Secure location-aware VM deployment on the edge through OpenStack and ARM TrustZone

Teodora Sechkova, Enrico Barberis, Michele Paolino
Virtual Open Systems
Grenoble, France
Email: {teodora, e.barberis, m.paolino}@virtualopensystems.com

***Abstract*—In recent years, there is an ongoing computational shift from the data center to the network edge. Due to the increased hardware capabilities of devices, the edge can also benefit from the dynamic and scalable services provided by the virtualization technologies. In turn, the edge computing brings low-latency and reduced network traffic, location-awareness and local caching. However, the new capabilities unlock new challenges in terms of security, data and workload location.**

In this work, we focus on the threats caused by the heterogeneous and distributed nature of the edge infrastructure. We build a trusted edge based on the hardware isolation of ARM TrustZone. Moreover, we use it as a secure foundation to perform location-aware virtual machine deployment utilizing the dispersed nature of the infrastructure. We measure the performance of our solution and discuss the overall overhead and potential improvements.

***Index Terms*—security, virtualization, cloud, edge computing, geo-fencing, asset tag, Trusted Execution Environment, TEE, OP-TEE, ARM TrustZone, VIM, Virtualized Infrastructure Manager, OpenStack**

I. Introduction

Nowadays, cloud computing is an established paradigm adopted by the business, the industry and the mass users alike. By the means of the virtualization technologies, the cloud provides dynamic and scalable compute, network and storage resources in the form of different services. In the meantime, the processing capabilities of the end devices are continuously growing, hence the ongoing shift of the computation from the data center to the network edge equipment. The utilization of the computing power of edge devices is a common subject of fog, edge and multi-access edge computing [1]. In particular, the distributed device locations closer to the users allow for low-latency communication, reduced network traffic, location-awareness and local caching. When such advantages are combined with virtualization technologies, edge computing becomes a natural part of concepts like Software-Defined Networking (SDN), Network Functions Virtualization (NFV) and Fifth Generation (5G) mobile networks.

As with any other new technology, edge computing brings new challenges together with the new opportunities. Undoubtedly, security is among the most important ones, though often neglected. Various surveys exist, analyzing cloud and edge security threats in different aspects and contexts [2]–[5]. The dispersed edge infrastructure often relies on wireless connectivity which makes it vulnerable to man-in-the-middle attacks. Remote or difficult to secure locations also increase the risk of tampering or replacement, therefore, extra measures need to be taken to verify the authenticity and integrity of the edge devices.

Another challenge is related to the data and workload location. In a cloud environment, the actual location of the servers running the workloads and storing the data is known only to the cloud owner or provider. Yet some data may be a subject of sensitive policies which restrict its placement into specific geographic boundaries. This is even more likely for the edge, where the physical hosts have inherently distributed nature. Moreover, the Quality of Service (QoS) may require placing the workloads in close proximity to the end user, for example, to achieve a required minimum latency of a streaming service.

Solving all challenges of the edge computing at once is an ambitious task, however, any of its sub-tasks is an interesting research topic by itself. In this work, we focus on the problem of workload location and we propose a location-aware virtual machine (VM) deployment on the edge. We built an edge infrastructure of trusted ARM devices, able to support hosts authentication and integrity check as well as to securely store geolocation information. In order to achieve completeness of our edge computing system, we use the management capabilities of OpenStack [6] and we integrate attestation and geo-fencing functionality in the OpenStack Compute project.

The next sections are organized as follows. Section II gives some background on the topics of trusted computing and geo-fencing. The architecture and details of the proposed solution are given in Section III followed by experimental setup and results in Section IV. In the end, we present a short survey of related works and give our conclusions in Sections V and VI.

II. Trusted computing and geo-fencing background

A. Trusted computing

The challenge of building a trusted virtualized infrastructure has been a topic of many works which have identified the need for a hardware root of trust and external attestation [7], [8].

For the former, the trust into a device cannot be based only on software, there needs to be a hardware component which serves as a trust anchor or what is called a hardware root of trust. Currently there are two main trends in this area, driven by two standardization bodies: the Trusted Computing Group (TCP) and the Global Platform (GP). The TCP define

the Trusted Platform Module (TPM) [9], a collection of cryptographic functions often implemented by a dedicated hardware chip outside the main processor. The GP specify the Trusted Execution Environment (TEE) [10], this is a secure isolated environment on the same System-on-Chip (SoC). Two main industry technologies are implementing a TEE:

- *Intel SGX* or Intel Software Guard Extensions [11] is an Intel architecture extension with protected areas of execution in the memory, called enclaves. The developers can specify parts of their application which run isolated into the enclaves, protected from other applications, the operating system (OS) or even the firmware.
- *ARM TrustZone* [12] provides a hardware isolation for the parallel execution of a secure environment and a rich OS, together with a mechanism for the context switch between the two.

The latter is concentrated on the remote attestation of a device as an additional measure of trust. A formal definition of the process is given in [8] while a popular open-source implementation is the OpenAttestation project [13].

B. Geo-fencing

The problem of location-aware workload placement and migration can be seen as restricting the workload based on pre-defined location requirements. In the literature, this is also called geo-fencing. The term geo-fence has a broader sense as a virtual perimeter for a real-world geographical area formed by location attributes [14], [15]. The determination of the geographical position of an object is known as geolocation. A device geolocation can be discovered by an integrated Global Positioning System (GPS), Radio Frequency Identification (RFID), the device IP address or even manually provisioned. Once the location is determined it has to be stored securely on the device, the process is often referred to as geo-tagging. The secure geolocation is a separate research topic which is not in the focus of this work. Here, we concentrate on the processes of geo-tagging and geo-fencing.

In the context of cloud and edge computing, geo-fencing is used to set-up policies based on the geolocation attributes provisioned to the hosts of the virtualized infrastructure [14]. Such policies can be related to government laws that restrict the workloads to a concrete country but also can define a secure zone for placing a critical service or can be based on the proximity to the user and the insurance of real-time processing.

III. TrustedVIM solution

As described in the previous sections, our work has two main goals, secure and location-aware workload execution. The solution that we propose in order to achieve them is called Trusted Virtualized Infrastructure Manager (TrustedVIM). As its name suggests, it has two main functions - creating a trusted virtualized infrastructure on one side and the infrastructure management on the other. The first one helps us building a secure edge on which we can safely execute VM workloads and store essential information such as keys and tags (location, asset identification, etc.). The management function provides a remote attestation of the edge compute hosts and a location-awareness during VM deployment. The main components of the TrustedVIM architecture are shown in Figure 1.

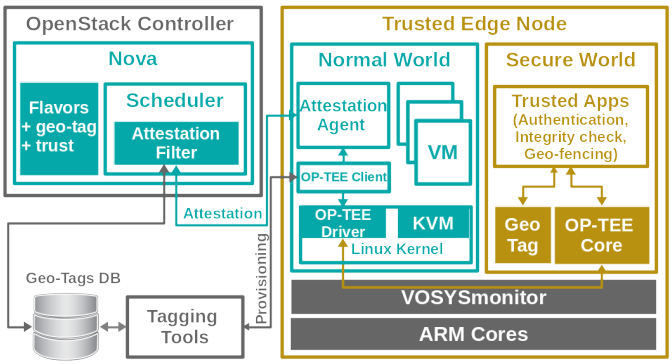

Fig. 1. TrustedVIM Architecture

A. ARM TrustZone-based edge nodes

The key component is the edge infrastructure based on ARM TrustZone-enabled nodes running VOSYSMonitor [16].

This mixed-critical infrastructure provides each node (host) with an isolated environment for storing and running sensitive data and applications. Each device has two execution environments, called Normal World and Secure World. The Normal World hosts a rich operating system, KVM-enabled Linux in the case of this work, where the virtualization services and the workloads are running. Meanwhile, the Secure World provides a secure environment for the critical tasks of the Normal World as well as a secure storage for cryptographic keys and other critical data. The context switch between the two worlds is done by VOSYSMonitor - a thin monitoring software layer developed by Virtual Open Systems.

In addition, the edge nodes make use of secure boot to propagate the trust to the upper software layers. It is accomplished by a vendor-specific mechanism relying on a hardware root of trust and is propagated to the Linux user-space by U-Boot's verified boot [17]. The boot process is multi-staged and starts from the boot read-only memory (ROM) and the first-stage boot loader (FSBL). After the FSBL is determined as secure, it loads the open source boot-loader U-Boot which on its turn verifies the signature of the Linux kernel image and only then loads the operating system.

B. OpenStack extensions

We supplement the management capabilities of OpenStack with attestation extensions integrated directly into the OpenStack Compute (Nova) project. By doing this, we build upon the features of a well-established open-source cloud manager and we add the missing pieces for solving our problem. The extensions are part of the instance (VM) scheduling process and are responsible for the verification of the compute nodes' authenticity and integrity and the location-awareness. On the request of a new OpenStack instance creation, the attestation

service filters the available compute nodes by their trust state. Only nodes that are decided to be trustworthy are passed through and can be used as hosts for the instance placement.

Nova-scheduler is one of the internal components of OpenStack Nova and is part of the flow for provisioning a new instance. By default, the nova-scheduler is configured as a filter scheduler and implements a decision-making mechanism selecting the host on which new instance should be created. During this process the scheduler iterates through all compute nodes, evaluating each against a set of filters. The list of resulting hosts is ordered by weights and is used to select the one that meets all VM requirements. If the scheduler cannot find a candidate, it means that there are no appropriate hosts where that instance can be scheduled [18].

As part of our work, we implement a custom filter, called *attestation_filter*, part of the filter scheduler. It is complemented by an *attestation_agent* application running on each compute node. The OpenStack Nova is configured to run the attestation filter during the default filtering procedure. It establishes a connection with the *attestation_agent* and gathers information about the trust status of the platform. Based on this data it informs the scheduler if each of the hosts passes or not.

C. OP-TEE Secure services

Open Portable Trusted Execution Environment (OP-TEE) is an open-source implementation of a TEE meeting the Global Platform TEE specifications. OP-TEE is designed to run into the Secure World in parallel with a Rich Execution Environment (REE) which is, in this case, a Linux OS running in the Normal World. The underlying hardware isolation is based on ARM TrustZone [19].

We implement several services providing the security of the edge nodes as OP-TEE trusted applications inside the Secure world where they cannot be affected by software running in the Normal World. We use the OP-TEE secure storage to keep the geolocation of the edge nodes in a place where it cannot be maliciously modified.

1) Edge node authentication: The edge nodes authentication is initiated by the OpenStack Controller and is performed before the start of the provisioning. The Controller uses a ”Challenge-response” protocol, based on asymmetric cryptography, to verify that the compute node is actually the one that it is claiming to be. This is a fundamental service to avoid man-in-the-middle attacks.

We define a secure communication protocol between the OpenStack Controller and the edge nodes with the goals of verifying the node authenticity, keeping the message integrity and the communication secrecy. The OpenStack Controller knows the public key of all the compute (edge) nodes, while all the compute nodes know the public key of the OpenStack Controller. Obviously, the private key is known only by the actor itself. All keys are securely stored in OP-TEE. Thanks to this, it is possible to exchange a packet composed of the following elements:

- *Encrypted message*: granting secrecy since only the receiver can decrypt the content of the message. Moreover, it contains a nonce that makes ineffective reply attacks.
- *Signature*: acting as a Message authentication code (MAC) it guarantees the authenticity and integrity of the packet.

By following this protocol, we achieve a secure communication secure channel that allows end-to-end encryption between the OpenStack Controller and the OP-TEE operating system. Even if the Linux kernel on the edge node is comprised, an attacker cannot alter this communication channel. The only possible attack is the interruption of the channel, but this can be already interpreted as a ”red flag” by the OpenStack Controller.

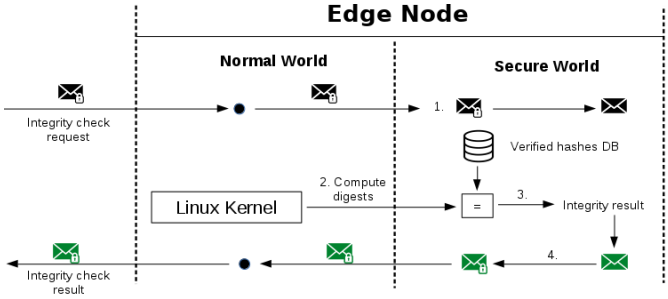

Fig. 2. Integrity check diagram

2) Linux kernel integrity check: Detecting all signs of infection is a task hard to achieve. Having said that, this implementation intends to serve as a proof of concept and to demonstrate the feasibility of the work and measure its performance.

In order to interfere with the execution of a VM a malicious software (commonly referred to as malware) has to gain high privileges. Usually, an infection of the kernel space is needed. The main malware category that aims at this is the so-called ”rootkits”. One common technique used by rootkits is to tamper the kernel code to perform malicious actions. A valid countermeasure is a run-time integrity check of the kernel code memory. Although this doesn’t prevent the execution of a rootkit but only the detection of it, this information is valuable to mark the infected host as not trusted and take countermeasures. We define our main attack vector as a rootkit and implement the Linux integrity check by monitoring two essential tables present in the Linux kernel:

- *sys_call_table*: this table contains all the pointers to the functions implementing the system calls. Since the interface between user and kernel space goes entirely through this interface, it is mandatory to verify the integrity of this table to avoid tampering of those.
- *vectors*: in the ARM architecture whenever an exception occurs the processor switches the execution to the correct entry in the vector table. This table is also critical for the safety and it should not be never changed.

Thanks to OP-TEE it is possible to verify at run-time that these tables are not modified. To perform this it is sufficient to compute the digest of the tables and verify that it is exactly as the one expected. OP-TEE embeds in its code the digests of the tables computed offline using the verified kernel image. Figure 2 summarizes the steps of this integrity check.

3) Location-aware VM deployment: The location-aware VM deployment can be split into two phases. The geo-tag provisioning phase when we load and store the geolocation to the edge compute nodes and the actual deployment phase.

The components that play a part during the first phase are the geo-tag database and a tagging tool. Since the acquisition of the edge location is not the goal of this research, we use pre-defined static geolocation information and leave the dynamic geolocation for future works. The tagging tool has the role of provisioning the geo-tag to the trusted edge node. The tags are stored in a database with a JSON format. The tool extracts the needed tag from the database, calculates its hash digest and sends it to the nodes Linux part (Normal World). Then the control is transferred to the OP-TEE where the digest is stored. The proposed geo-tag structure consists of GPS coordinates and an additional text description:

```
{"host": {
   "trusted": "true/false",
   "tag":     "tag string",
   "geolocation": {
      "latitude" :  12.345,
      "longitude": -67.890  }
}}
```

Alternative geo-tag structures can consider regions, road names, cities blocks, buildings or any other geographical concept instead of GPS coordinates.

After the tag is provisioned to the trusted edge compute nodes, the VM deployment can start. The OpenStack Flavors are updated to include the geo-tag attributes. Before VM instantiation, Nova (through the nova-scheduler and its filters) checks the DB for a tag with matching attributes, calculates the tags hash and attests the compute node. Through the already existing mechanism of encryption and signing the hash value is sent to the Attestation agent and the OP-TEE trusted applications (TAs). The TA receives the calculated hash and compares it with the stored one. If the two values match, a success is returned and the VM provisioning on this node can continue.

IV. Experimental setup and results

The following experiment is designed in order to evaluate the performance overhead added by the proposed security and geo-fencing enhancements. Our work extends the VM deployment process, hence, we are comparing the boot and delete times of a VM in two scenarios. The first one is a vanilla OpenStack deployment with one controller node and one edge compute node without any extensions. The second one includes the security and location-awareness features. However, the added logic is executed during the VM scheduling, before the VM provisioning phase, therefore the results are independent from the VM and guest OS size. Each run is repeated 50 times.

TABLE I
Boot and delete 50 VMs, without security and geo-fencing

Action	Min (sec)	Median(sec)	90%ile(sec)	Max(sec)	Avg(sec)	Count
boot_instance	12.816	12.939	13.084	13.4	12.97	50
delete_instance	2.35	2.366	2.39	2.55	2.378	50
total	**15.182**	**15.319**	**15.504**	**15.778**	**15.384**	**50**

TABLE II
Boot and delete 50 VMs, with security and geo-fencing

Action	Min (sec)	Median(sec)	90%ile(sec)	Max(sec)	Avg(sec)	Count
boot_instance	12.821	12.986	15.129	15.211	13.549	50
delete_instance	2.353	2.372	2.556	2.595	2.414	50
total	**15.197**	**15.435**	**17.524**	**17.664**	**15.963**	**50**

OpenStack is deployed with Devstack [20] in a multi-node configuration:

- *One x86 controller node:* Intel(R) Xeon(R) CPU E5-2623 v4 @ 2.60GHz, 32GB memory, Ubuntu 16.04.4 LTS, KVM-enabled 4.4.0-128 Linux kernel
- *One aarch64 compute node:* Xilinx Zynq UltraScale+ MPSoC ZCU102 with a quad-core ARM Cortex-A53, 4GB memory, Ubuntu 18.04.4 LTS, KVM-enabled 4.14.0 Linux kernel

The hardware configuration of the VMs is based on the default OpenStack m1.tiny flavor (1 VCPU, 1GB Disk, 512MB RAM) and the booted guest OS is a CirrOS cloud image. For the custom set up, we create a flavor derived from m1.tiny extended with trust and location properties that have to be matched during the VM scheduling process. The measurements are performed by Rally [21], a testing tool for multi-node deployment evaluation developed as an OpenStack project [22].

The results can be seen in Tables I and II. Our implementation affects the VM creation while the deletion remains unchanged which is confirmed by the delete times. The security and location-awareness add an average overhead of 0.579s to the total VM boot time which is $\sim 4\%$ of the vanilla deployment result. However, the Min and Median values between the two scenarios are very close which leads to the conclusion that there is a variation caused by the networking speed. This can be avoided by a better implementation and is a direction for future improvements.

V. Related work

There is ongoing research in the direction of reducing the trust in the cloud providers and securing the infrastructure through hardware technologies. The authors of [23] follow this path and propose a common platform for the integration

of edge devices with public cloud infrastructures. Their work utilizes Intel SGX on the cloud and ARM TrustZone for securing the edge, confirming in their experiments that a trusted environment based on ARM TrustZone has a negligible performance overhead. Another interesting work is [24] which shows an orchestrator based on Kubernetes capable of scheduling containers on a heterogeneous cluster of servers taking into account the Intel SGX support. This allows the execution of critical parts of the workloads in a hardware-secured environment without trusting the providers.

An exhaustive research by [14] is proposing a complete solution for a secure cloud infrastructure. They are addressing the problems of platform integrity, external attestation, boundary control and VM integrity together with a reference architecture. The hardware technology on which the framework relies is Intel Trusted Execution Technology (TXT) [25]. The Intel TXT provides the hardware root-of-trust for measurements and a TPM is used for secure storage and reporting. Based on that they present a concept for a complete cloud architecture. In a similar way, the work of [26] suggests an architecture for trusted geo-fenced hybrid clouds based on Intel TXT and a TPM. They focus on improving the scalability of existing trustworthy boot and integrity measurement techniques as well as the geo-location of servers and virtual machines and the prevention of compromising the geo-fencing software.

The authors of [14] and [26] have laid the foundations of a trusted cloud whichs need to be further extended to accommodate the edge computing shift. As shown in [23] and [24], there are ongoing efforts in this direction with open challenges related to the sensitive data and workload placement.

VI. Conclusion

In this work, we propose secure and location-aware extensions to the workloads placement process with a focus on the virtualized edge. We reflect on the workloads privacy issues and the location-aware edge and propose a way to safely incorporate the device geolocation during the workload placement. Additionally, we address the security challenges of the edge computing paradigm by offering an attestation mechanism verifying the authenticity and the integrity of the devices, based on the hardware isolation of ARM TrustZone.

We evaluate the computational overhead of our work by conducting a series of experiments and compare it with an identical deployment without our extensions. Our security solution has a small impact on the VMs boot time ($\sim$ 4%), with a margin for improvement.

Our future efforts are pointed to the direction of securing the VMs by virtualizing the TEE and as a result providing each VM with access to its own isolated execution environment. Another important research topic is the adoption of light-weight virtualization technologies at the edge, such as unikernels and containers and ensuring their security.

Acknowledgment

This work was partly funded by the European Commission under the European Union's Horizon 2020 program grant agreement number 761508 (5GCity project). The paper solely reflects the views of the authors. The Commission is not responsible for the contents of this paper or any use made thereof.

References

[1] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *arXiv preprint arXiv:1808.05283*, 2018.

[2] S. Lal, T. Taleb, and A. Dutta, "Nfv: Security threats and best practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, 2017.

[3] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[4] Z. Mahmood, "Data location and security issues in cloud computing," in *2011 International Conference on Emerging Intelligent Data and Web Technologies*. IEEE, 2011, pp. 49–54.

[5] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[6] "Openstack," [accessed 08-February-2019]. [Online]. Available: https://www.openstack.org/

[7] G. Hunt, G. Letey, and E. Nightingale, "The seven properties of highly secure devices," *tech. report MSR-TR-2017-16*, 2017.

[8] G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. OHanlon, J. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," *International Journal of Information Security*, vol. 10, no. 2, pp. 63–81, 2011.

[9] "Tpm main specification," [accessed 08-February-2019]. [Online]. Available: https://trustedcomputinggroup.org/resource/tpm-main-specification/

[10] "Global platform specifications," [accessed 23-January-2019]. [Online]. Available: https://globalplatform.org/specs-library/

[11] V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptology ePrint Archive*, vol. 2016, no. 086, pp. 1–118, 2016.

[12] S. Pinto and N. Santos, "Demystifying arm trustzone: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, p. 130, 2019.

[13] "Openattestation project," [accessed 08-February-2019]. [Online]. Available: https://01.org/OpenAttestation

[14] R. Yeluri and E. Castro-Leon, *Building the Infrastructure for Cloud Security: A Solutions View*. Apress, 2014.

[15] W. J. Haddock and P. Lunsford, "Geo-fencing technologies and security."

[16] P. Lucas, K. Chappuis, B. Boutin, J. Vetter, and D. Raho, "Vosysmonitor, a trustzone-based hypervisor for iso 26262 mixed-critical system," in *2018 23rd Conference of Open Innovations Association (FRUCT)*. IEEE, 2018, pp. 231–238.

[17] S. Glass, "Verified u-boot," 2013.

[18] "Openstack schedulung," [accessed 08-February-2019]. [Online]. Available: https://docs.openstack.org/mitaka/config-reference/compute/scheduler.html

[19] "Open portable trusted execution environment," [accessed 08-February-2019]. [Online]. Available: https://github.com/OP-TEE/optee_os/blob/master/Notice.md

[20] "Devstack documentation," [accessed 08-February-2019]. [Online]. Available: https://docs.openstack.org/devstack/latest/

[21] "Rally documentation," [accessed 08-February-2019]. [Online]. Available: https://rally.readthedocs.io/en/latest/

[22] T. Pflanzner, R. Tornyai, B. Gibizer, A. Schmidt, and A. Kertesz, "Performance analysis of an openstack private cloud," 2016.

[23] R. Pettersen, H. D. Johansen, and D. Johansen, "Secure edge computing with arm trustzone," in *Proc. 2nd Int. Conf,. Internet Things, Big Data Secur.(IoTBDS)*, 2017, pp. 102–109.

[24] S. Vaucher, R. Pires, P. Felber, M. Pasin, V. Schiavoni, and C. Fetzer, "Sgx-aware container orchestration for heterogeneous clusters," *arXiv preprint arXiv:1805.05847*, 2018.

[25] J. Greene, "Intel trusted execution technology (white paper)," *Online: http://www. intel. com/txt*, 2012.

[26] K. Jayaram, D. Safford, U. Sharma, V. Naik, D. Pendarakis, and S. Tao, "Trustworthy geographically fenced hybrid clouds," in *Proceedings of the 15th international middleware conference*. ACM, 2014, pp. 37–48.