Intrusion detection and attack tolerance for cloud environments: the CLARUS approach

Georges Ouffoué[1], Antonio M. Ortiz[2], Ana R. Cavalli[2], Wissam Mallouli[2], Josep Domingo-Ferrer[3], David Sánchez[3], Fatiha Zaidi[1]
[1] Univ. Paris-Sud, LRI; CNRS Orsay, France
Email: {ouffoue, zaidi}@lri.fr
[2] Montimage, Paris, France
Email: {antonio.ortiz, ana.cavalli, wissam.mallouli }@montimage.com
[3] UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics,
Universitat Rovira i Virgili, Av. Paisos Catalans, 26, 43007, Tarragona, Catalonia
Email: {josep.domingo, david.sanchez}@urv.cat

***Abstract*—The cloud has become an established and widespread paradigm. This success is due to the gain of flexibility and savings provided by this technology. However, the main obstacle to full cloud adoption is security. The cloud, as many other systems taking advantage of the Internet, is also facing threats that compromise data confidentiality and availability. In addition, new cloud-specific attacks have emerged and current intrusion detection and prevention mechanisms are not enough to protect the complex infrastructure of the cloud from these vulnerabilities. Furthermore, one of the promises of the cloud is the Quality of Service (QoS) by continuous delivery, which must be ensured even in case of intrusion. This work presents an overview of the main cloud vulnerabilities, along with the solutions proposed in the context of the H2020 CLARUS project in terms of monitoring techniques for intrusion detection and prevention, including attack-tolerance mechanisms.**

I. Introduction

Cloud computing, as a paradigm of distributed systems, has emerged over the last decade. Following the definition provided by the NIST [1], cloud computing enables organisations to run applications (often referred to as services) on a pay-as-you-go basis on top of reliable, highly-available, and scalable software and hardware infrastructures referred to as clouds. This new concept has revolutionised the IT industry by considerably reducing production and operating costs for companies. From a technical perspective, the cloud is the union of two main concepts: grid computing and virtualization. Grid computing enables the aggregation of distributed resources, while virtualization provides the following three properties: pooling, location independence, and elasticity.

Despite the various facilities offered by the cloud, security remains the main Achille's heel for its wide adoption. Cloud infrastructures, like many others taking advantage of the Internet, are also facing attacks on availability, integrity and confidentiality of platforms and user data. Recently, new attacks exploiting cloud vulnerabilities (such as VM escape, hacked interfaces and APIs, account hijacking, etc.) are reducing the effectiveness of traditional detection and prevention means (e.g., firewall, intrusion detection systems, etc.) available in the market. However these systems may not be sufficient to fully ensure security in environments such as the cloud. Moreover, they can only detect existing attacks (whose signatures exist in viral bases) with relatively high rates of false positives and negatives, while the cloud must be able to ensure a level of safety even in the presence of unknown (non-previously defined) intrusions.

Research on cloud security has been focused on the design of secure protocols for exchanging messages that meet the confidentiality, integrity and availability requirements [2]. Because of this limitation, a cloud system may fail to perform its mission when an attack is successful, and it may be unable to recover quickly. The work presented in [3] views security as a system property that must be continuously managed during the whole lifetime of a system. Research on attack and intrusion detection systems has mainly focused on how to detect as many attacks as possible, as soon as possible, and at the same time, to reduce the false alarm rate. However, it has been increasingly recognised that a variety of critical applications need to be able to continue in operation or to provide a minimal level of service even when they are under attack or have been partially compromised; hence the need for attack-tolerant systems.

In this sense, the H2020 CLARUS project arises to provide an attack-tolerant framework to improve trust in cloud computing while securely unlocking sensitive data to enable new and better cloud services. The ultimate goal of CLARUS is developing a secure framework for storing and processing data outsourced to the cloud so end-users can monitor, audit and control their stored data to regain trust in cloud computing.

The remainder of this paper is organised as follows: Section II introduces an overview of some of the most common cloud-based threats; the CLARUS project is presented in Section III; Section IV provides a deep analysis of the security, metrics, and monitoring mechanisms used in CLARUS. Section V shows the intrusion tolerance mechanisms that will be developed in the CLARUS framework. Finally, conclusions and future work are outlined in Section VI.

II. Cloud based threats

The new threats introduced by cloud computing can be classified as follows [4]:

A. Data leakage vulnerabilities

1) Data loss: mostly occurs due to malicious attackers, data deletion, data corruption, loss of encryption keys, faults in the storage system, or natural disasters [5]. Data loss can have catastrophic consequences in the business, which may result in bankruptcy [6]. Another source of data loss are insecure interfaces and APIs (Application Programming Interface) [4], [7]. Cloud Services Providers (CSPs) expose APIs to third-parties to interact with their cloud services. In fact, whether launching applications services (SaaS), deploying their own applications (PaaS), or managing virtual machines (IaaS), cloud consumers use these APIs. Since the APIs are accessible from anywhere in the Internet, malicious attackers can use them to compromise the confidentiality and integrity of the enterprise customers [5].

2) Cache-based side-channel: the prowess of modern cryptography led to the design of algorithms mathematically proven robust and secure [8]. Although these algorithms are safe, their software and hardware implementations may be prone to attacks called covert-channel or side-channel attacks. A side-channel attack uses communication means that are not normally designed to leak the information [9]. These attacks consist of two steps. First, a detailed analysis of the power consumption, the electromagnetic emanation or any other source is made. Then, the exploitation of this analysis gives the attacker the ability to recover some bits of the encryption keys. In the cloud, the main micro-architectural leakage source is the cache. The first study that revealed cached-based side-channel in cloud computing platforms was [10]. This method permitted to execute a coarse-grained keystroke attack on the public cloud platform Amazon Web Services (AWS) [11].

There are also other vulnerabilities such as unauthorised access to data, data ex-filtration, denial of service, metadata modification, data sniffing/spoofing, etc., that are also oriented to get protected information from the cloud.

B. Virtualization vulnerabilities

Cloud computing leverages the concept of virtualization, that usually enables a new software layer (hypervisor) atop the operating system. However, virtualization introduces new security concerns [12]:

- **Detecting a virtualized environment**: officially, a virtualized platform acts as a real platform, but an attacker has the possibility to detect if a machine is virtualized by measuring some instructions at execution time. Then, performing a comparison of the obtained values with previous measurements, the virtualization can be detected.
- **Identifying the hypervisor**: all of the hypervisors have their own vulnerabilities and flaws. Normally a cloud user cannot know on which hypervisor his/her VMs are running. However a spy can identify the hypervisor by using some specific instructions not well supported by each kind of hypervisor.
- **VM escape**: bad configuration within the hypervisor can break isolation between the hypervisor and the host. A virtual machine could directly interact with the host operating system. This would compromise all the data stored in the affected virtual machine and can affect other virtual machines deployed on the same host.
- **VM hopping**: this kind of attack is based on a virtual machine accessing other virtual machines. This vulnerability allows remote attacks and malware to compromise the overall virtual machine security, also granting access for the attackers to the host computer, the hypervisor and other virtual machines.
- **Insecure VM migration**: information related to credentials, personal data, and other sensitive information can be retrieved by attackers in case of performing an insecure virtual machine migration.
- **Malicious VM creation**: the creation of malicious virtual machines constitutes a high risk for the system since they can replace the legitimate virtual machines offering similar services, but making a malicious use of the exchanged data.

III. The CLARUS project

CLARUS (http://www.clarussecure.eu/) is a H2020 European research project [13] that focuses on providing solutions to the usual security and privacy threats that affect cloud computing and hinder a franker migration by end users. By developing a secure framework for the storage and processing of data outsourced to the cloud and by increasing the users control on the security and privacy of their data, CLARUS aims at enhancing trust in cloud services. Regarding privacy, end users will have the guarantee that no other than themselves will be able to see or infer their sensitive data outsourced to the cloud. Regarding security, users will be assured that no intruder can hack the cloud and/or impersonate them and that no denial of service will occur. By means of the solutions provided by CLARUS, end users will be able to monitor, audit and control their outsourced data without impairing the functionality and cost-saving benefits of cloud services, which constitute the main appeal for migrating to the cloud. The main innovation provided by CLARUS is mainly focused on the development (and adaptation) of security tools for cloud environments, and on the integration of these mechanisms into a single proxy that will provide the means to effectively bring security and privacy to cloud-based data operations.

As shown in Fig. 1, the CLARUS solution is envisioned as a proxy located in a domain trusted by the end user (e.g., a server in his/her companys intranet, a plug-in in the users device) that implements security and privacy-enabling features towards the cloud service provider. To enhance privacy, CLARUS will implement a set of privacy-enabling mechanisms to ensure that the users sensitive data are properly

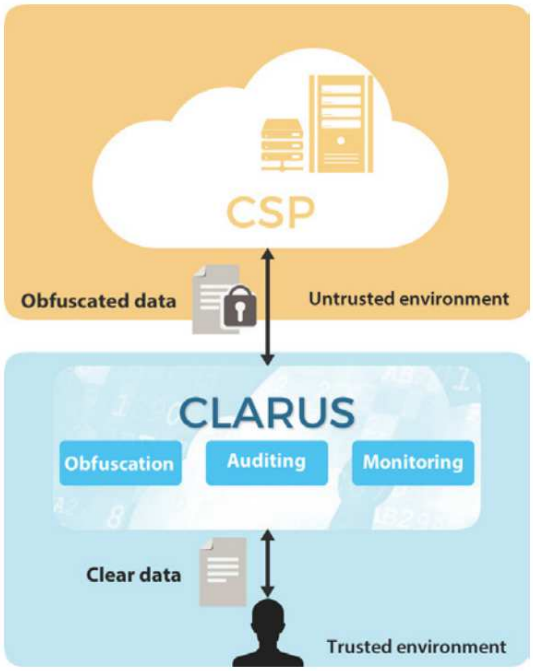

Figure 1. CLARUS architecture

protected before they are outsourced to the cloud in a way that cloud service functionalities are still preserved. To do so, CLARUS relies and innovates on the current state of the art on functionality-preserving cryptography [14], data anonymization [15] and splitting techniques [16]. To enhance security and users trust, CLARUS will also implement a set of auditing and monitoring services, so that users can directly supervise how data are being protected and detect security threats. By means of constant monitoring, CLARUS will also provide an attack-tolerant framework, so that potential security breaches within the cloud can be dynamically detected and appropriate mitigation measures can be activated on-line, so reducing the effects of the detected attacks.

In this way, users privacy, security and trust can be significantly enhanced with respect to current cloud security solutions, which are commonly located within the cloud platform and compel customers to blindly trust cloud providers.

IV. Security, metrics and monitoring in CLARUS

For CLARUS, monitoring represents a key tool to evaluate the behaviour of the system, being able to supervise its correct operation and to early detect any security and privacy issue. This section presents a global overview about monitoring mechanisms, requirements and data sources, illustrating how they will be integrated in CLARUS, as well as a description of those attacks that can be detected using monitoring mechanisms and the metrics that lead to the attack detection.

A. What is monitoring

Monitoring is the process of dynamically collecting, interpreting and presenting metrics and variables related to a system's behaviour in order to perform management and control tasks [17]. The idea behind monitoring is to measure and observe performance, connectivity, security issues, application usage, data modifications and any other variable that permits to determine the current status of the entity being monitored. By keeping a constant view of the different entities, we can obtain a real-time status of Key Performance Indicators (KPI) or Service Level Agreements (SLA) compliance as well as faults and security breaches. Monitoring can be performed in several domains that include user activity, network and Internet traffic, software applications, services and security. The monitoring processes should not disturb the normal operation of the protocol, application, or service under analysis.

A typical monitoring platform is basically composed of three main elements:

- **Monitoring probes**: they directly collect data from the sources to be further analysed by the analysis module. Probes can be distributed in diverse locations of the system (e.g., network, application, user side, etc.) in order to obtain a global view of the system.
- **Database**: diverse databases can be used. Commonly, one database is used to store the raw data gathered by the probes, and other databases containing diverse information (e.g., rules, security, performance) are also consulted to correlate the collected data in order to extract the required information.
- **Analysis module**: it contains the means to examine the input collected by the probes, correlate it with the information stored in the database and produce an output that can be used for different purposes.

Additional components can be added with the aim of performing supplementary tasks. For example, visualization components may be used to show the requested statistics to the involved actors.

B. Monitoring techniques

The general processes involved in monitoring are: define the detection method to track and label events and measurements of interest; transmit the collected information to a processing entity; filter and classify the events and measurements based on pre-defined criteria; and finally, generate decisions associated to the results obtained after the evaluation [17]. Regarding how to collect events and measurements, monitoring techniques can be classified into three main categories: active, passive and hybrid approaches.

Active monitoring: the System Under Observation (SUO) is stimulated in order to obtain responses to determine its behaviour under certain circumstances or events. This technique permits directing requests to the concerned entities under observation (see Fig. 2). However, it presents some drawbacks. The injection of requests towards the SUO might affect its performance. This will vary depending on the amount of data required to perform the desired tests or monitoring requests. For large amounts of data, the SUO processing load

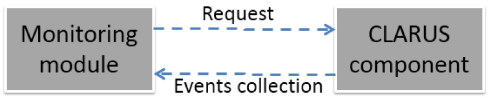

Figure 2. Basic active monitoring deployment

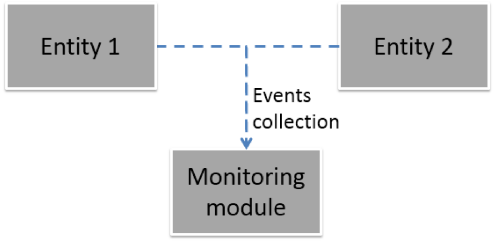

Figure 3. Basic passive monitoring deployment

might increase and produce undesirable effects. Secondly, the injected information might also influence the measurements that are being taken, for example incurring in additional delay. Lastly, active monitoring injects data that could be considered invasive. In a network operator context, it could limit its use and applicability [18].

Passive monitoring: consists on capturing a copy of the information produced by the SUO without a direct interaction [2] (see Fig. 3). This technique reduces the overhead required on active monitoring. Conversely, certain delay should be considered when analysing large amounts of data. Additionally, in some cases it is not always possible to perform real-time monitoring because of required offline data post-processing [3]. This technique has the advantage over the active approach of not performing invasive requests.

Hybrid monitoring: by combining the aforementioned approaches we can perform both active and passive monitoring. The idea is to keep passive trace collection and to inject requests as necessary to maintain a continuous flow of observations and measurements [19] (see Fig. 4). In this case, preconfigured, on-demand, or event-based monitoring requests can be performed. Hybrid monitoring can be used in systems with limited resources, for example by periodically sending passive traces (with limited information) and requesting details if complementary information is needed. In this way, the resources consumed by the monitoring modules are reduced.

The selection of one of these monitoring approaches will depend on different premises like the level of intrusion allowed by the SUO, the type of data collected (which sometimes does not require a direct request to be obtained), the specified security policies, or post-processing and real-time constrains.

C. Monitoring requirements

The main objectives of monitoring are: functional verification of the system under test, performance analysis, verification of security properties, and detection of security vulnerabilities

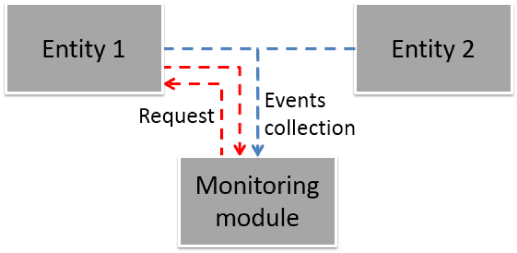

Figure 4. Basic hybrid monitoring deployment

and attacks. In order for a monitoring platform to be effective in the early detection of any performance, security, or privacy violation, the following requirements must be fulfilled [20]:

- **Data capturing performance**: traffic speed and data volume should not affect the collection of system information.
- **Extensibility**: it should be easy to incorporate new services to be monitored without requiring a great effort from the administrators.
- **Scalability**: the monitoring system should be adaptable to the increase of the data to be collected, or the services and applications to be monitored.
- **Near real-time operation**: in order to achieve early detection of performance or security issues, data capture must be aligned with real-time analysis capabilities.
- **Granularity**: the monitoring system should be capable of differentiating among the different protocols, services and applications that are monitored.
- **Diversity**: diverse network devices, protocol stacks, services, applications, etc., should be supported by the monitoring platform.
- **Low cost**: the amount of computing, storage and communication resources consumed by the monitoring system should be as low as possible.
- **Security**: the monitoring system should not add new vulnerabilities to the system.
- **Transparency**: the monitoring functions should not interact or disturb the normal operation of the system.

V. Intrusion and Attack Tolerance in CLARUS

Intrusion or attack tolerance of a system is generally understood as the capability to continue to function properly with minimal degradation of performance, despite intrusions or malicious attacks [21]. One of the main features of CLARUS is its attack-tolerant nature. Thus, the early design of the solution will integrate attack-tolerant mechanisms that will enable it to continue operating in the event of an attack.

A. Intrusion tolerance techniques

In order to enable attack-tolerance, and depending on the individual circumstances of the attack and the system itself, diverse techniques can be applied to keep the system operating in the event of an attack:

- **Redundancy and diversity**: redundancy refers to the extra reserved resources allocated to a system that are beyond its need in normal working conditions. Whilst, diversity means that a function should be implemented in multiple ways, differently at different times.
- **Voting**: is used to resolve any differences in redundant responses and to arrive at a consensus result based on the responses of perceived non-faulty components in the system. It has two complementary goals: masking of intrusions, thus tolerating them, and providing integrity of the data.
- **Acceptance test**: this issue usually consists of a sequence of statements that will raise an exception if the state of the system is not acceptable.
- **Threshold scheme and distributed trust**: the general idea is to devise a method to divide data D into n pieces in such a way that it needs at least k shares to reconstruct original data D. Anything less, reveals no information at all.
- **Dynamic reconfiguration**: reconfiguration after the detection of an intrusion in traditional systems is mostly reactive and generally performed manually by the administrator, thus, involves some downtime.
- **Indirection**: the common goal of all indirection techniques is to separate clients and servers by an additional layer that plays the role of protection barriers.

B. Mapping between attacks and countermeasures

Since attack-tolerance is the final objective, we should recognize that identification of intrusion and attacks is a key to obtain tolerance. In other words, intrusion and attack detection of some form is required to provide intrusion attack tolerance. In CLARUS, depending on the detected attack, the detection will be made by the Montimage Monitoring Tool (MMT) [22], and a series of countermeasures will be applied in order to mitigate its effects. Table I shows the countermeasures that will be applied by the CLARUS framework depending on the detected attack.

Table I. COUNTERMEASURES APPLIED IN CLARUS FOR EACH KIND OF ATTACK

Attack	Countermeasures
Access by unauthorised users	Expelling of the unauthorised user; source IP blocking; alarm for the administrator to manage the IP blocking
Impersonation of authorised users	Alarm to notify both the administrator and the affected user; access temporary limited for the affected user until credentials update
Insider attacks	Immediate user expulsion; source IP temporary blocking; notification to the users and administrators
Access to trusted zone	User expelling; alarm to the administrator; temporary close of the affected zone
Modification of metadata stored in CLARUS	Warning to the user and administrator; temporary user repelling; metadata restore; alarm to the administrator
Man in the Middle attacks	Module operation hold; alarm to the administrator
DoS/DDoS attacks	Temporary new access blocking; warning/alarm to the administrator
Data leakage	Depends on the attack nature
VM escape	Stop host; alarm to the administrator
VM hopping (jumping)	Alarm to the administrator
Malicious VM creation	Virtual machine creation stop; alarm to the administrator
Insecure VM migration	Virtual machine stop; deletion in the destination host; alarm to the administrator
Sniffing/spoofing virtual network/links	User block; virtual network redeployment; Warning to the administrator

C. How to approach intrusion tolerance in CLARUS

To date, there exist a number of solutions that provide security for cloud systems using one or a combination of the techniques we presented in Section V-A. However, there are very few that are able to manage the intrusions and the attacks and able to insure the reactions and countermeasures with the aim of protecting the system and guaranteeing its normal behaviour in hostile environments. The framework presented in [23] leverages the MAFTIA framework for general intrusion tolerance [24]. The detection relies on event analysis, while the intrusion detection is based on threshold cryptography. After the detection, the recovery module reallocates the VM running on these hosts and the hosts are turned off. But, the main bottleneck of this architecture is the performance overhead.

Other approaches have been developed to cope with intrusion-tolerance. The work in [25] explores how to build distributed systems that are attack-tolerant by design. The idea is to implement systems with equivalent functionality that can respond to attacks in a more safe way. As a result a number of code variants are produced, which ensures the system will be more resistant to attacks. In [26], the authors propose a formalism based on graphs to model an intrusion tolerant system. In this model they introduce system's response to (some of) the attacks.

In CLARUS, we propose to design an attack tolerant system that integrates intrusion detection methods, diverse defence strategies, and countermeasure techniques. To cope with all these issues it is necessary to consider information security as a permanent issue that needs to be managed in order to obtain attack-tolerant systems.

We consider that a system is correct-by-construction if we create a formal model of the system, derive some other models from the first one, verify that the new models satisfy the global security properties of the system and finally generate source code according to the chosen model that face to the potential attack. Fig. 5 illustrates our approach in which the original formal model is adapted considering the measurements made by the monitoring system.

The practical use of the proposed framework, techniques and tools will be demonstrated and evaluated in real application deployments. In particular, an eHealth and a Geodata application will test the proposed approach.

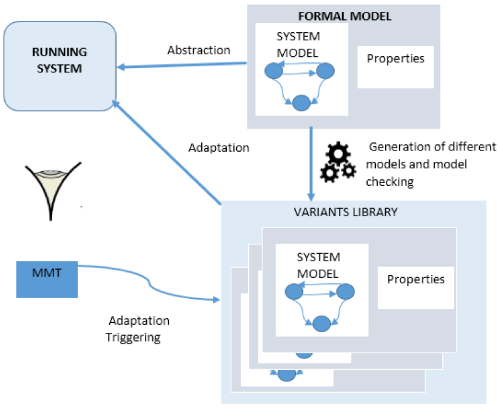

Figure 5. Intrusion tolerance framework

VI. Conclusions and Future Work

It is not enough to only detect known or new intrusions; a cloud system needs to react against attacks and repel them or at least mitigate their effects, while keeping the system running. This is one of the key ideas of CLARUS, of which main innovation will be the design and implementation of a framework for cloud systems, tolerant to intrusions and attacks, which can continue to deliver its services even in the presence of attacks. The current version of the CLARUS architecture is focused on an individual CLARUS proxy (even though this proxy may manage several users within the same organisation). The future work is oriented to the deployment of several CLARUS proxies that will interact with each other in order to implement, for example, collaborative services outsourced to the cloud. Specific application cases involving distinct final users will provide a testbed for evaluation: an eHealth application involving a public hospital and a Geodata application.

Acknowledgements

The project leading to this paper has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 644024.

References

[1] P. Mell and T. Grance, "NIST Special Publication 800-145 The Definition of Cloud Computing," http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[2] J. Curtis, "Passive measurement," http://wand.cs.waikato.ac.nz/old/wand/publications/jamie_420/final/node9.html.

[3] K. Anagnostakis, S. Ioannidis, S. Miltchev, M. Greenwald, J. Smith, and J. Ioannidis, "Efficient packet monitoring for network management," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2002, pp. 423–436.

[4] S. Y. Z. Muhammad Kazim, "A survey on top security threats in cloud computing," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, 2015.

[5] "The top cloud computing threats and vulnerabilities in an enterprise environment," http://www.cloudcomputing-news.net/news/2014/nov/21/top-cloud-computing-threats-and-vulnerabilities-enterprise-environment/.

[6] "Vulnerable apis continue to pose threat to cloud," http://www.darkreading.com/risk/vulnerable-apis-continue-to-pose-threat-to-cloud/d/d-id/1138983.

[7] L. B. Raju M, "Survey about cloud computing threats," *(IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 5, pp. 384–389, 2014.

[8] "RSA algorithm," https://people.csail.mit.edu/rivest/Rsapaper.pdf.

[9] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, 1999, pp. 388–397.

[10] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS, 2009, pp. 199–212.

[11] "Amazon AWS," http://aws.amazon.com/fr/.

[12] I. Studnia, E. Alata, Y. Deswarte, M. Kaâniche, and V. Nicomette, "Survey of security problems in cloud computing virtual machines," *Computer and Electronics Security Applications Rendez-vous (C&ESAR). Cloud and security: threat or opportunity*, vol. 5, pp. 61–74, November 2012.

[13] D. Sánchez and J. Domingo-Ferrer, "Clarus - a framework for user centred privacy and security in the cloud," in *Position paper at Cloudscape VII*, 2015.

[14] M. Azraoui, K. Elkhiyaoui, M. Onen, and R. Molva, "Publicly verifiable conjunctive keyword search in outsourced databases," in *1st IEEE International workshop on Security and Privacy in the Cloud (SPC)*, 2015.

[15] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and S. Martínez, "t-Closeness through microaggregation: strict privacy with enhanced utility preservation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 1041–4347, Oct. 2015.

[16] A. Calviño, S. Ricci, and J. Domingo-Ferrer, "Privacy-preserving distributed statistical computation to a semi-honest multi-cloud," in *1st IEEE International workshop on Security and Privacy in the Cloud (SPC)*, 2015.

[17] V. Stankovic and L. Strigini, "A survey on online monitoring approaches of computer-based systems," June 2009. [Online]. Available: http://openaccess.city.ac.uk/531/

[18] T. Chen and L. Hu, "Internet performance monitoring," in *Proceedings of the IEEE*, Sep 2002, pp. 1592 – 1603.

[19] M. Zangrilli and B. Lowekamp, "Using passive traces of application traffic in a network monitoring system," in *Proceedings of the 13th IEEE International Symposium on, High performance Distributed Computing*, 2004, pp. 77 – 86.

[20] J. Abrial, *The B Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[21] F. Wang, R. Uppalli, and C. Killian, "Analysis of techniques for building intrusion tolerant server systems," in *IEEE Military Communications Conference (MILCOM)*, vol. 2, 2003, pp. 729–734.

[22] W. Mallouli, B. Wehbi, E. M. de Oca, and M. Bourdelles, "Online network traffic security inspection using mmt tool," *System Testing and Validation*, vol. 192, 2012.

[23] V. Karande and A. Pais, "A framework for intrusion tolerance in cloud computing," in *Advances in Computing and Communications*, ser. Communications in Computer and Information Science, 2011, vol. 193.

[24] R. Stroud, I. Welch, J. Warne, and P. Ryan, "A qualitative analysis of the intrusion-tolerance capabilities of the maftia architecture," in *International Conference on Dependable Systems and Networks*, 2004, pp. 453–461.

[25] C. Robert, B. Mark, and V. R. Robbert, "Investigating Correct-by-Construction Attack-Tolerant Systems," Department of Computer Science, Cornell University, Tech. Rep., 2011.

[26] B. B. Madan and K. S. Trivedi, "Security modeling and quantification of intrusion tolerant systems using attack-response graph," *Journal of High Speed Networks*, vol. 13, no. 4, pp. 297–308, 2004.