabling the extraction of features. Given a data matrix X^k of n observations, also called individuals, composed of p variables, the PCA explains the variance-covariance structure of the set of variables through a few new variables, called principal components or factors, which are functions of the original variables. Principal components represent linear combinations of the p variables with important properties: the computed principal components, which are in general 2 or 3, respectively have the highest

variances so that they best represent the data in a reduced dimension space and highlight their linear relations. Also, components are uncorrelated and the total variance of all the principal components equals the total variance of the original variables. More precisely, the principal components are computed by firstly, solving the eigenvalue problem of the variance-covariance matrix described by Equation 1

$$Ce_i = \lambda_i e_i \tag{1}$$

where C represents the variance-covariance matrix, λ_i $(i = 1, 2, \ldots, p)$ are the corresponding eigenvalues and e_i represents their corresponding eigenvectors. Secondly, computing the first m eigenvectors $(e_1, e_2, \ldots, e_m)$ which correspond to the m largest eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_m)$ where $(m < p)$.

Many works such as [15] and [16], rely on PCA in network intrusion detection. The authors in [15], have showed how PCA can be used for real-time anomaly detection. The authors in [16] have confirmed that a detection mechanism based on PCA can be suitable for large amounts of real time data.

As we need to detect attacks in a highly heterogeneous and dynamic workload, using PCA fits perfectly our study context, since it does not require any distribution model assumption on the data while many statistical based intrusion detection methods assume a normal or at least known distribution model.

3.2 Problem modeling and detection algorithm

he first step of the detection process, consists in data monitoring and collection, which is done for all VMs, belonging to all tenants. Especially, the metrics we consider are CPU (%), memory (MEM (KB/s)), bandwidth sent (TX (Kb/s)) and bandwidth received (RX (Kb/s)). Most data sets contain one or a few unusual observations [15]. When an observation is different from the majority of the data or don't assume the same statistical distribution model, it is considered as an outlier. The authors in [17], demonstrated that PCA is very sensitive to outlier data. Since we are interested in the global behavior of a tenant and not a particular VM, as a first filter for outlier data, we use an arithmetic average. That way, the detection algorithm uses, for each tenant, a data matrix X^k that represents the arithmetic average of the global activity of tenant k as input. An entry x_{ij}^k represents the average calculated on all the i^{th} rows corresponding to the j^{th} variable (column) of all the VMs $vm_{ij}^{vk}(v = 1, 2, \ldots, V_k)$, belonging to tenant k, as explained by Equation 2. Using the arithmetic average has no influence on our study. In fact, we have demonstrated in [10] that the behavior of a simple bot (VM) is very close to that of the global botcloud, due to the strong correlation of the different bots.

$$x_{ij}^k = \frac{1}{V^k} \sum_{v=1}^{V^k} vm_{ij}^{vk} \tag{2}$$

Our detection algorithm always relies on the previous activity of the tenant, and detects any new changes in its behavior. We represent this previous activity by a sliding window of time which we name W_t^k.

$$W_t^k = \begin{bmatrix} x^k_{(t-w)\ 1} & \cdots & x^k_{(t-w)\ p} \\ \vdots & \ddots & \vdots \\ x^k_{t\ 1} & \cdots & x^k_{t\ p} \end{bmatrix}$$

The second step, consists in normalizing the data of W_t^k by calculating it's standard score S_t^k as explained by Equation 3. The reason of using normalized data resides in the difference of scales in which metrics and data are collected. Furthermore, using normalized data allows a better control on the detection threshold.

$$S_{ij}^k = \frac{W_{ij}^k - \overline{W}^k_{.j}}{\sigma(W_{.j}^k)} \tag{3}$$

Contrarily to the basic PCA presented in Section 3.1, our approach, relies on the computation of the whole eigenvectors $(e_{1t}^k, e_{2t}^k, \ldots, e_{pt}^k)$ related to all the eigenvalues $(\lambda_{1t}^k, \lambda_{2t}^k, \ldots, \lambda_{pt}^k)$ of Equation 1, which constitutes the third step in the detection process. That way, our approach considers all the data set without any information loss. The covariance matrix used in the eigenvectors computation are obtained through Equation 4.

$$C_t^k = \frac{1}{n}(S_t^k)^T \times S_t^k \tag{4}$$

The set of the p eigenvectors constitutes a factorial space, represented by M_t^k the $p \times p$ eigenvector matrix, $M_t^k = [e_{1t}^k, \ldots, e_{pt}^k]$. This factorial space entity represents the heart of our approach. Indeed, in [11], we showed that the factorial space composed by the eigenvectors, defines tenants' activities in general and the botcloud's activity in particular. We also showed that, whatever the attack rate, the factorial space of a botcloud is almost constant. In addition, it appears always as an outlier of legitimate activities. All that has led us to define a generic factorial space that represents a botcloud realizing a DDoS flooding attack. The latter serves as a reference R for our detection algorithm. In other words, unlike most of approaches that use PCA for outlier detection, our approach compares the workload with a defined outlier pattern, which is the factorial space R. Any similarity with this pattern is considered as a potential attack.

Consequently, the fourth step, is the comparison of the calculated M_t^k with R. To that aim, we use the notion of dissimilarity between matrices. The dissimilarity measure between the two matrices which we name D_t^k, is calculated by the Frobenius norm $||M_t^k - R||_F$ [18] as depicted in Equation 5

$$D_t^k = ||M_t^k - R||_F = \sqrt{\sum_{i=1}^{p}\sum_{j=1}^{p} |(M_t^k - R)_{ij}|^2} \tag{5}$$

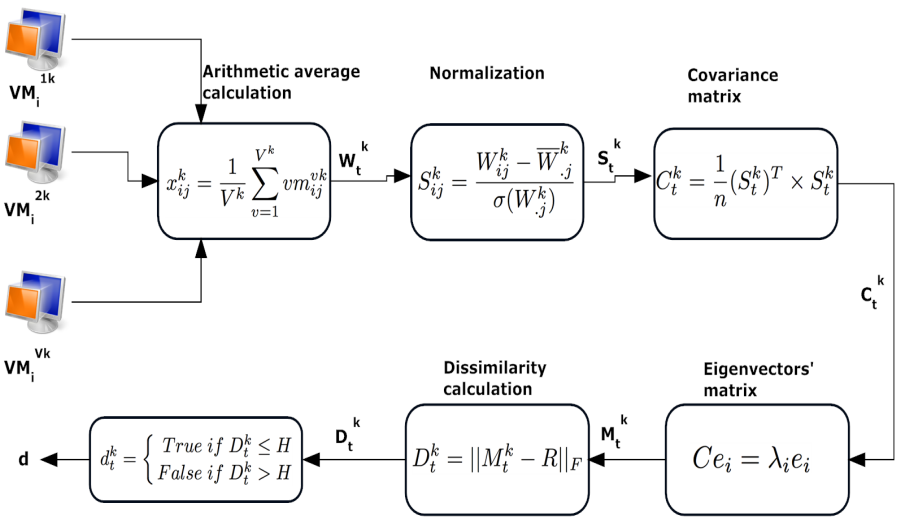

Fig. 1. Detection algorithm steps

Finally, we fix a threshold that we name H to decide whether or not the dissimilarity D_t^k value stands for a DDoS attack. The decision rule d_t^k is given by

$$d_t^k = \begin{cases} True\ if\ D_t^k \leq H \\ False\ if\ D_t^k > H \end{cases} \tag{6}$$

The Figure 1 summarizes the described detection steps in detail.

4 Evaluation and discussion

4.1 Evaluation framework

In this section we present the results we obtained for the evaluation of our detection algorithm. We note that our algorithm treats simultaneously all monitored tenants.

For the evaluation, we have used the real traces (logs) we obtained every minute in [10] by injecting them into a simulator built on the R tool[6]. Table 2 summarizes the different elements we monitored and that characterize these experimentations. We have realized five experimentations. For each experiment, we have a botcloud realizing an UDP flood attack at different rates, from 8 MB/s up to 80 MB/s per source, reaching aggregated attack rates of 328 MB/s to 3.125 GB/s. For each experiment, we present the results we obtained through

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

facing the botcloud's activity to 25 other legitimate tenants picked randomly, which represents a large workload of hundreds of VMs.

In order to evaluate our detection algorithm, we have varied the decision threshold H from 1.29 to 1.42. 1.29 constitutes the lower bound of H because it represents the highest dissimilarity value between the reference factorial space R and the botcloud while realizing the different attacks. Regarding the upper bound, we have varied H by steps of 0.03 till having 5% of error rate, which corresponds to the 1.42 value. For the current study, we initialized our algorithm with $w = 7$. Thus, the detector relies on a sliding window W of 7 minutes. The choice of w represents the subject of a study that is left for future work.

UDP flood attack rate	#Physical servers	#Tenants (incl. attack.)	#VMs (incl. attackers)	#Attacking VMs
8 MB/s	41	123	1,288	41
16 MB/s	41	118	1,261	41
40 MB/s	43	123	1,310	43
56 MB/s	41	114	1,241	41
80 MB/s	40	103	1,198	40

Table 2. Summary of the scenarios' numerical parameters

4.2 Evaluation results

We have computed the confusion matrix of all the simulations we conducted. A confusion matrix contains information about actual and predicted classifications done by a classification system. From the latter, we have calculated different statistical indicators such as Receiver Operating Characteristic (ROC) curves, Accuracy, Error rate, Matthews Correlation Coefficient, Positive and Negative Predictive Values.

Accuracy and error rate The accuracy[7] of a measurement system is the degree of closeness of measurements of a quantity to that quantity's true value. It has a value between 0 and 1. Figure 2.a describes the accuracy values obtained over the five experimentations. We note that the highest ACC we obtained corresponds to $H = 1.32$, which are 0.9955, 0.9944, 0.9957, 0.9955 and 0.9939. These Accuracy values reflect the efficiency of our algorithm. In similar way, the error rate measurement which represents the opposite of the accuracy and defined by the remoteness of measurements of a quantity to that quantity's true value. So, lower is the error rate, more reliable is the detection. Figure 2.b

[7] http://www.bipm.org/en/publications/guides/

describes the error rates obtained over the five experimentations. We note that the lowest ERR rates we obtained are 0.0045, 0.0056, 0.0043, 0.0045 and 0.006, which corresponds to $H = 1.32$.

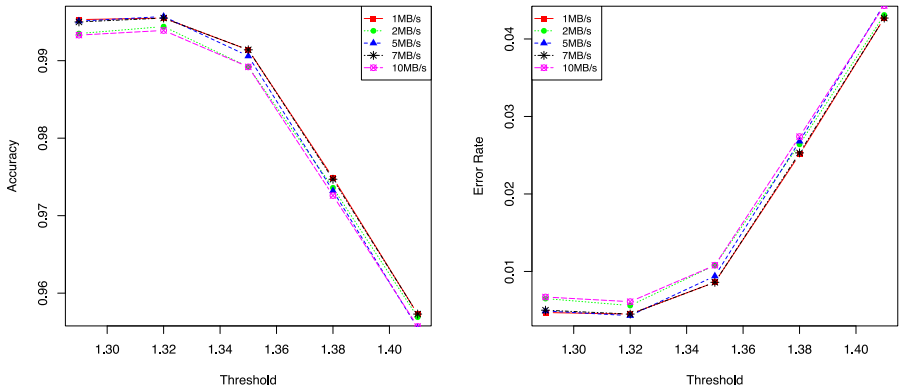

Fig. 2. Accuracy and Error rate graphs ; (a) Accuracy (b) Error rate

Positive and Negative Predictive Values The positive and negative predictive values (PPV and NPV respectively) are the proportions of positive and negative results in statistics and detection tests that are true positive and true negative results. The PPV and NPV describe the performance of the detection. Closer are PPV and NPV values from 1, better is the detection. Figure 3 describes the obtained values of PPV and NPV over the five experimentations. For both measures the highest values we obtained over the experimentation belongs to $H = 1.32$. The PPV obtained values are: 0.9474, 0.9500, 0.9545, 0.9535 and 0,9444. Concerning the NPV measure we always got values over than 0.99. These values are very close from 1, which again reflects the efficiency and good prediction of our algorithm.

ROC curves and Matthews correlation coefficient Figure 4.a represents the different ROC curves we obtained. In the five cases, the drawn curve is always widely on top of the bisector, which prove the efficiency of our algorithm.

The Matthews correlation coefficient (MCC) [19] is used as a measure of the quality of binary (two-class) classifications. It represents a correlation coefficient between the observed and predicted binary classifications; it returns a value between -1 and +1. A coefficient of +1 represents a perfect prediction, 0 no better than random prediction and -1 indicates total disagreement between prediction

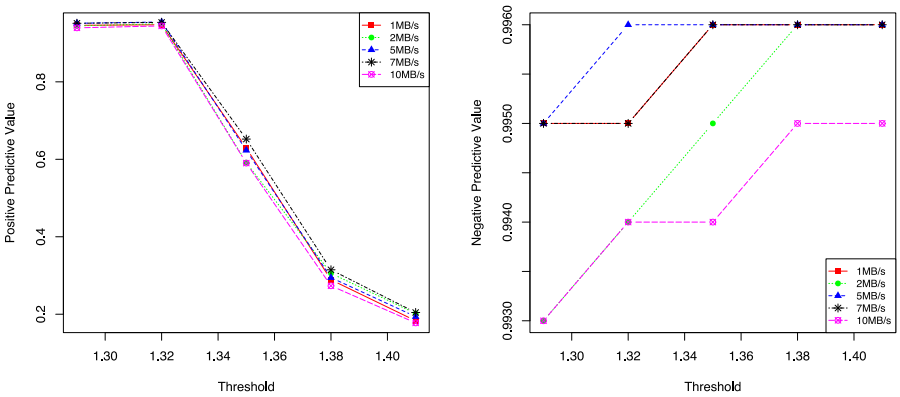

Fig. 3. Predictive values graphs; (a) Positive Predictive Values; (b) Negative Predictive Values

and observation. Figure 4.b describes MCC values we obtained over the experimentation. The highest MCC values we obtained belongs also to $H = 1.32$, which are 0.7927, 0.7669, 0.8224, 0.8120 and 0.7352. These results reflect the good prediction of our algorithm.

The consistency of the results over the threshold variation confirms the efficiency of our algorithm. We also note that the best threshold we got is 1.32. Over this threshold, we have got a very good performance: an Accuracy rate always over 99%, an Error rate always less than 0.6%, a PPV value rate always over than 94% and a NPV value always higher than 99%. These results are obtained through the comparison with a generic factorial space obtained through a botcloud realizing attacks at defined rates (between 8MB/s and 80MB/s per source). The best detection results are obtained through the experimentation at 40MB/s (median value of attack rates). Indeed, we got better performances through 40MB/s experiment because the respective botcloud's factorial space is the closest from R.

The obtention of such very good results is due to the use of the arithmetic average in a centralized approach. However, due to the complexity of the detection algorithm, a distributed approach is mandatory. Consequently, we plan to study such an implementation to see whether the detection performances undergo degradations.

5 Conclusion and future work

In this paper, we have addressed the problem of the conversion of the cloud computing into an attack vector. To remediate the problem, we have presented a novel approach based on PCA and system metrics, which enables a source-based

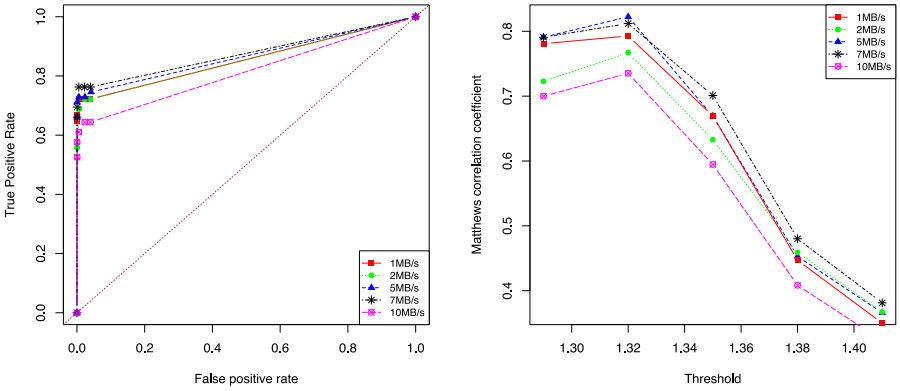

Fig. 4. ROC and Matthews correlation coefficient; (a) ROC curves; (b) Matthews correlation coefficient

detection of DDoS flooding attacks in a cloud computing environment. We have validated our approach by a simulation approach that relies on a real traces that we obtained through *in situ* experimentations. We have proved the efficiency and resiliency of our detection algorithm over the different statistics that took into account dozens of tenants that involve hundreds of virtual machines activities which represents a large workload amount.

These results only represent a step of our work. A short-term future work will focus on proposing a fully distributed approach of our detection algorithm, able to deal with scalability issues. Mid-term future work consists in extending this study in order to characterize other attacks such as application level attacks and even to detect infected legitimate VMs exhibiting an almost an almost normal behavior. Finally, our long-term research direction will look at the development of an autonomous self-protection system for CSPs against DDoS attacks leveraged by a cloud infrastructure.

Acknowledgment

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

References

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

2. Kassidy P. Clark, Martijn Warnier, and Frances M. T. Brazier. Botclouds - the future of cloud-based botnets? In Frank Leymann, Ivan Ivanov, Marten J. van Sinderen, and Boris B. Shishkov, editors, *Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011)*, pages 597–603. Science and Technology Publications, 2011.
3. T. Peng, C. Leckie, and K. Ramamohanarao. Survey of network-based defense mechanisms countering the DOS and DDOS problems. *ACM Computing Surveys (CSUR)*, 39(1), 2007.
4. 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.
5. Jérôme François, Issam Aib, and Raouf Boutaba. Firecol: a collaborative protection network for the detection of flooding DDoS attacks. *IEEE/ACM Trans. Netw.*, 20(6):1828–1841, 2012.
6. Denver Dash, Branislav Kveton, John Mark Agosta, Eve Schooler, Jaideep Chandrashekar, Abraham Bachrach, and Alex Newman. When gossip is good: Distributed probabilistic inference for detection of slow network intrusions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1115–1122. AAAI Press, 2006.
7. C.V. Zhou, S. Karunasekera, and C. Leckie. A peer-to-peer collaborative intrusion detection system. In *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on*, volume 1, page 6, 2005.
8. Ji Li, Dah-Yoh Lim, and Karen Sollins. Dependency-based distributed intrusion detection. In *Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, DETER, pages 8–8. USENIX Association, 2007.
9. 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, 2005.
10. 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), Accepted Paper*. IFIP/IEEE, may 2014.
11. Hammi Badis, Rida Khatoun, and Guillaume Doyen. A factorial space for a system-based detection of botcloud activity. In *Sixth IFIP International Conference on New Technologies, Mobility and Security (NTMS'2014), Accepted Paper*. IFIP/IEEE, March 2014.
12. Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.*, 33(3):3–12, 2003.
13. Joep Ruiter and Martijn Warnier. Privacy regulations for cloud computing: Compliance and implementation in theory and practice. In Serge Gutwirth, Yves Poullet, Paul De Hert, and Ronald Leenes, editors, *Computers, Privacy and Data Protection: an Element of Choice*, pages 361–376. Springer Netherlands, 2011.
14. Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.

15. Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. A novel anomaly detection scheme based on principal component classifier. Technical report, DTIC Document, 2003.
16. D. Brauckhoff, K. Salamatian, and M. May. Applying pca for traffic anomaly detection: Problems and solutions. In *INFOCOM 2009, IEEE*, pages 2866–2870, April 2009.
17. Yuh-Jye Lee, Yi-Ren Yeh, and Yu-Chiang Frank Wang. Anomaly detection via online oversampling principal component analysis. *Knowledge and Data Engineering, IEEE Transactions on*, 25(7):1460–1470, 2013.
18. Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 2. Siam, 2000.
19. Pierre Baldi, Søren Brunak, Yves Chauvin, Claus AF Andersen, and Henrik Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, 16(5):412–424, 2000.