Detecting Botclouds at Large Scale: A Decentralized and Robust Detection Method for Multi-Tenant Virtualized Environments

Rémi Cogranne, Guillaume Doyen, Nisrine Ghadban, Badis Hammi

Detecting Botclouds at Large Scale: a Decentralized and Robust Detection Method for Multi-Tenant Virtualized Environments

Rémi Cogranne, *Member, IEEE,* Guillaume Doyen, Nisrine Ghadban and Badis Hammi

***Abstract*—Cloud computing has gained an important role in providing high quality and cost-effective IT services by outsourcing part of their operations to dedicated cloud providers. If intrinsic security issues of this architecture have been extensively studied, it has recently been considered as a ready-to-use platform able to perform malicious activities, thus offering new targets for indirect threats. However, its large scale, the heterogeneous and dynamic nature of the activities it executes, as well as multi-tenancy and privacy-related issues, make the security operation complex. Consequently, cloud providers can hardly detect and mitigate malicious activities they unknowingly host. Leveraging the autonomic paradigm represents a promising solution to face such a complexity, but it requires efficient grounded monitoring and analysis functions to efficiently detect malicious activities hidden within the large number of legitimate ones. In this effort, this paper presents a robust and cost-effective solution to detect malicious activities in a public virtualized environment. Its contribution is twofold: (1) a scalable and robust workload estimation of the virtual host activities in a cloud and (2) a detection algorithm able to discriminate infected hosts with low malicious activities hidden within their legitimate workload and potentially scattered on several tenants. For both of these contributions, we establish their theoretical performance, which demonstrates their optimality, and we evaluate their efficiency on a dataset made of real data collected on *PlanetLab*. Finally, we study the scalability on a large dataset that consists of simulated data resulting from the real dataset modeling. This demonstrates to what extent the proposal exhibits an excellent sharpness and a reasonable cost, even at a very large scale.**

***Index Terms*—Decentralized algorithm, Big Data, Source-end anomaly detection, Scalable methods, Low-rate DDoS, Hypothesis testing theory.**

The work on this paper has been co-funded by (1) the French Investment for Future (Développement de l'Economie Numérique), Request (REcursive QUEry and Scalable Technologies) project, started in 01/02/2014 and (2) the CRCA and FEDER Transdisciplinarity CyberSec Platform <D201304601> ; www.cybersec.utt.fr.

Rémi Cogranne is with the Lab. for System Modelling and Dependability, ICD, UMR 6281 CNRS, Troyes University of Technology, Troyes, France. Guillaume Doyen is with the Autonomous Network Environment Team, ICD, UMR 6281 CNRS, Troyes University of Technology, Troyes, France. Nisrine Ghadban was with the Autonomous Network Environment Team, ICD, UMR 6281 CNRS, Troyes University of Technology, Troyes, France when this work has been done, she is now with the faculty of engineering, Lebanese University. Badis Hammi is with Telecom ParisTech, France.

Email : {remi.cogranne ; guillaume.doyen ; nisrine.ghadban }@utt.fr ; badis.hammi@telecom-paristech.fr

Corresponding author: Rémi Cogranne

I. Introduction

TODAY, cloud computing is a widely adopted solution for the production of IT services and it has become an indispensable actor in the operation of various infrastructures. The cloud is an IT model where massively scalable information technology capabilities are delivered as a service and on-demand to an outside clientele using Internet technologies. It offers many advantages, such as rapid deployment of services, cost reduction for its users, billing on demand and scalability of support infrastructure. Despite the cloud's benefits for legitimate users, malicious users can use it as a large-scale and ready-to-use platform that eases the deployment of an attack toward any third party connected to the Internet. A Stratsec study [1] was conducted in 2012 to investigate the potential benefits of a cloud service infrastructure for a malicious third party. The experiment consisted in purchasing Infrastructure as a Service (IaaS), offered from five major cloud service providers in current market, in order to exploit their services and produce different types of attacks against a third party. The experiments lasted 21 days in total and 48 hours for Distributed Denial of Service (DDoS) without interruption. However, no response was received from any of the service providers.

Many cases of abusive use of cloud resources are reported in the literature: Clark et al. [2] showed how the cloud can be exploited to create a large Botcloud, in a few minutes using a minimum of effort and going through the operation of Amazon EC2 services in order to carry out DDoS and click-fraud attacks. In 2011, researchers at Kaspersky[1]reported that cybercriminals had been using Amazon Simple Storage Service (Amazon S3) as a launching point for their SpyeEye botnet operation for at least a couple of weeks. Even more,

[1]securelist.com/amazon-s3-exploiting-through-spyeye-13/

according to Solutionary firm [3], cloud services like Google and Amazon AWS host numerous instances of malware, primarily from US-based cloud servers. Recently, the Booters phenomenon has risen [4] thus providing an interface for DDoS as a Service. The attack generated by a Booter depends on the Botnet behind it and its power makes its notoriety. Thus, in order to strengthen the Booter’s power, cloud resources represent a boon.

The success of these experiments demonstrates that the simplicity, robustness, flexibility and low cost of cloud services make it an appropriate solution for those who wish to host malicious services. Indeed, Cloud Service Providers (CSPs) make it possible to use their resources without strong security mechanisms. Ragan et al. [5] tested in 2014 the account creation process for more than 150 CSPs and showed that only a third of them required any credentials beyond an email address, additional information like a credit card, phone number, or filling out a captcha.

This paper is a contribution to the source-end detection of malicious activities a CSP can host unknowingly. Given the large spectrum of such activities, leveraging a network-based anomaly detection approach, as commonly done in the state of the art, is not satisfying. Indeed, malicious activities may exhibit a very low or even null networking activity while consuming other resources (e.g. CPU, memory, storage). Brut-force attacks, which consist in leveraging huge computing resources to crack a set of passwords, are an example of such malicious activities that illustrates the need for host-based detection approaches able to capture any abnormal usage of available resources. However, in this context, implementing such a detection solution presents several challenges that our research work faces:
1) Generic detection approach and heterogenous legitimate activities: Knowing that in the context of a public CSP, malicious software can execute any sort of malicious activity, an appropriate detection approach must be sufficiently generic and scalable to allow the detection of any form of abnormal activity. Additionally, the heterogeneous and highly dynamic nature of the legitimate workload a cloud provider hosts must be considered to enable a sharp discrimination of malicious activities in such an unstable environment.
2) Data volume and velocity: Due to the volume and velocity of the data generated by monitoring probes, whose load is related to the number of virtual hosts[2] executed in a cloud computing infrastructure, it is imperative to consider an efficient but low-cost approach in terms of resources and computing time that can support the scaling factor.
3) Confidentiality and privacy: Monitoring user activities with probes located at the tenant level induces legal problems [6] related to the privacy of the cloud users. Therefore, the proposed solution must take it into account by using a non-intrusive approach that only solicits data available to a cloud provider such as those made available at the virtualization layer.

The contribution of the present paper stands for a practical and efficient method for source-end detection of malicious activities in large scale virtualized environments. By source-end detection, it is meant that the malicious activities are identified by inspecting the activities of virtual hosts in clouds. This approach thus allows an immediate identification of infected virtual hosts and consequently the mitigation of the attack, prevents any collateral damage when targeting a remote third party, and avoids the difficult problem of tracing back an attack. In order to deal with the dynamics and large scale of public cloud infrastructures, the proposed method relies on a fully decentralized model leveraging an incremental Principal Component Analysis (PCA), where a detection engine exhibiting a low computation cost is hosted in each server of the cloud infrastructure. There, each detection engine is associated to the set of virtual hosts the server executes and all the engines can communicate to infer any global information. As such it is thought to scale up to possible extremely large number of servers, tenants and virtualized hosts, which stands for the scaling factors of current and future cloud infrastructure. Among the different malicious activities a virtualized infrastructure can host, we consider the case of DDoS attacks. If the latter are easy to detect from a networking perspective at any aggregation point toward their target (e.g. egress router of a cloud infrastructure), they are harder to discriminate from a system perspective from other activities, especially when, within the virtual host which executes their supporting bot, they are mixed with a legitimate activity.

This paper further extends prior work by the authors. It leverages the workload estimation engine presented in [7] and the dataset presented in [8] (other previous works in that field such as [9] are out of the scope of this contribution since they do not follow the same research methodology). The original contributions presented in this paper are: (1) a statistical method for the detection of infected virtual hosts executing a malicious activity potentially scattered over several tenants; (2) the theoretical assessment of the method performance; (3) its validation in terms of performance on a real dataset integrating a botnet perpetrating a DDoS attack and (4) its scalability support by leveraging a large-scale realistic dataset generated from the real data.

The paper is organized as follows. Section II provides an overview of the contributions taking part in our research field. Section III formally states the problem of source-end detection of malicious activities in virtualized hosts. Then, section IV presents a decentralized methodology for workload estimation which acts as the ground for a behavioral anomaly detection. The problem of malicious activity detection is addressed in Section V where an optimal statistical test is presented and its statistical performance is established analytically. Section VI exhibits results demonstrating (1) the relevance of our approach to accurately evaluate a cloud workload; (2) the capability of our detection solution to detect botnet activities under various conditions, and (3) it assesses the scalability support of our overall contribution in terms of virtual hosts, tenants and servers. Finally, Section VII concludes the paper.

[2]In the following, we denote virtual hosts to indifferently refer to virtual machines or containers.

II. Related Work

A substantial part of the present work lies at the intersection of the three following research domains: (1) workload estimation in cloud environments (2) distributed Principal Component Analysis (PCA) for Big Data and (3) source-end DDoS Detection. We survey all of them below.

A. Cloud Workload Estimation

The problem of workload characterization and prediction has been widely studied [10], [11]. Among the proposals, some research focuses on creating mathematical and statistical models to characterize the workload in a cloud environment. The classification of tasks based on CPU and memory usage in [12] relies on the statistical identification of qualitative coordinates (i.e., small, medium, large). A similar approach is applied in [13] to study the CPU and memory usage of tasks and jobs and discover task shapes and duration. In [14], periodicity and patterns with homogeneous behaviors are identified with spectral and autocorrelation analyses. The two works which are the closest to ours are (1) a decentralized clustering approach for a dynamic mix of heterogeneous applications in cloud environments [15] and (2) a gossip protocol for dynamic resource management in large cloud environments [16]. The proposed mechanisms respectively leverage autonomic control and decentralized algorithms to handle an efficient VMs provisioning regarding resource utilization. In comparison, the first part of our work does not aim at classifying activities but rather provides the best average estimation of the global activity. Besides, our method does not rely on a priori knowledge or model but is fully data-driven for high adaptivity while being completely decentralized.

B. Principal Component Analysis

Principal component analysis (PCA) is a powerful technique for analyzing and identifying patterns in data. It finds the most important axes to express the scattering of data by determining the subspace which holds the largest variance. This subspace is spanned by the principal axes and the projection of data in this subspace constitutes the principal components, which reflect the approximate distribution of data.

1) Background: PCA applied on the data matrix $\mathbf{X}$ of V observations (in column), also called individuals, composed of p variables, solves the eigenvectors decomposition problem:

$$\mathbf{C}\mathbf{w}_i = \lambda_i \mathbf{w}_i, \qquad i = 1, 2, \cdots, p, \tag{1}$$

where $\mathbf{C}$ is the covariance matrix of matrix $\mathbf{X}$, calculated when the observations have zero means by $\mathbf{C} = \frac{1}{V}\mathbf{X}^\top\mathbf{X}$ with $\mathbf{X}^\top$ the transpose of $\mathbf{X}$. The values $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ represent the eigenvalues sorted in descending order and $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_p]$ is the corresponding eigenvector matrix, where $\mathbf{w}_i$ is the i^{th}-axis direction. The mapping of the data to principal axis $\mathbf{w}_i$, which is referred to as the i^{th} principal axis, whose variance is λ_i , is given by the projection onto w_i: $\mathbf{Y}_i = \mathbf{w}_i^\top \mathbf{X}$. In other words, $\mathbf{Y}_i$ represents the contribution of the i-th axis $\mathbf{w}_i$.

The computational complexity for calculating the covariance matrix is $\mathcal{O}(Vp^2)$, and it requires $\mathcal{O}(p^2)$ memory storage units. The computational complexity of the eigen-decomposition problem is $\mathcal{O}(p^3)$ [17]. To this cost, one should also include the communication costs, which is $\mathcal{O}(Vp)$ over a distance $\mathcal{O}(1)$.

2) PCA in Big Data: PCA is widely used in Big Data context [18] since it is usually leveraged as a first step in data mining. However, because of the complexity of the covariance matrix computation, PCA still appears as a challenging problem when dealing with large datasets, and hence an active research topic. In this area, the main strategies to make PCA scalable, consists in reducing the dataset in order to allow the computing of PCA. In [19] for instance, coresets are defined as small sets that provably approximate the original data. This method is based on *merge-and-reduce* that permits the solving of computational problems such as PCA in parallel. In [20], a model based on PCA is built on a small subset of a large network producing a large amount of data. The principal components of this smaller subgraph allow the out-of-sample extension property.

3) Distributed PCA Approaches: Several attempts have been made to alleviate the problem of scalability of PCA in networking contexts by mainly distributing the computation. This was proposed for instance in [21], [22] where algorithms are investigated to compute the eigenspace, but still with high computational costs. Recently, in [23], collective-PCA technique was proposed, in which a fusion center only receives the principal components, instead of the whole time series. In [24], [25], the most relevant principal axis is estimated by the power iteration method. However, this method requires the computation of the sample covariance matrix and since the latter can only be achieved by gathering all the dataset on one single node, it is inappropriate for large-scale data. In [25], the power iteration method is used but with sparse matrices in order to reduce computational complexity. In [26], the covariance matrix is first estimated by means of a consensus averaging algorithm, then each node performs a local eigenvector decomposition. The Distribute Adaptive Covariance Matrix Eigenvector Estimation (DACMEE) algorithm, presented in [27], recursively updates the eigenvector estimates without explicitly constructing the full covariance matrix that defines them. Nodes share only the first fused observation and compute compressed covariance matrices. However, they still require the eigen-decomposition of several matrices which does not fit the context of large-scale systems. Similarly, Candid Covariance-free fast Incremental PCA (CCIPCA) [28] algorithm offers a very good compromise between statistical accuracy and computational speed. It also has the advantage of not having major dependence on tuning parameters. However, this incremental algorithm is centralized and not adapted for solutions built on decentralized approaches.

C. Source-end Detection of DDoS Attacks

Distributed Denial of Service attacks (DDoS) have been widely studied in the literature [29]–[31], since they represent one of the main attacks in computer networks due to their easiness to perform and important impacts [32]. Recent works typically focus on widely distributed low rate attacks. A notable prior work in this area is for instance [33] that uses edge network routers in order to identify attack sources and to mitigate or discard attack traffic. Those approaches can either be easily scalable when edge network routers are detecting attacks independently, with the drawback of not leveraging information from other routers to improve detection performance. On the opposite, some prior work focuses on distributed or collaborative method for detection [32], [34]. Such approaches exploit information from different routers at the cost of complexity and detection delay and the scalability support of such approaches still has to be established.

If target level and intermediate level detection approaches have been extensively studied [31], [32], [35], there are only few works treating source-end detection due to implementation and design complexity. Indeed, a source-end solution needs to be implemented in a large number of locations and a dedicated collaboration of the different instances is required, in order to be efficient. The authors in [36] proposed EDS, an extrusion detection system, that relies on source and destination IPs and port numbers of each packet in order to draw a graph that detects the attacks. The proposed method suffers from a severe scalability issue and could not handle the huge amount of cloud's workload. In [37], the authors proposed Botcloud detection method, using Artificial Immune System (AIS). Through this process, they can detect the probability of botnet infection based on the process of independent Poisson process and detection based on negative selection. The authors in [38] proposed EyeCloud, a system that detects members of Botcloud that abuse cloud resources. To reach its goal, EyeCloud relies on Virtual Machine Introspection (VMI). The resulting data is then treated by clustering methods. The authors in [39] proposed *srcTrace*, a method for the detection of DDoS attacks generated by a Botcloud. The latter has two components *Bot_Chase* and *Mp_Trace* and relies on the analysis of traffic flows entropy variation. The authors of [40] proposed a method for the detection of VMs members of a Botcloud. The method relies on entropy calculation on each of the selected parameters, for each VM. After that, a *k-means* clustering method is applied on the obtained values.

To conclude, to the best of our knowledge, if cloud infrastructure still suffers from malicious activities they host unknowingly, there is currently no dedicated solution able to (1) analyze the activity of each host individually (2) capture the large-scale and dynamic nature of such infrastructure and (3) detect low footprint malicious activities, possibly distributed over several tenants and hidden within legitimate workload. These open problems motivate the present work that proposes a generic and robust method to address them.

III. Problem Formalization and Methodology

The present paper proposes a host-based approach whose purpose is to detect virtual hosts infected by botnets performing a malicious activity. An overall picture of the detection system in a typical multi-tenant architecture is depicted in Figure 1. It shows different servers hosting a virtualization layer executing different virtual hosts belonging to different tenants; one tenant may have several virtual hosts located on different servers. Monitoring probes are located on the physical host and use the virtualization layer to capture metrics related to the usage of resources consumed by the virtual hosts. The metrics are related to resources offered by the CSP to the tenants through virtualization means. They are typically the processing usage, memory consumption or network input/output. The probes can communicate together and have the knowledge of the belonging of virtual hosts to a given tenant. They also compute the detection algorithm we present subsequently and raise local alarms in case of abnormal behavior detection. In this context, our detection approach leverages a two-step approach for which we formalize the problem subsequently. The first consists in estimating the cloud workload to enable the detection algorithm to get the reference of what a normal behavior is, while the second compares outlier virtual hosts to the signature of well-known resource consumption attack patterns.

Symbol	Denotation
T	Number of tenants
S	Number of servers
V	Number of VMs
M	Number of metrics for system activity
L	Number of measurements used for detection
$\mathbf{x}_{t,v}$	System metrics from virtual machine v at time t
$\mathbf{C}$	Covariance matrix
$\mathbf{w}$	Principal axis
λ	Eigen value
η	Estimate of $\lambda\mathbf{w}$
ϕ	Intermediate estimate of $\mathbf{w}$ at servers level
$\psi^\star$	Ideal estimation of $\mathbf{w}$ at tenants level
ψ	Approximation of $\psi^\star$; intermediate estimate of $\mathbf{w}$ at tenant level
t	Index of time measurement for system activity
i	Iteration index
ϵ	Convergence error threshold
θ	Time of data extraction
Θ	Angle between the i^{th} estimated and real principal axis

TABLE I: Notations used to describe our algorithm
.

For clarity purposes, all notations used in the following are provided in Table I.

A. Cloud Workload Estimation

Generally speaking, let us denote vector $\mathbf{x}_{t,v} \in \mathbb{R}^M$ the data activity from virtual host v, $v \in \{1,\dots,V\}$ at time t, with M the metrics related to the activity of the virtual host.

Our estimation approach relies on a statistical model grounded by a PCA of the virtual host activities, stating that all virtual hosts can be classified into coarse grain behaviors [41]; typically some activities are related to *the computation* with the use of CPU and memory while some others are mostly related to *networking* by sending or receiving data over the network. It is worth noting that in the present work, we aim at estimating a coarse grain classification of activities independently from the activity volume. Consequently, we propose to model the legitimate activity of virtual hosts using the following linear model:

$$\mathbf{x}_{t,v} \approx \widehat{\mathbf{W}}_{t,v}\mathbf{c}_{t,v} \qquad (2)$$

where matrix $\widehat{\mathbf{W}}_{t,v}$ represents the estimation of r , $k < M$ main principal components and vector $\mathbf{c}_{t,v}$ is the weighting vector that represents the contribution of each principal component for the specific measurement of system activity metrics $\mathbf{x}_{t,v}$. Such a linear model presents a good trade-off between modeling accuracy and simplicity to design a statistical test.

One can note that in Eq. (2), as opposed to the PCA background provided in Section II-B1, the PCA depends on time t and virtual host v. Since computational and communication costs prevent from carrying out PCA calculations under such conditions, we propose a novel strategy to ensure the scalability support of the computation. The latter relies on two main points: (1) a fully decentralized approach which prevents any computation element to act as a bottleneck, and (2) the avoidance of the sample covariance matrix computation and its eigen-decomposition, thus allowing to significantly reduce the computational complexity.

B. Abnormal Activity Detection

On the basis of the workload activity estimation, the detection algorithm enables the detection of outlier virtual hosts and especially those performing a malicious activity with a low footprint, covered within a much larger activity related to all legitimate virtual hosts. To that aim, we leverage the data-driven nature of PCA which should not be impacted by sparse activities, thus providing an overall detection accuracy.

Let us denote $\mathbf{a}$ the vector that represents the impact of a malicious activity on the infected virtual hosts and let $\mathbf{e}_{t,v} = \mathbf{x}_{t,v} - \widehat{\mathbf{W}}_{t,v}\mathbf{c}_{t,v}$ represent virtual host v workload estimation error. The problem of malicious activity detection can be formalized as a choice between the following hypotheses:

$$\begin{cases} \mathcal{H}_0 & : \ \mathbf{x}_{t,v} = \widehat{\mathbf{W}}_{t,v}\mathbf{c}_{t,v} + \mathbf{e}_{t,v}, \\ \mathcal{H}_1 & : \ \mathbf{x}_{t,v} = \widehat{\mathbf{W}}_{t,v}\mathbf{c}_{t,v} + \mathbf{e}_{t,v} + \mathbf{a}. \end{cases} \quad (3)$$

This formalization of the problem allows to clearly understand the main underlying difficulties. First, as discussed in subsection II-B, computing an accurate estimation of the principal components $\widehat{\mathbf{W}}_{t,v}$ for large-scale and dynamic data sources is challenging and requires a dedicated approach to circumvent the data volume and velocity issues. Second, the trace of the malicious activity $\mathbf{a}$ on the system cannot be known, especially because its operating mode largely influences the measurable impact on system activity. Formally, in such a case, the hypotheses are referred to as being composite and an optimal solution scarcely exists. Third, the metrics are, by far, not independent and their correlation must thus be carefully taken into account. Similarly, the legitimate activity $\widehat{\mathbf{W}}_{t,v}\mathbf{c}_{t,v}$, which refers to a nuisance parameter, has no interest for the detection of the malicious activity, while it must be removed with caution in order to maintain a high detection accuracy. Last, mastering the detection false alarm rate *a priori* together with all the previously identified challenges, while being crucial to provide an accurate and flexible detection solution, remains of an overall complexity.

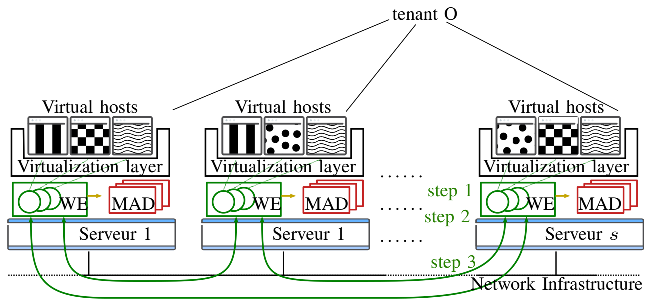


Fig. 1: Overview of the proposed multi-tenant workload estimation and Botcloud detection. Step 1 represents the local Workload Estimation (WE) update at virtual hosts level. Step 2 represents the averaging over servers and step 3 is the random selection of virtual hosts for a gossip exchange of information. Once the iterative WE is carried out, the proposed statistical test allows Malicious Activity Detection (MAD).

IV. Decentralized Estimation of Legitimate Activities

We introduce a decentralized version of Candid Covariance-free fast Incremental PCA (CCIPCA) algorithm [28], to incrementally compute the principal components of matrix $\mathbf{X}_t$ without estimating the covariance matrix. In order to satisfy with the privacy respect of tenants' activities, we briefly describe the proposed *combine and adapt* method for principal components estimation. The main iterative process of the method is based on the three following steps. First, at the level of each physical server, the estimation of the principal components of the virtual hosts it executes are averaged, regardless of their tenant owner. Second, this averaged value is shared between probes associated with virtual hosts that belong to the same tenant. This leverages the belonging of a virtual host to a dedicated tenant. Those two steps form the *combine* phase. Last, the *adapt* phase consists in adjusting, for each virtual host, the combined estimation with its system activity. This approach offers the advantages of (1) allowing a good adaptivity with respect to dynamic variations of activities in time, (2) separating the heterogeneous behaviors within different similar activities, and (3) representing the activities with axes gathering similar activities regardless of the volume of those activities.

Given the PCA computation reminded in section II-B, in our decentralized method, for each virtual host with index v, each probe replaces the covariance matrix $\mathbf{C}$ by a local estimate $\mathbf{x}_{t,v}\mathbf{x}_{t,v}^\top$. Thus, at iteration step i, virtual host v is associated with the estimation of the principal components denoted $\eta_{t,v}(i)$ with, at initialization, $\eta_{t,v}(0) = \mathbf{x}_{t,v}$. This is represented as step 1 in Figure 1.

In the original method [28], the first *combine* phase consists in gathering all estimations from all nodes. Since this is hardly possible in a large-scale cloud infrastructure, we propose to split this phase into two steps, the first consisting in averaging, for each server, all the estimates $\eta_{t,v}(i)$ from the virtual hosts it executes:

$$\phi_{t,s}(i) = \frac{1}{|\mathcal{V}_s|} \sum_{v \in \mathcal{V}_s} \eta_{t,v}(i). \quad (4)$$

In Equation (4), s represents the index of a server, $\mathcal{V}_s$ represents the set of all virtual hosts on server s, $|\mathcal{V}_s|$ denotes the cardinality of set $\mathcal{V}_s$, that is the number of hosts on server s and the notation $v \in \mathcal{V}_s$ means that the virtual host with index v is hosted on server s ; the sum over all $v \in \mathcal{V}_s$ thus represents the sum over the set of all virtual hosts hosted on a given server s. This is represented as step 2 in Figure 1.

The second step for combining estimations is very similar but performed for all virtual hosts belonging to a particular tenant. Ideally, we would like to obtain the average of all estimations $\phi_{t,s}(i)$ from VMs belonging to the tenant owner O:

$$\psi^{\star}_{t,O}(i) = \frac{1}{|\mathcal{S}_O|} \sum_{s \in \mathcal{S}_O} \phi_{t,s}(i), \tag{5}$$

with, similarly to (4), $\mathcal{S}_O$ the set of all servers that host at least one host that belongs to tenant owner O, $|\mathcal{S}_O|$ the number of servers in set $\mathcal{S}_O$; the sum over all $s \in \mathcal{S}_O$ thus represents the sum over the set of all servers that executes a host belonging to tenant owner O.

It is worth noting that combining first estimation from all virtual hosts from the same server, regardless of their tenants, and then combining the estimation over virtual hosts that belong to the same tenant, regardless of the server that hosts them, allows to quickly combine the estimations from a wide range of virtual hosts at a small communication and computational costs, while preserving tenant privacy.

Due to the same scalability reasons, which may make the communication cost to compute Equation (5) prohibitive, we propose to perform the computation of the estimate $\psi^{\star}_{t,O}(i)$ at a reasonable communication cost, by replacing it by the approximation obtained using a symmetric gossip as described in [42], [43]. More precisely, at iteration t, the probe associated with virtual host v randomly selects one of its neighbors u, belonging to the same tenant, and they both exchange estimations, obtained at the server level, to perform an update as follows:

$$\begin{aligned} \psi_{t,v}(i) &= (1-\rho) \times \phi_{t,v}(i) + \rho \times \phi_{t,u}(i), \quad (6)\\ \psi_{t,u}(i) &= \rho \times \phi_{t,v}(i) + (1-\rho) \times \phi_{t,u}(i), \quad (7) \end{aligned}$$

with $\rho \in [0,1]$ a mixing parameter that defines the rate of update. It is worth noting that the estimations at tenant level may differ from a virtual host to another due to the random selection of a neighbor. Hence we adopt the notation $\psi_{t,v}(i)$ instead of ψ_T. Since the updates of the principal axes described in Equations (6)-(7) are performed for all virtual hosts of all tenants, the latter should have very similar estimations at the end of the gossip process. This is represented as step 3 in Figure 1.

Finally, the last step of the workload estimation consists in adapting the estimations $\eta_{t,v}(i)$ at the virtual hosts level, by taking into account the combined estimations obtained at the tenant level $\psi_{t,v}(i)$. To this end, we propose to use the CCIPCA the amnesic factor proposed in [28] that allows forgetting the oldest estimates and hence allows handling the dynamicity of system activity. Adapting the CCIPCA to the case of dynamic leads us to redefine the update as follows:

$$\eta_{t,v}(i) = \frac{i-1-l}{i}\psi_v(i-1) + \frac{1+l}{i}\mathbf{x}_{t,v}^{\top}\mathbf{x}_{t,v}\frac{\psi_{t,v}(i-1)}{\|\psi_{t,v}(i-1)\|}, \tag{8}$$

where the positive parameter l is referred to as the amnesic parameter, which gives more weight to new samples to gradually decrease the effect of oldest ones. Note that the two modified weights still sum to 1.

Beyond the basic operations presented above, we propose two improvements which fasten the convergence of the estimations. The first improvement, related to the estimation of the principal components, simply consists in initializing the method with the latest estimates computed at the previous time $\eta_{t,v}(0) = \eta_{t-1,v}(i)$. This improvement relies on the fact that, though cloud activity is dynamic, it is not likely to change abruptly very often. The second improvement is mainly related to the detection accuracy. It consists in using the K latest measurements of system activity instead of the latest one only. Indeed, one can replace in all the previous equations vector $\mathbf{x}_{t,v}$ by matrix $(\mathbf{x}_{t-K+1,v}, \dots, \mathbf{x}_{t-K+1,v})$ at a very small extra computational cost. This second improvement does not help much the estimation of the principal components but it greatly helps to ensure that after the malicious activity starts, it will not be modeled within the main principal components by adding a lot more of less recent activity measurements.

Finally, one can note that the previous description only focuses on the estimation of the first main principal component. The estimation of other principal components, associated with smaller eigenvalues, can be computed using the orthogonality of the eigenvectors. By using the Gram-Schmidt process, it is thus possible to subtract the first axis $\eta_{t,v}(i)$ from the data contribution with the following projection:

$$\mathbf{x}^{(1)}_{t,v} = \mathbf{x}_{t,v} - \eta_{t,v}(i)\frac{\mathbf{x}_{t,v}^{\top}\eta_{t,v}(i)}{\|\eta_{t,v}(i)\|^2}, \tag{9}$$

and to iteratively compute the other principal components by applying the aforementioned method using the results from Equation (9).

V. An Optimal Source-end Detection Method

In this section, we demonstrate to what extent, by leveraging the workload estimation presented in the previous section, we are able to detect malicious activities which are concealed in infected virtual hosts. The rejection approach we follow has been widely used in literature [44]. However the specificity of the proposed approach is that, thanks to the PCA, the proposed linear model is data driven, built upon the data themselves. In a second time, to feature the exact nature of the malicious activity, we do rely on a signature-based approach where we compare the residual activity of virtual hosts to a given footprint.

A. Legitimate Workload Rejection

From empirical data, described in Section VI-A, it is proposed in the present paper to model distribution of estimation

errors $\mathbf{e}_{t,v} = \mathbf{W}^{\perp}_{t,v}\mathbf{x}_v$ (15) as a realization of multivariate Gaussian random variables. In addition, we noted that the attack traces depends linearly on the attack payload, or bandwidth: the Botcloud activity is roughly a linear function of the attack payload.
With those assumptions[3], the detection problem (3) can be formalized as a choice between the following statistical hypotheses:

$$\begin{cases} \mathcal{H}_0 \ : \{\mathbf{x}_{t,v} \sim \mathcal{N}(\mathbf{W}_{t,v}\mathbf{c}_{t,v}, \mathbf{\Sigma})\}\,, \\ \mathcal{H}_1 \ : \{\mathbf{x}_{t,v} \sim \mathcal{N}(\mathbf{W}_{t,v}\mathbf{c}_{t,v} + p\overline{\mathbf{a}}, \mathbf{\Sigma})\}\,. \end{cases} \tag{10}$$

where $\mathcal{N}(\mu, \sigma^2)$ represents the Gaussian distribution with expectation μ and variance σ^2, p represents a malicious activity payload factor and $\overline{\mathbf{a}}$ the intrinsic reference of the malicious activity for a standard payload.

As stated in Section III, the main challenges here consists in (1) dealing with nuisance parameter $\mathbf{W}_{t,v}\mathbf{c}_{t,v}$, (2) considering the covariance $\mathbf{\Sigma}$ between metrics and eventually, (3) designing a statistical test with properties known *a priori*. Subsequently, we first deal with the case of the nuisance parameters.

The covariance matrix can be estimated in the basis of the dataset of all the observations. It is thus easy to normalize the data as follows:

$$\mathbf{x}^{\star}_{t,v} = \mathbf{\Sigma}^{-1/2}\mathbf{x}_{t,v}, \tag{11}$$

where $\mathbf{\Sigma}^{-1/2}$ is the matrix defined as $\mathbf{\Sigma}^{-1/2}\mathbf{\Sigma}^{-1/2} = \mathbf{\Sigma}^{-1}$. It follows, from Equation (11) and from the properties of affine transformation of Gaussian multivariate random variables, that the distribution of $\mathbf{x}^{\star}_{t,v}$, after normalization, is given by:

$$\begin{cases} \mathcal{H}_0 \ : \left\{\mathbf{x}^{\star}_{t,v} \sim \mathcal{N}(\mathbf{H}_{t,v}\mathbf{c}_{t,v}, \mathbf{I})\right\}\,, \\ \mathcal{H}_1 \ : \left\{\mathbf{x}^{\star}_{t,v} \sim \mathcal{N}(\mathbf{H}_{t,v}\mathbf{c}_{t,v} + p\mathbf{\Sigma}^{-1/2}\overline{\mathbf{a}}, \mathbf{I})\right\}\,. \end{cases} \tag{12}$$

with $\mathbf{H} = \mathbf{\Sigma}^{-1/2}\mathbf{W}_{t,v}$ and $\mathbf{I}$ the identity matrix. One can note that the Gaussian distribution remains invariant under affine transformation [46, Chap. 6]. The proposed approach thus exploits invariance principle to transform the problem stated in Equation (10) into the easier one stated in Equation (12) while proving their equivalence. This allows the estimation of parameters of a linear estimation with independent Gaussian noise which has been well studied. In such a case, it is known that the least square estimation coincides with the maximum likelihood estimator and is given by: $\widehat{\mathbf{c}}_{t,v} = \left(\mathbf{H}^{\top}_{t,v} \times \mathbf{H}_{t,v}\right)^{-1}\mathbf{H}^{\top}_{t,v}\mathbf{x}^{\star}_{t,v}$.

With the estimated parameter $\widehat{\mathbf{c}}_{t,v}$, the estimation of the legitimate activity expectation is given by:

$$\widehat{\mathbf{x}}^{\star}_{t,v} = \mathbf{H}_{t,v}\widehat{\mathbf{c}}_{t,v} = \mathbf{H}_{t,v}\left(\mathbf{H}^{\top}_{t,v}\mathbf{H}_{t,v}\right)^{-1}\mathbf{H}^{\top}_{t,v}\mathbf{x}^{\star}_{t,v} \tag{13}$$

[3]These assumptions are verified in the numerical results provided in Section VI-B. Besides, we have conducted a standard (Mardia's) test for multivariate Gaussian distribution, which are based on skewness and kurtosis of whitened data: both tests failed to reject the multivariate normal distribution at the significance level of 5%. Eventually, note that the model of Botcloud activity has been adopted for clarity and simplicity and can easily be adapted with a specific behavior for each metric.

Therefore, it is straightforward from (13) to remove the estimated legitimate activity from the measured metrics as follows:

$$\mathbf{x}^{\star}_{t,v} - \widehat{\mathbf{x}}^{\star}_{t,v} = \mathbf{x}^{\star}_{t,v} - \mathbf{H}_{t,v}\left(\mathbf{H}^{\top}_{t,v}\mathbf{H}_{t,v}\right)^{-1}\mathbf{H}^{\top}_{t,v}\mathbf{x}^{\star}_{t,v} = \mathbf{H}^{\perp}_{t,v}\mathbf{x}^{\star}_{t,v}, \tag{14}$$

where matrix

$$\mathbf{H}^{\perp}_{t,v} = \mathbf{I} - \mathbf{H}_{t,v}\left(\mathbf{H}^{\top}_{t,v} \times \mathbf{H}_{t,v}\right)^{-1}\mathbf{H}^{\top}_{t,v} \tag{15}$$

represents the projection onto the orthogonal complement of the subspace spanned by the main principal components after removing covariance $\mathbf{\Sigma}^{-1/2}\mathbf{W}_{t,v}$.

Eventually, putting the definition of linear model of observations $\mathbf{x}_{t,v} = \mathbf{W}_{t,v}\mathbf{c}_{t,v} + \mathbf{e}_{t,v}$ and $\mathbf{H} = \mathbf{\Sigma}^{-1/2}\mathbf{W}_{t,v}$ into the method of rejection in Equation (14), a short algebra shows that:

$$\begin{aligned} \mathbf{x}^{\star}_{t,v} - \widehat{\mathbf{x}}^{\star}_{t,v} &= \mathbf{H}^{\perp}_{t,v}\mathbf{\Sigma}^{-1/2}\left(\mathbf{W}_{t,v}\mathbf{c}_{t,v} + \mathbf{e}_{t,v}\right), \\ &= \mathbf{H}^{\perp}_{t,v}\mathbf{H}_{t,v}\mathbf{e}_{t,v} + \mathbf{H}^{\perp}_{t,v}\mathbf{H}_{t,v}\mathbf{\Sigma}^{-1/2}\mathbf{e}_{t,v} = \mathbf{H}^{\perp}_{t,v}\mathbf{H}_{t,v}\mathbf{\Sigma}^{-1/2}\mathbf{e}_{t,v}. \end{aligned}$$

This shows that the efficiency of the rejection approach to exploit the geometrical properties of the data, here standing for the activity metrics, can project the data onto the subset spanned by the orthogonal complement of $\mathbf{H}_{t,v}$.

B. Source-end Statistical Detection of Malicious Activities

Given the geometric approach for the nuisance parameters rejection described above and formulated in Equations (13)-(14) as well as the affine property of Gaussian distribution, the statistical problem of source-end malicious activity detection can now be written as a choice between the following statistical hypotheses:

$$\begin{cases} \mathcal{H}_0 \ : \left\{\mathbf{H}^{\perp}_{t,v}\mathbf{x}^{\star}_{t,v} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{H}^{\perp}_{t,v}\right)\right\}\,, \\ \mathcal{H}_1 \ : \left\{\mathbf{H}^{\perp}_{t,v}\mathbf{x}^{\star}_{t,v} \sim \mathcal{N}\left(p\mathbf{H}^{\perp}_{t,v}\mathbf{\Sigma}^{-1/2}\overline{\mathbf{a}}, \mathbf{H}^{\perp}_{t,v}\right)\right\}\,. \end{cases} \tag{16}$$

To address this statistical problem, it appears that the optimal detector, maximizing the detection accuracy for a prescribed false alarm rate [47], is simply given by the projection of the residuals $\mathbf{H}^{\perp}_{t,v}\mathbf{x}^{\star}_{t,v}$ onto the vector $p\mathbf{H}^{\perp}_{t,v}\mathbf{\Sigma}^{-1/2}\overline{\mathbf{a}}$ of expectation under hypothesis $\mathcal{H}_1$. This optimal test, often referred to as the *match space detector* [48] is formally given by:

$$\delta(\mathbf{x}^{\star}_{t,v}) = \begin{cases} \mathcal{H}_0 \ \text{if} \ \Lambda(\mathbf{x}^{\star}_{t,v}) \leq \tau, \\ \mathcal{H}_1 \ \text{if} \ \Lambda(\mathbf{x}^{\star}_{t,v}) > \tau, \end{cases} \tag{17}$$

where it is proposed to normalize the generalized likelihood ratio $\Lambda(\mathbf{x}^{\star}_{t,v})$ such as:

$$\Lambda(\mathbf{x}^{\star}_{t,v}) = \frac{\overline{\mathbf{a}}^{\top}\mathbf{\Sigma}^{-1/2}\mathbf{H}^{\perp}_{t,v}\mathbf{x}^{\star}_{t,v}}{\left\|\mathbf{H}^{\perp}_{t,v}\mathbf{\Sigma}^{-1/2}\overline{\mathbf{a}}\right\|_2}. \tag{18}$$

One of the main advantages of the proposed methodology is that the statistical performance of the proposed test can be analytically established. To this end, one needs to establish the statistical property of the generalized likelihood ratio $\Lambda(\mathbf{x}^{\star}_{t,v})$. While the expectation under hypothesis $\mathcal{H}_0$ is straightforward to establish, the calculation of expectation under $\mathcal{H}_1$ and of the variance requires the following short algebra:

$$\begin{aligned}\left(\mathbf{H}_{t,v}^{\perp}\boldsymbol{\Sigma}^{-1/2}\overline{\mathbf{a}}\right)^{\top}\mathbf{H}_{t,v}^{\perp}\mathbf{x}_{t,v}^{\star} &= \overline{\mathbf{a}}^{\top}\boldsymbol{\Sigma}^{-1/2}\mathbf{H}_{t,v}^{\perp}\mathbf{H}_{t,v}^{\perp}\mathbf{x}_{t,v}^{\star},\\ &= \overline{\mathbf{a}}^{\top}\boldsymbol{\Sigma}^{-1/2}\mathbf{H}_{t,v}^{\perp}\mathbf{x}_{t,v}^{\star}.\end{aligned}$$

because matrices $\boldsymbol{\Sigma}^{-1/2}$ and $\mathbf{H}_{t,v}^{\perp}$ are both symmetric and because $\mathbf{H}_{t,v}^{\perp}\mathbf{H}_{t,v}^{\perp} = \mathbf{H}_{t,v}^{\perp}$ since $\mathbf{H}_{t,v}^{\perp}$ is an orthonormal projector.
Using the distribution of $\mathbf{H}_{t,v}^{\perp}\mathbf{x}_{t,v}^{\star}$ as given in (16), by applying once more the affine transformation of Gaussian one has:

$$\begin{cases}\mathcal{H}_0 : \left\{\Lambda(\mathbf{x}_{t,v}^{\star}) \sim \mathcal{N}(0,1)\right\},\\ \mathcal{H}_1 : \left\{\Lambda(\mathbf{x}_{t,v}^{\star}) \sim \mathcal{N}(p\left\|\mathbf{H}_{t,v}^{\perp}\boldsymbol{\Sigma}^{-1/2}\mathbf{a}^{\star}\right\|_2, 1)\right\}.\end{cases} \quad (19)$$

Establishing the false-alarm probability is straightforward from Equation (19) as, from the definition of the cumulative distribution function, one has:

$$\begin{aligned}\alpha_0(\tau) &= \mathbb{P}_{\mathcal{H}_0}\left[\Lambda(\mathbf{x}_{t,v}) > \tau\right] = 1 - \mathbb{P}_{\mathcal{H}_0}\left[\Lambda(\mathbf{x}_{t,v}) \leq \tau\right]\\ &= 1 - \Phi(\tau).\end{aligned} \quad (20)$$

Equation (20) also immediately allows the setting of the decision threshold τ to achieve a prescribed false alarm rate as: $\tau(\alpha_0) = \Phi^{-1}(1-\alpha_0)$. Note that, here, Φ and Φ^{-1} respectively denote the Gaussian standard cumulative distribution function and its inverse.

Conversely, the probability of correctly detecting a malicious activity with a payload p, also referred to as the power function, is given by:

$$\begin{aligned}\beta(p,\alpha_0) &= \mathbb{P}_{\mathcal{H}_1}\left[\Lambda(\mathbf{x}_{t,v}) > \tau(\alpha_0)\right]\\ &= 1 - \Phi\left(\tau - p\left\|\mathbf{H}_{t,v}^{\perp}\boldsymbol{\Sigma}^{-1/2}\mathbf{a}^{\star}\right\|_2\right),\\ &= 1 - \Phi\left(\tau - \Phi^{-1}(1-\alpha_0)\left\|\mathbf{H}_{t,v}^{\perp}\boldsymbol{\Sigma}^{-1/2}\mathbf{a}^{\star}\right\|_2\right)\end{aligned} \quad (21)$$

Finally, to improve the detection accuracy, instead of testing each metrics observed at each time $t, t-1, \ldots$ independently, we propose to consider system activities over a longer period of time, thus bringing more information to the detection framework. For simplicity and computational efficiency, it is proposed here to apply the detection process only every L samples. Consequently, we propose to leverage a fixed-window size statistics, which is defined as:

$$\Lambda^{(L)}(\mathbf{x}_{t,v}, \ldots, \mathbf{x}_{t-L+1,v}) = \frac{1}{\sqrt{L}}\sum_{u=t-L+1}^{t}\Lambda(\mathbf{x}_{u,v}), \quad (22)$$

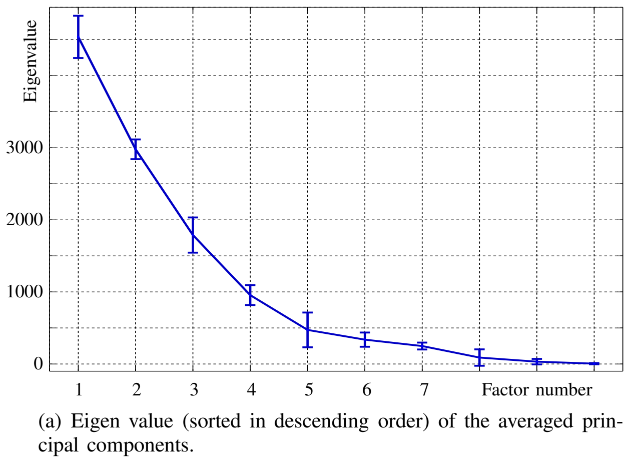
with $\Lambda(\mathbf{x}_{u,v})$ the detection statistic at time u as defined in (17). It can be shown that, adapting the algebra detailed in [49], [50], gathering L independent metrics of system activity, the performance of the test proposed in (22) is given by:

$$\alpha_0^{(L)}(\tau) = 1 - \Phi(\tau), \quad (23)$$

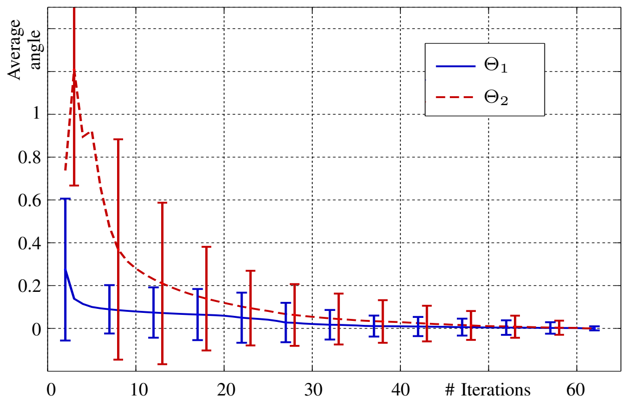
for the false alarm probability while the detection accuracy is defined as:

$$\beta^{(L)}(p,\tau) = 1 - \Phi\left(\tau - \sqrt{L}p\left\|\mathbf{H}_{t,v}^{\perp}\boldsymbol{\Sigma}^{-1/2}\overline{\mathbf{a}}\right\|_2\right). \quad (24)$$

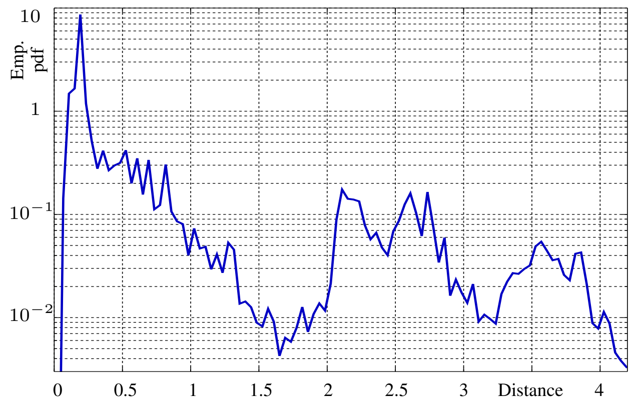
The previous results (20)-(24), related to the performance of the proposed source-end detection method for malicious activities, exhibit relevant operational properties. First of all, relation (23) emphasizes the possibility for the user to set a desired false alarm rate and to adjust the threshold accordingly, without needing any extra parameter or knowledge. Second, Equation (21) provides valuable insights by explicitly showing



(a) Eigen value (sorted in descending order) of the averaged principal components.



(b) Averaged angle between estimated first two principal components vs. ground truth.



(c) Distribution of distance between estimated activity and real, measured, activity.

Fig. 2: Overall workload estimation performance: (a) Relationship between the relative magnitude of the eigenvalues and the number of factors; (b) Convergence of the proposed strategy, measured by the average angle $\Theta_1 = \text{acos}(\mathbf{w}_1^{\top}\widetilde{\mathbf{w}}_1)$ and $\Theta_2 = \text{acos}(\mathbf{w}_2^{\top}\widetilde{\mathbf{w}}_2)$; (c) Density of the distances between real data and its estimate (semi-log scale)

how the detection depends mostly on two parameters only, that are, the desired probability of false alarm α_0 and the malicious activity payload p. The detection power, explicitly provided in (24) also points out the tradeoff between the detection delay L and the accuracy. Indeed, the greater L is, the larger the detection delay is and, more generally, the less reactive the window-based detection system (22) is. However, gathering samples allows the increasing of the detection accuracy for the same false alarm probability. Eventually, it is worth noting that the detection also depends on the norm of the projected attack through the term $\left\|\mathbf{H}_{t,v}^{\perp}\mathbf{\Sigma}^{-1/2}\overline{\mathbf{a}}\right\|_2$. This norm is difficult to measure in practical cases. However, it reveals relevant insights. Indeed, this term shows that the more the malicious activity is in line with legitimate activities, the more the projection onto $\mathbf{H}_{t,v}^{\perp}$ will reduce the detection power. This is why we emphasize that the proposed method can hardly work for very large malicious activities, as the latter would fall into the principal components. In practice we note, however, that malicious activities have specific footprints and that they are likely distinguishable from legitimate ones, though part of them will unavoidably be modeled as part of legitimate ones.

VI. Numerical Results and Experimentations

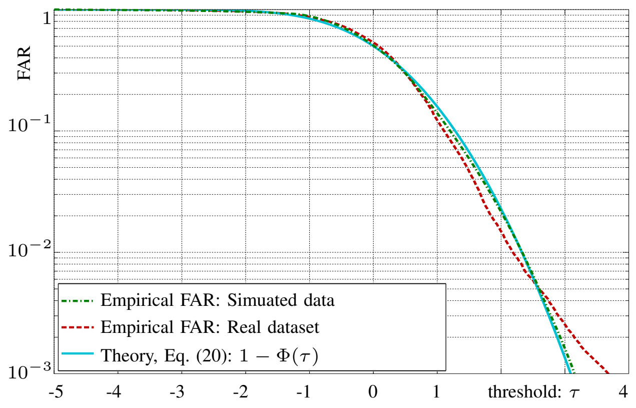
In this section, we present the numerical results we obtained by evaluating our approach. To that aim, we implemented both the workload estimation algorithm as well as the detection one in Matlab. As input data, we considered two datasets. The first one was collected on *PlanetLab*, which leverages a container-based virtualization technology, during a large-scale measurement campaign while the second one is a simulated dataset produced by synthesizing these real data. The general methodology used to generate the latter is provided in appendix A. First, we describe the overall scenario we consider. Second, we demonstrate to what extent our theoretical models are compliant with the reality of the algorithms and data. Third, we study the detection performance under various conditions, considering a botnet performing a DDoS attack. Four and last, we push forward the scalability of the cloud infrastructure we consider to evaluate the scalability support of our whole approach.

A. Scenario and Datasets

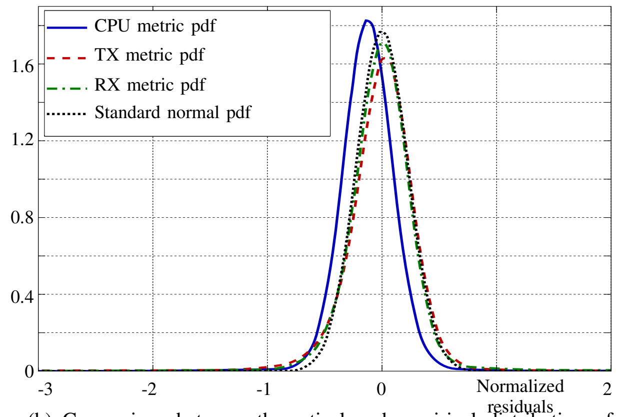
1) Measuring Legitimate Activities: The first dataset we consider for our evaluation consists in the capture of container activities from more than 40 *PlanetLab* servers hosting between one and two thousand containers that belong to about 120 tenants. The system traces have been harvested 15 times within 3 hours with a sampling frequency of one second using the *Slicestat*[4] tool. Over the whole dataset, this represents a number of distinct 97 physical servers, 254 tenants and 6426 containers for a total of more than 253 million system traces.

Among all the metrics captured through *Slicestat*, we kept the CPU activity (in %) the memory usage (in KB and in %), the size of virtual memory used, and the sending and receiving rates averaged over one minute, five minutes and

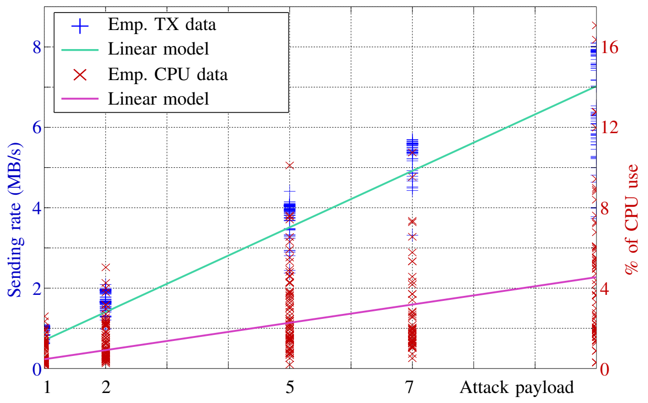
[4]codeen.cs.princeton.edu/slicestat/



(a) Comparison between the theoretical and empirical false alarm rate (FAR) α_0.



(b) Comparison between theoretical and empirical distribution of normalized residuals $\mathbf{H}_{t,v}^{\perp}\mathbf{\Sigma}^{-1/2}\mathbf{x}_{t,v}$.



(c) Activity metrics, CPU and TX, of Botcloud alone as a function of attack payload.

Fig. 3: Validation of assumptions: (a) sharpness of statistical analysis providing expression for false-alarm rate (FAR); (b) fidelity of the Gaussian statistical model used in the paper: (c) relevance of the linear model impact of attack payload on system activity.

fifteen minutes. This gave us a set of 10 metrics among which some are highly correlated, especially sending and receiving rates averaged over various durations. It is also worth noting that for all the results provided in the present section, we have estimated the two main principal components and that we kept, for each container, only the component associated with the largest absolute weight. The reasons are that with only 10 metrics, the number of principal components should be kept low and that this setting, aiming at reducing the complexity, already provides overall good results.

Before considering this dataset for our evaluation purposes, we have first pre-processed it to especially remove all inconsistent data. These are: (1) *PlanetLab* servers that are not always responding or even getting disconnected, and (2) containers with an absence of activities over a capture period. Also, since *PlanetLab* does not provide a balanced scheduling algorithm of virtual hosts on servers, we have removed (3) tenants with fewer than 10 containers and, (4) servers with fewer than 10 containers, in order to make our dataset better fit a realistic cloud infrastructure. In total, we ended up with about 89 million system activity traces which are those we consider in the following.

2) Featuring the Botcloud Activity: Regarding the malicious activity, we considered two different botnets deployed in isolated containers that belong to us, one per server. These bots are (1) Hybrid_v1.0[5], programed in PERL, a high-level language, and communicating via the HTTP protocol, and (2) Kaiten[6], programed in C, a low-level language, and using the IRC protocol for communications. The 15 experiments we carried out cover both UDP Flood attacks as well as SYN Flood TCP attacks both with corresponding attack payload from 1 MB/s to 10 MB/s per container.

Eventually, it is important to note that the Botclouds reference activity, referred to as $\overline{\mathbf{a}}$ in Section V, has been computed by simply averaging the whole measurements of containers whose activity corresponds to the Botcloud activity alone.

3) Generating Infected Container Activities: For all numerical experiments, we did not use the data collected from the containers that only runs the Botcloud. Instead, to meet the usual attack scenario in which legitimate users have the containers corrupted, we have "artificially" created infected containers by merely adding a randomly selected container that performs a legitimate activity with another randomly selected one that only runs the Botcloud. All the experiments have been repeated 10 times to limit the effect due to randomness of the chosen containers to be corrupted.

Hence, it is acknowledged that the computed Botclouds reference activity is not completely independent from the additional activity it is aimed at detecting. However, we mention that the detection is carried out on workload estimation residuals, which are quite different from the Botcloud activity alone. Besides, note that the Botcloud reference activity is an average of more than 2 million corrupted activity traces that have been collected; the detection system thus does not aim at detecting a specifically known pattern.

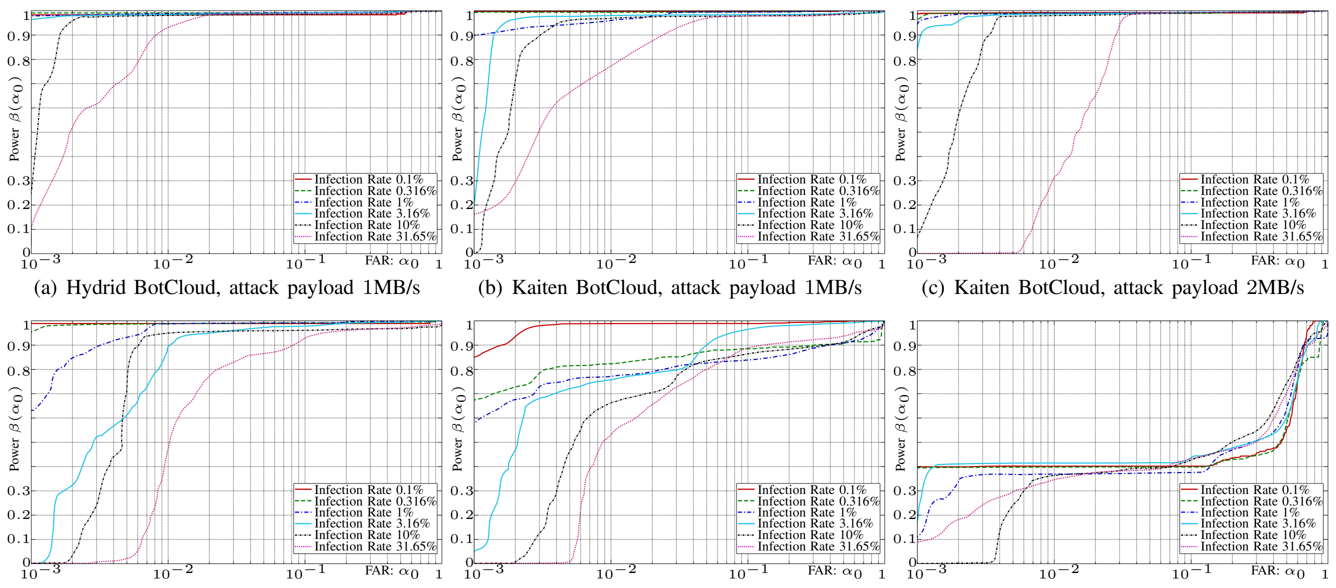
B. Verification of the Proposed Method and Models

1) Workload Estimation: The first part of our evaluation consists in verifying the accuracy of the assumptions our model relies on. The first assumption made concerns the decentralized workload estimation where the main types of behaviors are almost exclusively represented within the first few principal components. This assumption allows the design of a rather sparse linear parametric model, with few basis vectors. Figure 2 exhibits empirical results which verify this assumption. In particular, Figure 2(a) shows the averaged eigenvalues which stand for the weight of associated components. Here, by *averaged*, it is meant that all the activities from all the measurements have been analyzed at once. This figure clearly emphasizes the quickly decreasing values of eigenvalues. This demonstrates that only a few numbers, typically four at most, feature the axes along which all the data are distributed. The other axes represent either noise or abnormal events. Then, Figures 2(b) and 2(c) show the capability of the decentralized estimation approach to precisely compute the principal components. First, Figure 2(b) focuses on the accuracy of the estimation by showing, according to the number of iterations of the decentralized estimation process, the averaged angle between the first two estimated principal components and the real components computed in a centralized manner. This figure shows that a small number of iterations allows an accurate estimation. Interestingly, it also shows that the first principal component is estimated faster than the second. This phenomenon is due to the sequential nature of the estimation process where each component is estimated according to the most important ones. Consequently, it is necessary to estimate the first principal component prior being able to estimate the second one, hence explaining the slower convergence of the latter. Eventually, Figure 2(c) shows the distribution of the distance (in semi-log scale) between normalized metrics of the estimated and the real activity. By exhibiting most of the values in a short distance (typically 1), this last figure confirms the actual accuracy of the proposed decentralized approach for estimating principal components of containers' activities and more generally this confirms the efficiency of the data-driven approach for designing a linear model.

2) Theoretical Model of the Detection Process: We have then verified the validity of the assumptions on which the proposed detection method is based. For such validation, first Figure 3(a) presents a comparison between the theoretical and empirical false alarm rates obtained by executing our detection algorithm on both the real and simulated datasets. This result is fundamental to establish the relevance of the design of our detection method and it enables the mastering of various parameters such as the false alarm rate, detection accuracy, reactivity or detection delay and attack payload, as presented in Section V. The excellent fitting of the theoretical and empirical false alarm rate, up to probability smaller than 2.10^{-3}, as presented in Figure 3(a), emphasizes the sharpness of the theoretical results and hence, the relevance of the method. This is one of the main strengths of the present approach.

Similarly, Figure 3(b) presents a comparison between the empirical distribution of normalized system activity metrics after rejection of legitimate activity, that is $\mathbf{H}_{t,v}^{\perp}\mathbf{\Sigma}^{-1/2}\mathbf{x}_{t,v}$, and the Gaussian model. Once again, one can observe, for all the presented metrics, an excellent match between theoretical and empirical probability distribution functions, denoting the relevance of the statistical model and that of the linear rejection

[5]http://security-sh3ll.blogspot.fr
[6]www.hackforums.net/showthread.php?tid=974543



(a) Hydrid BotCloud, attack payload 1MB/s (b) Kaiten BotCloud, attack payload 1MB/s (c) Kaiten BotCloud, attack payload 2MB/s

(d) Kaiten BotCloud, attack payload 5MB/s (e) Kaiten BotCloud, attack payload 7MB/s (f) Kaiten BotCloud, attack payload 10MB/s

Fig. 4: Performance of the proposed detector over the real dataset for various attack payloads (from 1MBps up to 10MBps) and for various infection rates, that is, the ratio of infected virtual hosts.

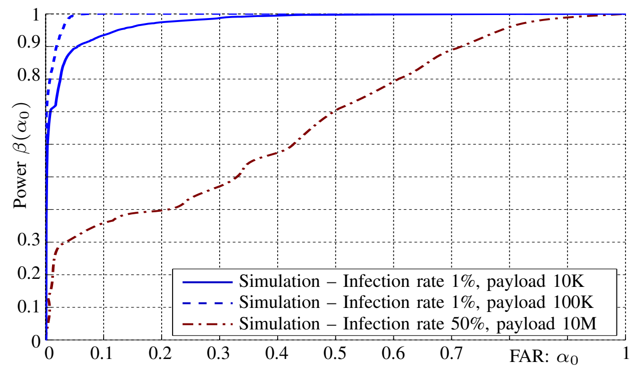


Fig. 5: Empirical ROC curves, detection accuracy or power function $\beta(p, \alpha)$, as a function of false alarm rate, for a few attacks settings

method for removing legitimate activities.

Finally, Figure 3(c) shows the relation between the attack payload and the Botcloud activity. To this end, it presents with red and blue crosses the CPU usage and the transmission rate (TX) as a function of the attack payload. One would expect that TX would be close to the attack payload. Interestingly, TX rate is, on average, about 0.7 smaller than the attack rate, and one can note the very large deviation of TX for the same attack rates. Similarly, this figure also highlights the excellent correlation between the CPU usage and the attack payload. This confirms the relevance of the linear approximation of the Botcloud activity as a function of the payload.

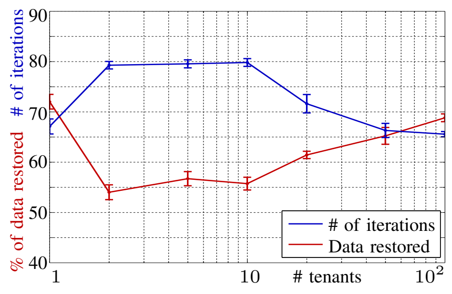
C. *Detection Accuracy on Real and Simulated Datasets*

This subsection presents a performance evaluation of the proposed method for a wide range of settings. First we investigate the efficiency over the *PlanetLab* dataset that features all the properties of real containers activities. In this empirical evaluation, we consider each harvesting campaign data separately. Botcloud activities, that are initially performed in 40 dedicated containers, are incrementally mixed with randomly chosen legitimate containers, thus varying the infection rate.

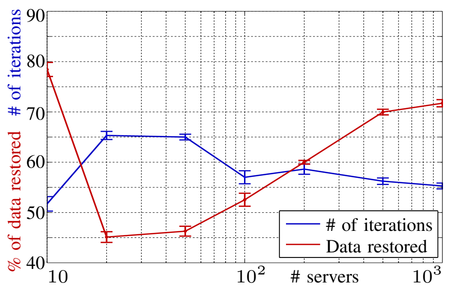
Figure 4 shows the efficiency of the detection method under different infection rates, attack payloads and for the two considered Botclouds. The detection accuracy is presented as ROC (Receiving Operational Characterics) that present the power function β as a function of the false alarm probability α_0. Figures 4(a) and 4(b) compare the two different implementations of the aforementioned Botcloud, both for an attack payload of 1MB/s per infected host. They show that the two Botclouds, while having different implementation features, can be detected with almost the same accuracy, that is nearly perfect.

The other figures present the detection accuracy for the proposed detection method for various attack payloads. We empirically observe that for all payloads, the efficiency of the method only slightly changes from a Botcloud to another. Consequently, the results obtained with Kaiten are solely presented. Interestingly, one can note that while the attack payload increases, the detection accuracy decreases, which seems counterintuitive. Indeed, Figure 4(c) shows that when the attack payload increases to 2MB/s, the detection accuracy remains almost perfect. However the decrease in detection accuracy becomes more obvious when the attack payload reaches 5MB/s, as depicted in Figure 4(d). In this case indeed, the highest number of infected hosts tends to lower the detection performance. The same effect becomes more perceptible for the results presented in Figure 4(e) for an attack payload of 7MB/s and it makes the detection accuracy rather poor at a payload of 10MB/s as shown in Figure 4(f).

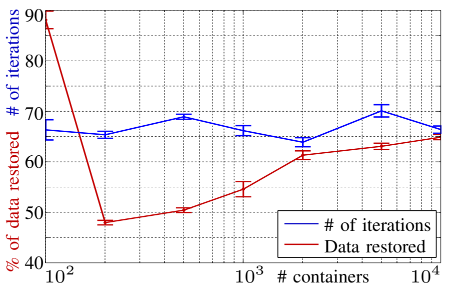
These results emphasize the limit of our approach which focuses on low rate attacks as referred to as the most difficult to detect. In the framework of the proposed methodology, the linear model, used to represent the legitimate data and provided in Section V and Equations (13)-(14) especially, is



(a) Evolution of iteration numbers and model accuracy with respect to the number of tenants

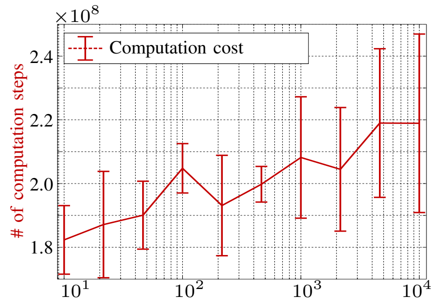


(b) Evolution of iteration numbers and model accuracy with respect to the number of servers

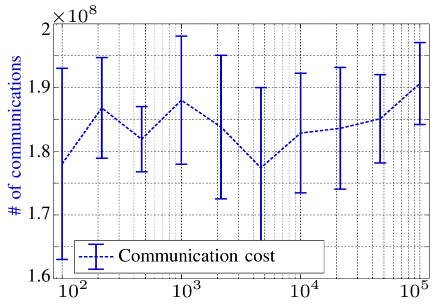


(c) Evolution of iteration numbers and model accuracy with respect to the number of containers

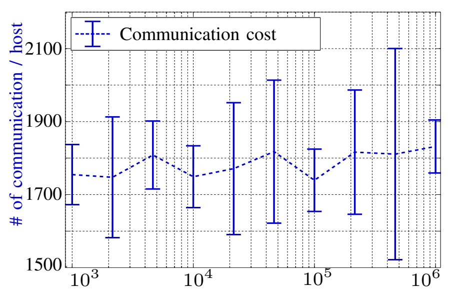
Fig. 6: Principal components estimation method efficiency, in terms of the number of iterations, and workload estimation accuracy according to the number of (a) tenants, (b) servers, and (c) containers.



(a) Computation cost, measured as the number of computation steps, with respect to the number of tenants



(b) Communication cost, measured as the number of messages, with respect to the number of servers



(c) Communication cost per virtual host with respect to the total number of containers

Fig. 7: Scalability of the method in terms of computation and communication costs with respect to the number of (a) tenants, (b) servers, and (c) containers

one of the method originality since it uses the data themselves to build the estimation of the principal components for such a linear model. As aforementioned in Section V, this method works accurately as long as the malicious activity remains negligible within the overall cloud activity. The effect that can thus be observed in Figures 4(d)-(f) is that, as the attack payload increases, the principal component analysis tends to model the activity of the Botcloud itself. One can note that the estimation of the principal components is made by using an average of the last 100 samples, which explains a rather decent detection just after the attack starts, hence ROC curves with somewhat decent power for low false alarm probability. Besides, one interesting phenomenon is that, though real data might be corrupted with effects that are hardly interpretable, the number of infected hosts does not seem to affect the detection much. This is due to the averaging method for principal components estimation that prevents overfitting for specific behavior of a minority of containers.

To confirm and extend the results obtained on the real dataset, we leveraged simulated data for which the legitimate activity and the attack can be tuned. The methodology for generating realistic simulated data is described in detail in Appendix A. The results presented in Figure 5 confirm the high accuracy of the proposed method for low-rate attacks as well as the limits for the detection of high attack payload. Indeed, Figure 5 gathers two results for lower attack rates, namely at 10 KB/s and 100 KB/s with 1% of infected containers among the $100,000$ considered here. In this setting, the method proposed in this paper performs with the highest accuracy. We can also note the limit of the proposed detection method that is achieved around 10 KB/s at which the detection accuracy significantly decreases. At the opposite, when setting the attack payload at 10MB/s with 50% of the containers infected, we observe a performance of the detector very similar to that observed on real data.

D. *Scalability of the Proposed Method*

Eventually, we evaluated the scalability of our whole approach. Indeed, in the context of cloud computing the number of virtual hosts, tenants and, to a lesser extent, servers can be very large. It is thus crucial to evaluate to what extent the method can fit deployment at such large scales.

First, the results presented in Figure 6 aim at showing how the workload estimation performs with respect to such scaling factors. In particular, they are given as a function of the number of tenants in Figure 6(a), number of servers in Figure 6(b) and number of containers in Figure 6(c). The baseline setting consists in 10 tenants, 100 servers and 1000 containers, each of these parameters being changed at once. To evaluate the accuracy and efficiency of the principal components estimation, we consider two metrics which are (1) the number of iterations required to achieve the overall convergence of the principal components estimation over the gossip protocol, depicted with the blue curve, and (2) the amount of data that falls into the first largest principal components depicted with the red curve.

The results overall show that neither the number of servers, nor tenants, nor containers have an impact on the number of iterations required for principal components estimation, which clearly demonstrates the scalability support of our approach. As for the accuracy of the principal components estimation, when the number of servers, tenants or containers is too low, their estimation is difficult because individuals with different behaviors are scattered apart. However, the accuracy rises at a quick pace when increasing the number of servers, tenants and containers, and seems to stabilize around 70%, which stands for a very good result given the fact that only one, out the two main principal components, is used for each container.

Finally, Figure 7 shows the evolution of computational and communication costs, with respect to the same scaling factors. To consider a very large scale which is not reachable with the real dataset, we consider here the simulated one. Note that to reproduce realistic conditions, we have used a larger number of containers (between one thousand and one million) and a smaller number of servers (between ten and ten thousand) while the number of tenants ranges from hundred to hundreds of thousands.

Figures 7(a) and 7(b) respectively show the computation cost with respect to the number of tenants and the communication cost as a function of the number of servers. For readability, we have only plotted one of these metrics in each figure because the main computationally expensive process is the workload estimation; the detection only requires a single locale evaluation of an expression. In this estimation process, almost each computation is followed by a communication to exchange estimates with one random neighbor over the gossip protocol. The number of communications and the number of computations are thus similar. These figures show that (1) the number of tenants only affects the number of computations required to estimate the workload a little, with a linear increase from 1.8×10^8 to 2.2×10^8 when the number of tenants grows up to three orders of magnitude, and (2) the number of servers has almost no impact on the communication cost when the latter grows up to three orders of magnitude too. Figure 7(c) finally addresses the communication cost with respect to the number of containers. This metric highlights a linear relationship between the overall communication cost of our approach and their number. But, for a better readability, we plot in figure 7(c) the number of communications per container which thus remains almost constant, regardless of their number. Altogether, these results confirm the overall scalability of the proposed method with the independence, or at worst the linear growth, of its cost according to the different considered scaling factors.

VII. Conclusion and Future Work

The work presented in this paper consists in designing and validating a solution for the detection of malicious activities perpetrated by virtual hosts infected by botnets in a public cloud context. To that aim, we have proposed a two-step approach based on PCA which first consists of (1) estimating the current workload of virtual hosts to highlight outliers by rejecting the legitimate activity, thus acting as a behavioral approach, and (2) then to compare the footprint of residual activities to known signatures. By avoiding the computation of the covariance matrix and decentralizing the workload estimation, we have demonstrated the capability of our approach to support very large scale infrastructure counting thousands of servers, hundreds of thousands of tenants and millions of virtual hosts. We have especially shown that the estimation accuracy remains constant whatever the size of the infrastructure and that communication costs related to the decentralized estimation remain also acceptable. In a second part, we have proposed a detection approach based on the statistical hypothesis theory which enables the detection of malicious activities perpetrated in virtual hosts. The theoretical performance of the detector has been established analytically and the accuracy of the model, as compared to empirical scenarios, has been shown. We have especially demonstrated to what extent our approach allows an operator to master the false-alarm probability and tune other parameters accordingly. To validate our approach, we have considered a large-scale dataset providing real container traces as well as a simulated ones which enabled us to reach very large scales. For both of these datasets, we have shown the capability of our system to detect activities of botnets perpetrating DDoS attacks whatever the infection rate and attack payload are. Our approach especially provides high performance against low footprint activities. Finally, by only considering the sole metrics available at the virtualization level, we have preserved the privacy of tenant activities and avoided the placement of probes into the tenant's space.

Main directions for future work are twofold. The first one we are currently exploring, is the adaptation and evaluation of our approach against other kinds of malicious activities. We are especially bringing our attention to data leakage which exhibits a footprint that largely differs from that of a DDoS attack. As longer term future work, we plan to implement our solution in a real monitoring probe to actually evaluate the performance of our approach in a real infrastructure.

References

[1] H. Pedram *et al.*, "Botcloud – an emerging platform for cyber-attacks," Tech. Rep., 2012.

[2] K. Clark *et al.*, "Botclouds-the future of cloud-based botnets," in *in CLOSER*. Citeseer, 2011.

[3] "Security Engineering Research Team (SERT) Quarterly Threat Intelligence Report. Q4 2013," Tech. Rep., 2013.

[4] J. J. Santanna *et al.*, "Booters–An analysis of DDoS-as-a-service attacks," in *Integrated Network Management (IM) IFIP/IEEE Intl' Symposium on*, 2015, pp. 243–251.

[5] R. Ragan and O. Salazar, "Cloudbots: Harvesting crypto coins like a botnet farmer," *BlackHat USA*, 2014.

[6] J. Ruiter and M. Warnier, *Computers, Privacy and Data Protection : an Element of Choice*, ch. Privacy regulations for cloud computing: Compliance and implementation in theory and practice, pp. 361–376, Springer, 2011.

[7] N. Ghadban *et al.*, "A Decentralized Approach for Adaptive Workload Estimation in Virtualized Environments," in *IFIP/IEEE Intl' Symposium on Integrated Network Management (IM)*, 2017, pp. 1186–1194.

[8] B. Hammi *et al.*, "Understanding Botclouds from a system perspective: A principal component analysis," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, 2014, pp. 1–9.

[9] H. Badis *et al.*, "A collaborative approach for a source based detection of Botclouds," in *IFIP/IEEE Intl' Symposium on Integrated Network Management (IM)*, 2015, pp. 906–909.

[10] A. B. Downey and D. G. Feitelson, "The elusive goal of workload characterization," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 4, pp. 14–29, 1999.

[11] M. C. Calzarossa *et al.*, , "Workload characterization: A survey revisited," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 48:1–48:43, 2016.

[12] A. K. Mishra *et al.*, "Towards characterizing cloud backend workloads: Insights from google compute clusters," *SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 4, pp. 34–41, 2010.

[13] Y. Chen *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.

[14] D. Gmach *et al.*, "Workload analysis and demand prediction of enterprise data center applications," in *Proc. of Intl' Symposium on Workload Characterization*, Washington, DC, USA, IEEE, 2007, pp. 171–180.

[15] A. Quiroz *et al.*, "Towards autonomic workload provisioning for enterprise grids and clouds," in *IEEE/ACM Intl' Conf. on Grid Computing*, 2009, pp. 50–57.

[16] F. Wihub *et al.*, "A Gossip Protocol for Dynamic Resource Management in Large Cloud Environments," in *IEEE Trans. Network and Service Management*, vol. 9, no. 2, pp. 213–225, 2012.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[18] V. Cevher *et al.*, "Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 32–43, 2014.

[19] D. Feldman *et al.*, "Turning Big Data into Tiny Data: Constant-size Coresets for K-means, PCA and Projective Clustering," in *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Philadelphia, PA, USA, 2013, pp. 1434–1453.

[20] C. Alzate and J. A. K. Suykens, "Multiway Spectral Clustering with Out-of-Sample Extensions through Weighted Kernel PCA." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 335–347, 2010.

[21] J. R. Bunch and C. P. Nielsen, "Updating the singular value decomposition," *Numerische Mathematik*, vol. 31, pp. 111–129, 1978.

[22] P. M. Hall *et al.*, "Merging and Splitting Eigenspace Models." *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 9, pp. 1042–1049, 2000.

[23] H. Kargupta *et al.*, "Collective principal component analysis from distributed, heterogeneous data," in *Eur. Conf. on Principles of Data Mining and Knowledge Discovery*. Springer, 2000, pp. 452–457.

[24] Y. Le Borgne *et al.*, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[25] S. B. Korada *et al.*, "Gossip PCA," in *Proc. of the ACM SIGMETRICS*, New York, NY, USA, ACM, 2011, pp. 209–220.

[26] S. V. Macua *et al.*, "Consensus-based distributed principal component analysis in wireless sensor networks," in *IEEE Intl' Workshop on Signal Processing Advances in Wireless Communications*, 2010, pp. 1–5.

[27] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed {PCA}," *Signal Processing*, vol. 104, pp. 120 – 135, 2014.

[28] J. Weng *et al.*, "Candid Covariance-Free Incremental Principal Component Analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 1034–1040, 2003.

[29] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, 2004.

[30] A. Chonka *et al.*, "Cloud security defence to protect cloud computing against http-dos and xml-dos attacks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1097 – 1107, 2011.

[31] C. V. Zhou *et al.*, "A survey of coordinated attacks and collaborative intrusion detection," *Comp. & Security*, vol. 29, no. 1, pp. 124 – 140, 2010.

[32] J. François *et al.*, "FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1828–1841, 2012.

[33] Y. Xiang *et al.*, "Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics," *IEEE Trans on Information Forensics and Security*, vol. 6, no. 2, pp. 426–437, 2011.

[34] Y. Chen *et al.*, "Collaborative Detection of DDoS Attacks over Multiple Network Domains," *IEEE Trans on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649–1662, 2007.

[35] C. Modi *et al.*, "A survey of intrusion detection techniques in cloud," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42–57, 2013.

[36] N. TulasiRam, *et al.*, "An Extrusion Detection System against Bot-Clouds," pp. 207–215, 2013.

[37] V. R. Kebande and H. S. Venter, "A cognitive approach for botnet detection using artificial immune system in the cloud," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, IEEE, 2014, pp. 52–57.

[38] M. Memarian *et al.*, "EyeCloud: A BotCloud Detection System," in *Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 1067–1072.

[39] B. Li *et al.*, "*You Can't Hide: A Novel Methodology to Defend DDoS Attack Based on Botcloud*, Proc Intl' Conf. on Applications and Techniques in Information Security. Springer, 2015, pp. 203–214.

[40] M. Bazm *et al.*, "Malicious virtual machines detection through a clustering approach," in *Cloud Technologies and Applications (CloudTech)*, IEEE, 2015, pp. 1–8.

[41] P. Garça-Teodoro *et al.*, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1–2, pp. 18 – 28, 2009.

[42] M. Jelasity *et al.*, "Gossip-based Aggregation in Large Dynamic Networks," *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, 2005.

[43] S. Boyd *et al.*, "Randomized gossip algorithms," *Information Theory, IEEE Trans on*, vol. 52, no. 6, pp. 2508–2530, 2006.

[44] C. Rao, *Linear statistical inference and its applications (second edition)*. John Wiley & Sons, 1973.

[45] A. Karasaridis *et al.*, "Wide-scale botnet detection and characterization," Usenix *HotBots* vol. 7, pp. 55–62, 2007.

[46] E. Lehmann and J. Romano, *Testing Statistical Hypotheses, Second Edition*, 3rd ed. Springer, 2005.

[47] M. Fouladirad *et al.*, "Optimal fault detection with nuisance parameters and a general covariance matrix," *Intl' Journal of Adaptive Control and Signal Processing*, vol. 22, no. 5, pp. 431–439, 2008.

[48] L. Scharf and B. Friedlander, "Matched subspace detectors," *Signal Processing, IEEE Trans on*, vol. 42, no. 8, pp. 2146 –2157, 1994.

[49] B. K. Guépié, "Sequential detection of transient changes: application to water distribution network monitoring," (in french) Ph.D. dissertation, Troyes University of Technology (UTT), March 2013.

[50] R. Cogranne, "A sequential method for online steganalysis," in *Information Forensics and Security (WIFS)*, IEEE, 2015, pp. 1–6.

[51] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[52] J. H. Ward Jr., "Hierarchical grouping to optimize an objective function," *Journal of the Am. Stat. Association*, vol. 58, no. 301, pp. 236–244, 1963.

Appendix A
Methodology for Realistic Data Generation

In order to evaluate the performance and cost of our detection approach against very large infrastructure, we have generated realistic synthesis data. To that aim, the methodology we considered consists in the four following steps (1) activity generation (2) behavior generation (3) association to a tenant and (4) association to a server.

A. Activity Generation

The first step of realistic data generation consists in assigning an activity to virtual hosts. To this end, it is proposed to characterize the joint distribution of projections of all the real datasets onto the first-main axes given by the PCA. The interest in using the principal components here is to be able to restore almost all the diversity observed in the real dataset while using only a limited number of variables, here typically six variables for the projection on the first six components instead of the 10 captured metrics. It is also worth noting that the joint distribution of data projections on main components allows us to take into account the conditional distribution of each component, thus intrinsically reproducing the correlation between the various captured metrics and leading to realistic activities.

The empirical joint distribution is captured using 21 points sampled to represent the boundaries of 20 quantities, for each projection individually. It is important to note that it is hardly possible to capture more precise joint distribution of data because 20 quantities for six variables represent a joint distribution characterized by 64 million data (20^6).

Then, using the empirical Cumulative Distribution Function (CDF) F it is proposed to adapt the inverse transform sampling to generate data drawn from the same distribution. The principle of inverse transform sampling is to generate a random variance uniformly distributed in the range $[0; 1]$ and to transform it using an inverse distribution function F^{-1} to ensure that the results have the desired distribution characterized by its CDF, the inverse function of F^{-1}. In the case of PCA projection, we used six independent random variables $x_1, \ldots, x_6$ that are distributed uniformly in $[0; 1]$. The first one is transformed using the inverse distribution of the first projection individually. Once the value of the first projection is set, one can measure the empirical distribution of the second PCA conditionally to the value of the first one. We apply the inverse transform sampling on this conditional distribution, and apply the same method on third projection knowing the values of the two firsts. The process is repeated until the six projections have been drawn. In order to get a higher accuracy, in this process it is proposed to make a linear interpolation between the two quantities values between which the random values x_n fall. This process ensures that when the end is reached the generated random variable follows the same joint distribution of PCA projection that mimics the real dataset.

At the end of this step, an activity is generated for each virtual host, representing the mean activity of the latter.

B. Behavior Generation

The second step to simulate realistic data consists in addressing the behavior over time of the virtual hosts. To this end, a similar approach is used, where the empirical joint distribution of the first six principal components is computed. The main difference here is that the components are computed from features extracted to characterize the behavior of virtual hosts over time. From this empirical joint distribution, we use the same method adapted from inverse transform sampling to generate data that follow the same distribution of features from the original dataset. This set of features characterize the activity of each tenant. In order to obtain behavior over time from this set of features we generated data with the same number of edges, following a normal distribution, with the same expectation and variance between abrupt changes and the same expectation and variance of change values. Note that this process is iterative in the sense that each change is added one by one, ordered by occurring time. The difficulty here is to ensure that the activity when evolving remains close to the mean activity samples from the first step. To this end, at each step we adjust the sign of the change choosing the one that minimizes the distance with the activity obtained from the first step.

C. Association to tenants and servers

Finally, the two remaining steps are the association of each virtual host to a tenant and to a server. To ensure that we associate to the same tenant data which are close to each other, we used a hierarchical cluster analysis method [51] based on Ward's distance [52] to cluster the data into exactly the number of groups that equals the number of desired tenants. In fact, such a clustering method is here relevant as it is aimed at gathering virtual hosts with similar activities, similar behaviors taking into account the volume of activity. The clustering method is applied on both PCA projection obtained to characterize the activity and those obtained to feature the behavior over time. This allows us to guarantee that all virtual hosts belonging to the same tenant has similar activities and similar behaviors.

The last step which consists in associating each virtual host to a server is made iteratively. For each new tenant, we measure the mean activity of all servers. Note that the activity of all servers is normalized, for each metric by the empirical mean observed on virtual hosts over the real dataset in order to scale each metric so that they can all be compared to each other. It is then tried to associate a new virtual host to each server and the Euclidean distance between the activity of this server and the mean activity of all the servers is measured. Once all those distances have been measured, the new virtual host is associated to the server for which adding this virtual host onto this server minimizes the distance between the activity of this server with the mean activity of servers. This simple process is used here to ensure that at the end, all the servers have similar overall activities and guarantees that a server is not given too much CPU consumption or receiving bandwidth for instance.

Rémi Cogranne holds the position of Associate Professor at Troyes University of Technology (UTT), France, since 2013. He had received his PhD in Systems Safety and Optimization in 2011 and his engineering degree in computer science and telecommunication in 2008 both from UTT. He has been a visiting scholar at Binghamton University for year between 2014 and 2017. During his studies, he took a semester off to teach in a primary school in Ziguinchor, Senegal and studied one semester at Jönköping University, Sweden. Since 2011, his work has been generously supported by various industrial and governmental contracts that lead to more than 55 papers and 3 International patents. His main research interests are in hypothesis testing with applications for image forenrics, steganalysis and steganography and for computer network anomaly detection.

Guillaume Doyen holds the position of Associate Professor at Troyes University of Technology (UTT), France, since 2006. He is affiliated with Charles Delaunay institute (ICD - UMR CNRS 6281) where is is the co-chair of the Cyber-Security transversal research program. His current research interest focuses on the design of autonomous management and control solutions applied to the performance and security of content distribution and cloud computing. He has published more than 40 papers in the network and service management community. He is an active member of high-venue conferences TPC (e.g. IEEE/IFIP CNSM, IEEE/IFIP IM and NOMS, IFIP AIMS), a regular reviewer for the top-ranked related journals (e.g. IEEE CommMag, IEEE TNSM, Springer JNSM, Wiley IJNM) and has been a co-chair of several events (e.g. ManSDN/NFV, IFIP AIMS). He has been involved in several research projects (e.g. ANR-Doctor, IA-Request, ANR BBNet).

Nisrine Ghadban received the Dipl.-Ing. degree in electrical engineering in 2012 from the Faculty of Engineering, the Lebanese University, Lebanon, and the M.Sc. degree in industrial control in 2012, from the Faculty of Engineering, the Lebanese University, Lebanon and from the University of Technology of Compiègne, France. In 2015, she received the Ph.D. degree in systems optimization and security at the University of Technology of Troyes, France and was a Postdoctoral Research associate with the Autonomic Networking Environment team, from 2016 to 2017. She is currently a professor at the Lebanese University, faculty of engineering. Her current research interests include machine learning, and signal processing, anomaly detection, Big Data.

Badis Hammi is a Postdoctoral fellow in Institut Mines Telecom ParisTech in France. He received his Master’s degree in University of Valenciennes and Hainaut Cambresis (2011) and his PHD in Troyes University of Technology in (2015). His main research topics of interest are in Security and Intrusion Detection in Wireless Environments and in Cloud Computing.