P-Cop: A Cloud Administration Proxy to Enforce Bipartite Maintenance of PaaS Services

Bruno Braga, Nuno Santos
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa
Lisbon, Portugal
Email: brunobraga@ist.utl.pt, nuno.santos@inesc-id.pt

Abstract—**Platform-as-a-Service (PaaS) infrastructures are highly dependent on cloud administrators. Ill-configured software systems or compromising insider activity can result in serious data breaches for clients of PaaS services. A general security approach against untrusted administrators is to employ operating system hardening techniques to limit access privileges on cloud nodes. However, this approach is overly inflexible for PaaS services, since superuser privileges tend to be required to apply a security patch, change a firewall rule, etc. This paper presents P-Cop, a system aimed to provide secure PaaS maintenance while preserving administration flexibility. To that end, P-Cop implements a *bipartite maintenance model* in which cloud administrator privileges can be elevated to superuser on a given node, but no sensitive guest computations can be allocated to the node until the issued command sequence has been endorsed by an auditor, i.e., a third-party mutually trusted by cloud provider and clients. P-Cop relies on a trusted proxy which supervises all privileged commands issued by the cloud administrators. Our current P-Cop design targets Docker-containerized PaaS services and leverages TPM hardware to enable remote attestation by external clients.**

I. Introduction

The danger of cloud mismanagement is an overlooked security risk that Platform-as-a-Service (PaaS) customers are often faced with. To maintain the software systems of PaaS infrastructures, cloud administrators hold privileged access to cloud nodes, including to those where client computations take place and customer data is located. Although in the vast majority of cases cloud administrators use their privileges judiciously, such privileges can be misused by introducing security flaws that can potentially result in serious data breaches for customers.

To overcome such risks, a common practice is to segregate administration roles so as to reduce the number of individuals that know the root passwords of critical systems. Unfortunately, such individuals can still introduce critical security flaws or obtain unrestricted access to customer data. To prevent unauthorized access to customer data, some systems preclude cloud administrators from obtaining root privileges entirely. Systems such as CloudVisor [1], [2] or BrokulOS [3] consist of hardened hypervisor or operating system, respectively, which expose carefully crafted administration interfaces that allow for performing a large range of management tasks without compromising the security of customers' computations. Such systems are normally coupled with Trusted Platform Module (TPM) [4] hardware and cloud attestation services [5], [6] to allow for remote attestation of systems by the customers. However, enforcing a permanent reduction of administrator privileges is hardly tolerable in PaaS services because some critical maintenance tasks require root privileges, e.g., apply security patches, modify firewall rules, install software packages, etc.

This paper presents P-Cop, a system which aims to secure PaaS services against cloud mismanagement threats without precluding cloud administrators from acquiring superuser privileges whenever necessary. The key idea to achieving this property is to enforce a bipartite maintenance model between cloud administrators and an *auditor*, i.e., a third-party whose role is to validate software configurations and which is mutually trusted by both cloud provider and customers. Under this model, cloud administrators have limited privileges in the common case. Whenever superuser privileges are necessary, cloud administrators can request privilege elevation so that they can log into cloud nodes as root and execute arbitrary commands. In such cases, to ensure that the resulting state is trustworthy, these commands must first be endorsed by the auditor before being definitely committed. To prevent data breaches, no guest computations can be allocated to the node from the moment a cloud administrator obtains root privileges until an auditor endorses pending operations.

To implement a bipartite maintenance model, P-Cop introduces a cloud administration proxy to be deployed on the PaaS backend. This proxy mediates all superuser operations performed by cloud administrators and supports their respective endorsement by the auditor. To acquire privilege elevation on a given cloud node, a cloud administrator requests a *super-session* to the proxy. After ensuring that no customer data is left on the node, P-Cop lets the administrator log into the target cloud node by establishing a tunneled SSL connection. Over that connection the cloud administrator can execute arbitrary root commands (i.e., UID = 0) while the proxy keeps on a local log a record of the command history of the super-session. Auditors can later retrieve the log from the proxy in order to analyze and endorse the super-session commands if no security vulnerabilities have been introduced, or revert them otherwise.

The main contribution of this work is the design of P-Cop. Since the P-Cop proxy is installed on the cloud backend, the system must shield itself against potential attacks by untrusted administrators. P-Cop must keep track of the software configuration of each cloud node in a way that is capable of surviving cloud node reboots. Moreover, it is necessary to provide evidence to PaaS clients that the software of P-Cop proxy and cloud nodes has been endorsed by a trusted auditor, otherwise no security assurances can be given to PaaS clients. To address such challenges, P-Cop incorporates new security protocols leveraging TPM chips deployed on the cloud nodes.

Without loss of generality, we built P-Cop for Docker-containerized PaaS services. We implemented both the P-Cop system and a PaaS service prototype, and performed empirical evaluation of the system based on benchmarks. Results show that our system adds small performance overheads to the PaaS service. Due to P-Cop, a time lag exists since super-session operations are issued by cloud administrators and their changes are reflected onto the PaaS clusters. The extent of that lag depends mostly on how tightly coupled cloud administrators and auditors operate.

II. Background and Goals

Containerized Platform-as-a-Service (PaaS) provides hosting cloud services for customer applications within *containers*. Containers are supported on a given server by software frameworks such as Docker [7], which modify the OS so as to enforce proper isolation between containers and manage their lifecycle. In a containerized PaaS infrastructure, the *compute* nodes (also named minions) is the place where guest containers execute. The *repository* nodes store container base images. Base images include pre-configured software (e.g., apache) which form the basis for customized container images to be instantiated on the minions. The *monitor* nodes implement the logic that ties together the entire service: they process container deployment requests, allocate containers to minions, manage resources, etc.

In the context of this paper, we highlight four main stakeholders: (1) the *cloud provider* owns the cloud infrastructure, (2) *publishers* are the service customers and are responsible for deploying containerized applications onto the cloud infrastructure, (3) *users* access publisher applications instantiated inside guest containers as regular Web applications, (4) *cloud administrators* are responsible for maintaining the software systems of the PaaS infrastructure.

The goal of our work is to enable the design of PaaS services that can prevent access to guest containers by untrusted cloud administrators. In spite of such restrictions, our solution must allow for flexible administration of the cloud by allowing cloud administrators to obtain superuser privileges whenever necessary. Thus, we assume that cloud administrators are untrusted. A cloud administrator can remotely reboot or power-cycle any of the cloud nodes to run an arbitrary operating system. He can then either mount the local file system and access persistent hard disk state, or install an arbitrary software stack on the node. If the node boots to the software stack installed on the local disk, the cloud administrator is limited to accessing the node through the administration interface provided by that specific software stack, e.g., an SSH console or a Web interface. The privileges that the administrator can acquire through that interface depend on how the software stack is configured. For example, the operating system can be hardened so that the administrator cannot obtain superuser privileges [3]. An untrusted administrator can install network sniffers and similar software in order to eavesdrop the network traffic, modify, suppress, or inject packets. However, we exclude attacks that involve physical access to the hardware, attacks based on software exploits, and side-channel attacks.

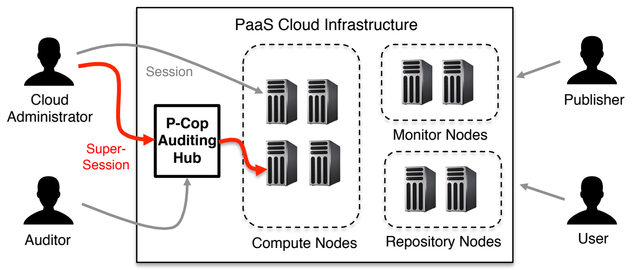

Figure 1. P-Cop architecture.

Every server installed in the cluster is equipped with a Trusted Platform Module (TPM) chip. Each TPM has a unique keypair Attestation Identity Key (AIK) whose private key is bound to each chip and the respective public key is certified by the cloud provider. This certification ascertains the property of that chip (and respective server) by the cloud provider. The certification of such keys is performed when the hardware is deployed within the cloud provider's premises. We require that the TPM chip and the cryptographic algorithms used in our solution are correct.

III. Architecture

We present P-Cop (PaaS Cop), a PaaS cloud management system that provides container isolation from cloud administrators while providing flexible maintenance capability. In our solution, rather than permanently restricting administration privileges of compute nodes, we allow such privileges to be temporarily elevated in order to let administrators log into a particular compute node and perform operations that require root access. Then, through a process of endorsement, such privileged operations must be validated by an *auditor* in order to ensure that no vulnerabilities have been introduced into the node software and therefore the node can be trusted to securely accommodate PaaS containers. Simply put, P-Cop enforces a bipartite maintenance policy between cloud administrators and auditors.

Figure 1 represents the architecture of the P-Cop system when deployed in a PaaS cloud backend. The central component of P-Cop is the *auditing hub*, which is a cloud administration proxy provided by independent servers. The auditing hub is responsible for keeping track of the software state of the compute nodes and managing super-sessions. Super-sessions consist of tunneled SSL connections over which a cloud administrator can access a compute node with root privileges. In regular sessions, the cloud administrator does not have root privileges. Another component of P-Cop is the *node guard*, which is a software agent that runs on each compute node and implements local actions upon requested by the auditing hub, namely attestation requests and local state transitions. Finally, P-Cop provides *client* software that allow stakeholders to interact with the system.

A. Restricted Operation Mode

Essentially, P-Cop ensures that PaaS guest containers can be allocated and execute only on compute nodes running a *trusted container runtime*. A trusted container runtime is a "containarized" software stack that satisfies container isolation properties. In particular, when such a software stack is installed on a compute node, cloud administrators cannot acquire superuser privileges (e.g., log into the root account). Instead, they are restricted to using a limited management interface that allows them to maintain the node without compromising the confidentiality or integrity of guest containers, e.g., collect logs, set up resource management policies, monitor the resource consumption, etc. This management interface can be provided by a remote user session over secure channel protocols like SSH, HTTPS, or similar.

To supervise which compute nodes of the PaaS infrastructure can accommodate guest containers, P-Cop maintains a list of *verified nodes*. Such nodes are assured to be installed with trusted container runtime software. This is achieved by requiring the software to be validated and signed by the auditor and then leveraging TPMs to attest that such software is running on the cloud nodes. P-Cop provides the means for publishers to instantiate their application on guest containers hosted by verified nodes and for users to process their data on such containers safely. P-Cop ensures that only the compute nodes that belong to this list eligible for hosting guest containers.

B. Privileged Operation Mode

In case a cloud administrator needs elevated privileges on a given verified node, P-Cop allows for opening a *super-session*. A super-session removes the restrictions imposed by trusted container runtime allowing the cloud administrator to log into the system with root privileges. However, to prevent security breaches, before establishing the session, P-Cop: 1) stops all running guest containers and wipes their content so as to avoid leaving forensic traces of user data and application code that could be accessed by the cloud administrator, and 2) removes the node from the verified list to avoid future instantiation of guest containers on compute nodes that are currently controlled by the cloud administrator and, therefore, are potentially insecure.

```
Nov 17, 2015 11:14:05 AM
ALL: Admin->Host:
service docker restart

Nov 17, 2015 11:14:06 AM
ALL: Host->Admin:
docker stop/waiting
docker start/running, process 6101
```

Figure 2. Example of a simple session log.

When the cloud administrator obtains access to a compute node through a super-session, he can perform arbitrary commands under root, e.g., install software, resize disk partitions, set up firewall rules, apply kernel patches, etc. Given that these operations can potentially leave the system in an insecure state (e.g., if a backdoor is left), the node cannot immediately be added back to the verified node list once the super-session ends. Instead, P-Cop adds the node to an *unverified node list* along with a log that provides a detailed account of all operations that were performed by the cloud administrator during the super-session. In particular, the log contains a record of all input commands provided by the cloud administrator and respective returned output (see Figure 2). To insert the node back into the verified node list, such operations must be properly endorsed.

C. Endorsement of Super-Sessions

Endorsement of super-session logs aims to ensure that the input commands issued by the cloud administrator have not introduced security breaches. This operation requires manual inspection of the logs by an *auditor*. The auditor is a trusted third-party that has special privileges on P-Cop for obtaining a list of super-session logs and approving or rejecting the operations performed within the super-session. If the super-session is approved, then the updated configuration of the compute node is deemed trusted. In this case, P-Cop removes the node from the unverified list and replaces it in the verified node list. Otherwise, if the super-session is rejected, a policy-based decision must be taken as to what to do next, which can be to revert the operations performed by the administrator, reinstall the software on the node, or other similar action. Such policy is defined by the auditor.

Essentially, the auditor is responsible for certifying the software of the PaaS infrastructure. In addition to the validation of super-session logs, the auditor must validate the software of the trusted container images and the software of the P-Cop system. Thus, in the interest of separation of privileges, the auditor role cannot be played by the cloud administrators. Instead, it must be assigned to a third party that is mutually trusted by the cloud provider, publishers, and users. The auditor may represent a collective entity that involve the participation of several stakeholders, e.g., cloud provider and customers.

IV. Design and Implementation

To design P-Cop, it was necessary to develop specific security protocols implemented between auditing hub, node guards, and P-Cop clients. Due to space limitations, the detailed description of the P-Cop protocols is provided in a companion technical report [8]. These protocols are necessary to overcome several challenges in system operation:

a. System initialization: Cloud administrators must first install a canonical software of the auditing hub. However, since cloud administrators are untrusted, P-Cop must include mechanisms to ensure the correct setup of the auditing hub and enable external remote attestation by the auditor.

b. Bootstrapping and attestation of compute nodes: After the auditing hub has been properly initialized, one of its main roles is to check the software configuration of the compute nodes and verify which of them have been properly set up with a trusted container runtime. Given that such nodes are also installed and configured by untrusted cloud administrators, P-Cop must include mechanisms to validate such nodes before being allowed to host guest containers.

c. Secure PaaS operations: A typical container-based PaaS service implements some core operations to support the life-cycle of guest applications. P-Cop must include defenses to secure such operations. Publishers must be assured that their applications are instantiated only on trusted compute nodes, i.e., compute nodes properly configured with a certified trusted container runtime. Secondly, users must be ensured that the applications they are accessing are authentic, can be properly identified, and execute on a trusted compute node.

d. Establishment and endorsement of super-sessions: In order for a cloud administrator to gain unrestricted access to compute nodes he must open a super-session. To provide super-session support while confining the execution of applications to trusted compute nodes, P-Cop must provide several guarantees. First, a super-session must only be granted to a cloud administrator once it has been assured that the targeted compute node is properly sanitized. Second, P-Cop must ensure that applications cannot be instantiated on a compute node of elevated administration privileges until the resulting configuration has not been endorsed by an auditor. Third, P-Cop must keep a complete record of super-session operations to ensure that the auditor can trace all configuration changes performed to the compute node. P-Cop includes mechanisms to address all these requirements.

We implemented the P-Cop system fully and a prototype of the PaaS software infrastructure. The P-Cop software components were developed using Java 8 with Java extensions for SSL. SSL credentials were generated using OpenSSL 1.0.1f and stored using Java keystores. Our trusted container runtime setup is based on Docker 1.9 running in a hardened Linux CentOS 7.1.1503 with kernel 3.10.0-229.4.2.el7.x86_64. Interactions with the TPM to generate quotes are performed using Trousers 0.3.7. SSL connections are maintained using OpenSSH 6.6.1 servers. All generated credentials use RSA-2048 bits keys. The PaaS software consists of a simple monitor which is responsible for handling requests from the publisher, administrator, and auditor, and for issuing requests to the auditing hub and to the compute nodes. PaaS repository is provided by the monitor. Minions receive and send application packages using Linux scp.

V. Evaluation

This section presents our evaluation of P-Cop in terms of performance and security.

A. Performance Evaluation

To evaluate the performance of our solution, we measure the time latencies of the operations that can most negatively affect the overall performance of the PaaS service: 1) attesting the infrastructure, which introduces delays in the system initialization that may affect the PaaS service uptime, 2) deploying applications, which can increase the time since the publishers deploy their applications on the PaaS service and the application becomes available to users, and 3) opening super-sessions, which introduce a delay until configuration changes are reflected into production.

Testbed. We evaluated P-Cop and our PaaS service prototype on a cluster that consists of: ten compute nodes, one monitor, and one auditing hub. All nodes have two Intel Xeon 3.00GHz, 2GB of RAM and an ethernet interface of 100Mbps. The base images run Linux CentOS 7.1.1503 with kernel 3.10.0-229.4.2.el7.x86_64. To simulate the network latencies experienced by the publishers, application deployment requests are issued from outside our cluster in a home network featuring a download rate of 55.7 Mbps and an upload rate of 5.6 Mbps. Requests by the clients of cloud administrators and auditors are issued within our cluster sharing a network bandwidth of 100Mbps.

Performance of attestation. Attestation is performed by auditors to the auditing hub and by the auditing hub to compute nodes. We use microbenchmarks to evaluate the performance of attestation. The measured time for both these operations is about 4 seconds. Nearly 24.6% of the overall attestation time is taken up by the quote operation of the TPM, while the remaining time is spent in cryptographic operations and network round-trip time of P-Cop protocol messages. Given that attestation operations have a relatively low duration and occur infrequently, they contribute with a residual impact in the initialization latency of PaaS services.

Performance of application deployment. To evaluate the deployment time of applications, i.e., the time since the publisher publishers an application and the application is instantiated on the cluster, we used two PHP applications of different sizes and complexities: *Hello* and *OSN*. Hello

Case	File transfers	Container setup	P-Cop operations	Total
Hello	00:04	07:15	00:10	07:40
OSN	00:17	08:05	00:18	08:21

Table I
DISCRIMINATION OF DEPLOYMENT LATENCIES FOR FIVE INSTANCES OF HELLO AND OSN (MINUTES:SECONDS)

is a simple PHP application that consists of a single static web page (1.5Kb) and is representative of the baseline deployment time for Docker-based application. OSN is an open source online social network application (11MB) which is representative of a complex and heavyweight application. Both applications run on official Apache Web server (version 2.4) container. We test each application changing the number of container instances on the cluster.

Table I shows the overall deployment times for five instances of Hello and OSN broken up into three main components: 1) file transfers between developer and monitor, and between monitor and minions, 2) container setup, and 3) P-Cop operations (SSL connections, exchanged messages, and P-Cop specific cryptographic operations). The results show that the main source of overhead comes from file transfers and application deployments, i.e., SSL file copy and Docker daemon, leaving less than 25 seconds for P-Cop operations, which represent 5% and 2% of the total deployment time for Hello and OSN, respectively.

Performance of super-sessions. For space constraints, we focus on the super-session opening time, which can be quantitatively measured. To measure this value, which determines how much time a cloud administrator needs to wait until he can log into a minion with superuser privileges, we deployed multiple instances of a test application on a given minion and initiated a super-session. Before the super-session can be established, the node is first sanitized and the applications are migrated to another node. We repeated the same experiment for both Hello and OSN applications.

Figure 3 shows the measured super-session opening time. Similarly to application deployment, the time grows linearly with the number of instances on the node. The time difference between OSN and Hello is smaller than the respective deployment times (in the order of seconds) because no file transfers need to be performed between the publisher and the monitor. Table II presents the contributions of several sources to the opening time of a super-session to a minion running five application instances. Four components are indicated: 1) file transfers between monitor and minions, 2) redeployment of applications on alternative minions, 3) removal of application files and containers from current minion, and 4) P-Cop operations. We can see that the most significant fraction of the measured times where spent by Docker in redeploying and deleting containers, namely 98% and 93% for Hello and OSN, respectively. Most of the time taken by the purging process comes from the Docker daemon.

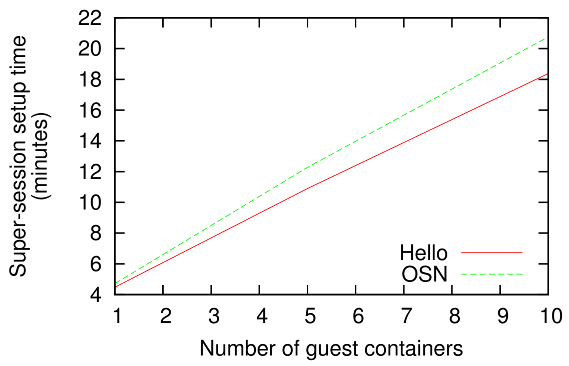

Figure 3. Administrtive latency for n instances of Hello and OSN running

B. Security Analysis

P-Cop provides confidentiality and integrity protection of guest containers for PaaS services. P-Cop protects publisher applications and user data from potentially malicious cloud administrators of the cloud infrastructure. Such protection is attained by leveraging TPM attestation, cryptographic techniques, and operating system access control mechanisms in the design of P-Cop.

However, P-Cop relies on a set of assumptions which, if violated, may result in security breaches. P-Cop does not protect against software exploits in the minions or in the software of the auditing hub. As a result, the auditor plays a critical role in order to rapidly identify faulty software and require well tested or even certified software. If the applications themselves are malicious, the users cannot also benefit from the protections offered by P-Cop: a malicious application with direct access to users' data can easily tamper with it or exfiltrate it. It is, therefore, essential for users to verify the identity and reputation of publishers before using their applications. In P-Cop, the auditors are also a crucial pillar in the system's root of trust. Ill-intended or negligible auditors can seriously compromise the security assurances offered by P-Cop. To mitigate this risk, the auditor role must be performed by a collective entity in which the participation of independent individuals is required to avoid collusion. P-Cop cannot offer protection against an adversary with physical access to the hardware, and in particular that can subvert the TPM chip. We rely on out-of-band mechanisms to guarantee the integrity of the cloud hardware.

VI. RELATED WORK

The line of research that is mostly related with P-Cop involves using trusted computing techniques to enhance security and trust in the cloud. This general approach, which was introduced by Santos et al. [9], has led to the development of cloud attestation systems such as CloudVerifier [6] and Excalibur [5] which allow for cloud customers to remotely attest the a cloud service based on TPM chips deployed on the cloud nodes. When coupled with hardened hypervisors [1], [10], cloud attestation systems enable

Applications	File transfers	Application deployment	Minion purge	P-Cop operations	Total
Hello	0.085 seconds	05:00	05:54	00:13	11:07
OSN	0.23 seconds	05:45	05:45	00:14	11:44

Table II
DISCRIMINATION OF MANAGEMENT REQUESTS' LATENCIES FOR 5 APPLICATIONS OF HELLO AND OSN (MINUTES:SECONDS)

customers to outsource computations into the cloud while ensuring computation isolation from cloud administrators. However, these systems are mostly focused on IaaS clouds.

Targeting exclusively PaaS services, Brown et al. [11] propose a trusted PaaS platform to address the lack of transparency by application end-users users. In contrast to P-Cop, however, their concern is about publishers that may deploy malicious or buggy applications onto the cloud, and not about untrusted cloud administrators. Thus, P-Cop provides complementary security assurances to this system.

Another related topic is about ways to improve the security of cloud administration. Butt et al. [12] propose a self-service cloud computing platform in which customers themselves can specify low-level configurations for the cloud infrastructure, but require a virtualized IaaS infrastructure. Secure Cloud Maintenance [13] supports different levels of administrator privileges on the compute nodes, but depend on a fully trusted administrator to configure an underlying SELinux OS; P-Cop overcomes this restriction. Hardened operating systems such as BroKulOS [3] implement fine-grained privilege separation for Linux administrators and can be used for building trusted container runtimes for P-Cop.

P-Cop's auditing capability is also related with secure logging systems. Systems like H-One [14] are based on a hardened hypervisor to log administrator operations using information flow tracking when managing guest VMs. Sinha et al. [15] do not rely on hypervisors, but rather leverage TPM for protecting the integrity of logs. Such techniques are orthogonal to P-Cop and can be used to improve the security of P-Cop's auditing hub.

VII. CONCLUSION

This paper presents P-Cop, a Platform-as-a-Service (PaaS) cloud management system that provides container isolation from cloud administrators while providing flexible maintenance capability. P-Cop prevents untrusted cloud administrators from accessing guest containers and therefore cause security breaches. To preserve the maintenance flexibility, P-Cop keeps access privileges restricted on cloud nodes, but allow cloud administrators to elevate their privileges on cloud nodes. To preserve the security, P-Cop provides mechanisms that allow a third-party auditor to verify that the maintenance operations are safe. Although P-Cop was targeted toward Docker-containerized PaaS, it can generally be applied to other PaaS services.

Acknowledgments: This work was partially supported by the EC through project H2020-653884 (SafeCloud), and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

REFERENCES

[1] F. Zhang, J. Chen, H. Chen, and B. Zang, "Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization," in *Proc. of SOSP*, 2011.

[2] D. G. Murray, G. Milos, and S. Hand, "Improving Xen security through disaggregation," in *VEE*, 2008.

[3] N. Santos, R. Rodrigues, and B. Ford, "Enhancing the OS against Security Threats in System Administration," in *Proc. of Middleware*, 2012.

[4] T. C. Group, "TPM Main Specification Level 2 Version 1.2, Revision 130," 2006.

[5] N. Santos, R. Rodrigues, K. P. Gummadi, and S. Saroiu, "Policy-sealed Data: A New Abstraction for Building Trusted Cloud Services," in *Proc. of USENIX Security*, 2012.

[6] J. Schiffman, T. Moyer, H. Vijayakumar, T. Jaeger, and P. McDaniel, "Seeding clouds with trust anchors," in *Proc. of WCCS*, 2010.

[7] "Docker," https://www.docker.com.

[8] B. Braga and N. Santos, "P-Cop: A Cloud Administration Proxy to Enforce Bipartite Maintenance of PaaS Services," INESC-ID, Tech. Rep. 3, May 2016.

[9] N. Santos, K. P. Gummadi, and R. Rodrigues, "Towards trusted cloud computing," in *Proc. of HotCloud*, 2009.

[10] J. M. McCune, Y. Li, N. Qu, Z. Zhou, A. Datta, V. D. Gligor, and A. Perrig, "TrustVisor: Efficient TCB Reduction and Attestation," in *Proc. of IEEE S&P*, 2010.

[11] A. Brown and J. S. Chase, "Trusted Platform-as-a-Service: A Foundation for Trustworthy Cloud-hosted Applications," in *Proc. of CCSW*, 2011.

[12] S. Butt, H. A. Lagar-Cavilla, A. Srivastava, and V. Ganapathy, "Self-service Cloud Computing," in *Proc. of CCS*, 2012.

[13] S. Bleikertz, A. Kurmus, Z. a. Nagy, and M. Schunter, "Secure Cloud Maintenance," in *Proc. of ASIACCS*, 2012.

[14] A. Ganjali and D. Lie, "Auditing cloud administrators using information flow tracking," in *Proc. of CCS*, 2012.

[15] P. England, L. Jia, J. Lorch, and A. Sinha, "Continuous Tamper-proof Logging using TPM2.0," in *Proc. of TRUST*, 2014.