Towards intrusion-resilient security monitoring in multi-cloud infrastructures

Hans P. Reiser
University of Passau
Innstr. 43
94034 Passau, Germany
hr@sec.uni-passau.de

Abstract
Multi-cloud architectures enable the design of resilient distributed service applications. Such applications can benefit from a combination of intrusion-tolerant replication across clouds with intrusion detection and analysis mechanisms. Such mechanisms enable the detection of attacks that affect multiple replicas and thus exceed the intrusion masking capability, and in addition support fast reaction and recovery from local intrusions.

In this work-in-progress paper we present a security analysis on which an intrusion detection and analysis service can be based on. We sketch the architecture of such a cross-cloud intrusion detection architecture that combines a set of well-known mechanisms. The goal of our approach is obtaining a resource-efficient service with optimal resilience against malicious attacks.

CCS Concepts • Security and privacy → Distributed systems security; Intrusion detection systems;

Keywords Cross-cloud replication, intrusion tolerance, intrusion detection

ACM Reference format:
Hans P. Reiser. 2017. Towards intrusion-resilient security monitoring in multi-cloud infrastructures. In Proceedings of XDOM0’17, Belgrade, Serbia, April 23, 2017, 6 pages.
DOI: http://dx.doi.org/10.1145/3071064.3071067

1 Motivation
It is an evident fact that computer-based systems are faced with malicious attacks. Increasing interconnectivity, increasing system complexity, and increasing means and motivation of attackers all contribute to continuously growing frequency and sophistication of these attacks. As it seems infeasible to prevent all attacks, the ability to cope with and survive malicious attacks that have successfully compromised (parts of) a system is of significant importance.

Fundamental concepts how to cope with intrusion faults can be classified into detection, recovery, and masking. Detection (combined with analysis) is useful to fix vulnerabilities, identify damage and define a strategy for recovery. Recovery includes all methods to get rid of the prejudicial effects of an intrusion by using some backward and/or forward recovery mechanisms. Transparently masking intrusions can be achieved by intrusion tolerance mechanisms.

Cross-cloud architectures are a promising step towards more resilient cloud-based systems. Many faults, such as an insider attack by a cloud administrator or an attacker exploiting a misconfiguration of a cloud management platform, typically affect only a single cloud provider. Cross-cloud systems can more easily handle such problems that are hard to cope with within a single cloud. A service can be partitioned over multiple clouds, causing an intrusion to affect only parts of the service, and, even better, a service can be distributed using some intrusion-tolerant replication mechanism, enabling the service to mask the intrusion.

In complement to enhancing security by masking intrusions, it is desirable to detect and react to malicious activities. Detecting dormant intrusions before they cause harm helps decreasing the risk caused by the intrusion. Accurate detection of what vulnerabilities enabled an intrusion helps enhancing the system by removing this vulnerability.

In this paper, we assume that cross-cloud partitioning and intrusion-tolerant replication are used to make some service resilient against intrusions. The main contributions of this work-in-progress paper are (1) a security analysis of a service that is used for detection and analysis of intrusions in such cross-cloud deployment, and (2) a proposed architecture for such security service that combines multiple approaches to protected itself against possible attacks and helps coping with attacks on single clouds and common-cause attacks on multiple clouds.

This paper is structured as follows: The next section describes our system model and presents a security analysis. Section 3 describes the high-level view of our architecture and explains its components. Section 4 discusses related work. Section 5 presents a short summary and future work.
2 Security model

2.1 Problem statement for this paper

A cross-cloud deployment of a service can use partitioning and intrusion-tolerant replication to mitigate attacks on a single cloud infrastructure. We want to complement such a service with an intrusion detection and analysis (IDA) service. The problems we investigate with our approach are the following:

- Can we enhance the cross-cloud service with an intrusion detection service that is able to detect common-mode intrusions in all service replicas?
- What architectural approach is adequate for this IDA service such that it is both resource-efficient and resilient against attacks on the IDA components on an individual cloud?

We approach these questions with a detailed analysis of security requirements and assumptions and with a proposal for an IDA service architecture.

2.2 System model

In our system model, we assume that we have

- multiple, independent public cloud providers. Each provider is responsible for its cloud management platform that enables a user to use resources on demand;
- a service operator (user) that is responsible for operating some intrusion-resilient service on cloud resources of potentially multiple cloud providers.

Our focus currently is on the IaaS and PaaS service models, in which the service operator can deploy its service on multiple cloud platforms. We could extend our approach to a SaaS model, in which a SaaS provider deploys some service on multiple IaaS or PaaS platforms. In that case, we assume the SaaS provider to be independent of the underlying cloud platform provider, and model the role of the SaaS provider as the service operator.

We currently base our architecture on the assumption that we have a hypervisor-based tenant isolation mechanism in place. This is commonly used for the IaaS model. In some cases, hypervisor-based isolation is also used in the PaaS model. For example Amazon’s AWS Elastic Beanstalk\(^1\) deploys services on a preconfigured platform on a dedicated virtual machine. Using other forms of isolations (such as OS containers) for PaaS is beyond the main focus of this paper, but could be considered in future work.

We further assume that there exists some abstract component responsible for detecting and analysing intrusions. The main tasks of this component is, first, the decentralized collection of security monitoring data, and, second, the centralized analysis of that data, which results in detection alerts about intrusions as well as reports with forensic details about these incidents.

2.3 Assets and Threats

Based on the distributed nature of the architecture for intrusion detection (decentralized collection of data in multiple cloud infrastructures, conceptually centralized analysis and reporting), there are two main assets that are at risk if the architecture is attacked:

- **A1 Low-level monitoring records:** The decentralized low-level monitoring records are acquired by the decentralized data collection entities.
- **A2 High-level analysis results:** The high-level analysis results are produced by the centralized analysis entity and will be used for triggering actions.

The three main classes of threats affect the confidentiality, integrity, and availability of these two assets.

In the confidentiality class, the main risk resides in the low-level monitoring data. This data contains potentially sensitive data that should not be leaked to an adversary. The high-level analysis results are derived information from this low-level data. Due to processing, aggregation and filtering, the high-level results will contain less (or no) sensitive data, and thus are less exposed to confidentiality threats.

In the integrity class, risks affect both low- and high-level data. Integrity violations may be used to either hide security incidents (i.e., produce false negatives) or to create fake alerts that cause unnecessary actions (i.e., produce false positives). Such manipulations are potentially feasible at the levels of low-level monitoring data and high-level analysis results.

In the availability class, risks arise because of the adversary suppressing data in order to hide security incidents (i.e., produce false negatives). In addition, the attacker could try to selectively suppress data in order to trigger alerts of his choice.

2.4 Attack vectors

In the following we consider these main types of attack vectors:

- **V1a** An attacker may compromise a subset of the service instances, such that the service can mask any effects of the attack;
- **V1b** An attacker may compromise all (or an excessive subset of all) service instances, such that the effect of the attack cannot be masked;
- **V2** An external attacker may compromise the local IDA service based on a previously gained control of the service replica (V1a/V1b).
- **V3a** An external attacker can compromise the cloud infrastructure, and on this basis locally attack both service instance and the IDA service.
- **V3b** A malicious insider (such as a cloud administrator) may abuse its privileges to locally attack both service instance and the IDA service.

\(^1\)https://aws.amazon.com/elasticbeanstalk/ – accessed 2017-02-28
2.5 Basic security controls

2.5.1 Confidentiality

Protecting the confidentiality of low-level monitoring data is a big challenge. We protect data in transit using secure communication channels. In all processing steps, however, the low-level monitoring data needs to be accessible and potentially exposed to an attacker.

For now, we simply assume that monitoring mechanisms will be designed such that they minimize the collection of sensitive data. We leave more advanced methods for protection of confidentiality to future work. In particular, we accept that if an attacker is able to compromise parts of the IDA service, the IDA service may remain operational while potentially sacrificing confidentiality guarantees.

2.5.2 Integrity and availability

There are several basic mechanisms that can be considered for protecting integrity and availability of low-level monitoring data as well as high-level IDA results.

The protection of low-level data needs to start at the source. Methods for trusted data acquisition are the first security controls that help protecting the integrity of low-level data. Well-known concepts for enhancing the trustworthiness of are trusted execution techniques as, for example, used in by prior work on trusted cloud computing based on the trusted platform module [11] and isolation concepts based on hardware such as Intel SGX [4] and ARM Trustzone [1].

The approach of signing the data as soon as possible after acquisition is a straightforward approach. The confidentiality of signing keys used for this operation is an essential problem to be addressed by an adequate architecture. The integrity of high-level results can also be enhanced by making the analysis process verifiable based on integrity-protected low-level data. Repeatability of the analysis can be achieved by defining a deterministic process. The availability of data can be enhanced by replication of the data as soon as possible and by employing adequate access control methods.

In the next section, we present our architecture that aims at combining multiple approaches to enhance the integrity and availability of an IDA service.

3 Architecture

A high-level view of our architecture is shown in Figure 1. A generic cross-cloud service instance (green boxes) is complemented by an intrusion detection and analysis (IDA) service (orange boxes). In our prototype we assume that the service instances are implemented as virtual machines, and the IDA service is external to that virtual machine.

Internally the architecture is composed of several low-level building blocks. The main reason for splitting the IDA block into multiple components is that we can make different security assumptions on each of the blocks and leverage different protection mechanisms. The composition of these blocks is illustrated in Figure 2.

3.1 Components

In the following, we describe each of the components and their requirements in more detail.

3.1.1 Service instance (SI)

The SI (part of a replicated or partitioned cross-cloud service) is running autonomously within a virtual machine in a cloud environment and is not aware of being monitored. The service typically has a public interface and thus, besides a public cloud management interface not shown in the figure, is the only component directly exposed to external attackers.
3.1.2 Monitor service (MS)

The MS is responsible for extracting monitoring data from the SI. MS executes on the same physical host as SI, but not within the same virtual machine. The requirements for the MS component are as follows:
- MS has a minimal attack surface exposed to the service.
- It leverages protection mechanisms against V2, V3a and V3b attacks.

The overall purpose of MS is the trustworthy and integrity-protected acquisition of monitoring data about SI. The monitoring service of each cloud is completely independent of each other instance in separate cloud environments.

3.1.3 Logging service (LS)

The LS provides a high-throughput, strongly consistent, persistent global logging service. The data stored by LS is replicated across all participating cloud environments. The requirements for the logging service are as follows:
- It is able to efficiently store large amounts of data.
- It provides strongly consistent replication of the data across nodes.
- It minimizes the possibility of common-mode attacks against all logging service replicas by minimizing the complexity of the LS code (i.e., LS will not perform complex computations).
- The global (cross-cloud) LS is resilient against malicious intrusions in a single instance (or a small subset of all instances).

The LS is the only component of the IDA for which all instances on multiple cloud environments directly interact with each other. The requirement of strong consistency simplifies the split between logging service and analysis engine (AE): As all LS instances have the same view of the data, all local AE instances can autonomously make deterministic analysis decisions based on data obtained from LS.

3.1.4 Analysis engine (AE)

The analysis engine is the most complex part of the architecture. This component may incorporate a set of well-known approaches using the collected monitoring data for detecting intrusions and delivering forensic data about detected incidents. The AE has the following requirements:
- It needs to be resource efficient, i.e., the utilization of computational resources (CPU time, main memory) should be minimized to the degree required for the expected resilience.
- Common-mode attacks exploiting vulnerabilities in the AE should affect a minimal subset of nodes.

The AE publishes its results as log records sent to the LS. Several options (outside the scope of this paper) exist how to make use of the AE output, ranging from operator notification to fully automated self-healing mechanisms.

3.2 Design and implementation

We are currently working on a prototype implementation of the described architecture that integrates several techniques to achieve strong resilience, high performance, and low resource overhead.

3.2.1 Monitor service (MS)

The basis for our prototype implementation of the monitor service is the CloudPhylactor architecture [13], which uses a dedicated monitoring virtual machine on the same physical host as the service instance. It makes use of mandatory access control of the Xen hypervisor to grant dedicated monitoring virtual machines the rights to access the main memory of other virtual machines, and thus enables secure deployment of virtual machine introspection (VMI) in cloud environments.

The monitoring service, hosted by the monitoring virtual machine, can employ VMI based on the LibVMI library for memory introspection. The MS can periodically examine that guest OS and guest applications (i.e., the service SI) for intrusions. It can also use VMI-based execution-flow tracing mechanisms to detect abnormal behaviour. MS can be protected against all attack vectors defined in Section 2.4.

Hardening against V2 attacks: The MS is a hardened component that is designed for minimizing the attack surface exposed to the service replica. As SI is monitored externally, there is no explicit interface towards MS that could be exploited by an attacker after compromising SI. In addition, MS does not perform any complex processing of the monitoring data, nor does it store large amounts of data. Instead, it is a minimalistic component that authenticates monitoring data with an internal key and forwards the data to the replicated log service (LS).

Note that this approach does not protect against V3a/V3b attacks. If the MS is compromised, the attacker can suppress or manipulate the monitoring data. After getting access to the signing key, it can even produce fake log records with genuine signatures.

Protection against V3a/V3b attacks: While being future work beyond our current prototype implementation, with additional effort, the monitoring service can be protected against V3a (external attacker controlling the local cloud management system) and V3b (malicious local administrator) attacks.

We are currently exploring two options for such protection. First, approaches from literature for building a trusted virtual machine monitor (TVM) based on trusted computing technology can be used to protect against malicious insider attacks. However, note that this approach does not protect against V3a/V3b attacks.
attacks. Second, the core part of the monitoring service could be protected with hardware-assisted isolation (such as Intel SGX-based enclaves or ARM TrustZone secure world).

3.2.2 Logging service (LS)
The LS is a distributed service replicated across all involved cloud infrastructures. Its purpose is to persistently store all monitoring data from all monitoring services, i.e., it aggregates all data from all service instances. Monitoring data is replicated with strong consistency, which is beneficial for the design of the analysis engine (see below).

**Minimalistic functionality:** Similar to the monitoring service, the logging service does not on its own perform complex analysis tasks on the data. It is thus not susceptible to vulnerabilities in complex analysis code. Such a minimalist design again contributes to hardening this component. However, due to the large data size, we do not consider the execution in trusted enclaves as a protection mechanism against malicious insiders or attackers gaining control over a local cloud infrastructure like we did for the monitoring services.

**Intrusion-tolerant replication:** We protect the integrity and availability using intrusion-tolerant replication across multiple clouds based on the BFTSMaRt library [3]. The MS acts as a client for the replicated LS. Thus, as soon as the MS has committed a monitoring log record to the LS, it has been made persistent across the cloud infrastructures.

While a simple implementation of LS in a default configuration of BFTSMaRt is a first step towards an integral prototype of our architecture, we have identified improvements that we will experimentally validate in future work:

- Latency of log operations is not a concern for our architecture. We can efficiently batch many log operations into a single BFTSMaRt request, minimizing the overhead for ensuring strong replica consistency.
- As described in the previous section, we consider using a trusted component for our MS. Such trusted component can be used to implement more efficient hybrid intrusion-tolerant replication mechanisms [14].
- Proactive recovery is a mandatory mechanism in an intrusion-tolerant replication system. Nevertheless, most practical research prototypes ignore this requirement, and retrofitting proactive recovery in an existing system is a non-trivial task. Inspired by our SPARE [5] prototype, we will consider the integration of proactive recovery in our system design.

The interface of the logging service is limited to append and retrieval operations. This design has the advantage that it constrains the damage that could result from a rogue client. With the exception of the possibility that a malicious client may append fake monitoring log records, the LS is resilient against malicious clients.

### 3.2.3 Analysis engine (AE)

The core part of our architecture is the AE. It is designed as a framework that can make use of several detection and analysis plug-ins. We make use of well-known state-of-the-art approaches for analysing malicious attacks using VMI-based monitoring. Examples for how such plug-ins can be implemented can be found in VM-Watcher [7] and VMI-Honeymon [10], among others. The analysis engine feeds back its results as new log records to the logging services.

The separation between the AE and LS enables us to use a best-effort performance optimization: The LS is subject to the usual BFT constraint that tolerating \( f \) malicious faults requires at least \( n \geq 3f + 1 \) replicas (\( n \geq 2f + 1 \) in case of optimization with a hybrid approach). In a typical deployment, our architecture spans 4 cloud infrastructures and is able to tolerate one malicious cloud. As mentioned before, the replicated logging service offers a strongly consistent view of the data. Multiple instances of the analysis engine that use a deterministic algorithm to produce analysis results will all produce the same results if working correctly. We use a static assignment such that each analysis is executed only by \( f + 1 \) nodes (i.e., 2 nodes in case of \( f = 1 \), saving half of the computational effort). If two inconsistent results are published, an additional analysis engine will also calculate the result, in order to decide which of the results was wrong.

As we are assuming an eventually synchronous timing model, in case of a missing reply (i.e., only one out of two AEs have published a result), it is impossible to decide if the other AE is faulty, or just slow in producing a result. For handling such a situation, we define a temporal threshold after which a third AE will also start calculating the same result. For distributing analysis work on AEs, we currently assume a static distribution. In future work it could be analysed if an automatic load balancing could yield better performance.

### 3.3 Discussion

For validating and evaluating the proposed architecture, we implement a proof-of-concept prototype using Xen-based private cloud infrastructures managed by OpenNebula. The benefit of this approach is that we have full control over the infrastructure and can experiment with mechanisms not available in today’s public cloud infrastructures such as Amazon EC2 or Microsoft Azure.

These limitations mainly affect the MS for two reasons. First, we make use of virtual machine introspection mechanisms not available in today’s public cloud infrastructure. Systems such as CloudPhylactor [13] demonstrate that secure virtual machine introspection can technically be made available in public cloud infrastructures. Second, we propose to make use of trusted execution mechanisms. Such mechanisms also have been suggested by prior work and can eventually offer more secure cloud deployments in future generations of public cloud infrastructures.
4 Related Work

Making use of multiple cloud infrastructures has been investigated by several previous publications. For example, DepSky [2] aims at improving the availability, integrity, and confidentiality of data stored in the cloud using replication of the data on multiple clouds, and SuperCloud [9] is a concept that offers “user-centric clouds” on top of multiple usual cloud infrastructures with enhanced security and dependability properties. The architecture that we propose complements such systems by augmenting them with a resilient intrusion detection and analysis service.

Making security services intrusion tolerant has also been the subject of previous publications, mostly focussing on firewalls. Garcia et al. propose such intrusion tolerant firewall for protecting SIEM systems [6]. Sousa et al. [12] propose a self-healing intrusion tolerant firewall architecture for protecting critical infrastructures. Kuang and Zulkernine [8] propose an intrusion-tolerant architecture for an intrusion detection system. Besides not targeting at cloud infrastructures, these publications focus on network-based intrusion detection, whereas our main focus is on host-based (or, more specifically, VMI-based) intrusion detection and analysis.

5 Conclusions and future work

In this paper, we have presented an architecture for intrusion-resilient security monitoring in multi-cloud infrastructures. Based on a security analysis that yields requirements on this security service, the proposed architecture splits this service into separate components for monitoring, logging, and analysis. We leverage and combine several existing approaches in order to enhance the resilience of this security service against malicious attacks.

This paper presented on-going work in progress. We plan to analyse security and performance aspects of our prototype in controlled lab environments (private cloud infrastructures) as well as to the extent feasible with current offers – in public clouds. We expect to obtain valuable insights into the trade-off between enhanced security properties vs. the performance and resource cost compared to widely used approaches such as monitoring data collection with Apache Kafka. By tailoring replication strategies to the specific properties and requirements of our distributed intrusion detection and analysis service, we expect to achieve performance and resource overhead close to a crash-tolerant replicated service, while being resilient against malicious faults.

Acknowledgements

This work has been supported by the “Bavarian State Ministry of Education, Science and the Arts” as part of the FORSEC research association.

References

[1] Ahmed M. Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. 2014. Hypervision Across Worlds: Real-time Kernel Protection from the ARM TrustZone Secure World. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS ’14). ACM, New York, NY, USA, 90–102.

[2] Alysson Bessani, Miguel Correia, Bruno Quaresma, Fernando André, and Paulo Sousa. 2013. DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. Trans. Storage 9, 4, Article 12 (Nov. 2013), 33 pages.

[3] Alysson Bessani, Joao Sousa, and Eduardo E. P. Alchieri. 2014. State Machine Replication for the Masses with BFT-SMART. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN ’14). IEEE Computer Society, Washington, DC, USA, 355–362.

[4] Stefan Brenner, Colin Wulf, David Goltzsche, Nico Weichbrodt, Matthias Lorenz, Christof Fetzer, Peter Pietzuch, and Rüdiger Kapitza. 2016. SecureKeeper: Confidential ZooKeeper Using Intel SGX. In Proceedings of the 17th International Middleware Conference (Middleware ’16). ACM, New York, NY, USA, Article 14, 13 pages.

[5] Tobias Dißler, Rüdiger Kapitza, Ivan Popov, Hans P. Reiser, and Wolfgang Schr der Peikschat. 2011. SPARE: Replicas on Hold. In Proc. of the 18th Network and Distributed System Security Symposium (NDSS ’11). 407–420.

[6] Miguel Garcia, Nuno Neves, and Alysson Bessani. 2013. An intrusion-tolerant firewall design for protecting SIEM systems. In 2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W). 1–7.

[7] Xuxian Jiang, Xinyuan Wang, and Dongyan Xu. 2007. Stealthy Malware Detection Through Vmm-based “Out-of-the-box” Semantic View Reconstruction. In Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS ’07). 128–138.

[8] Liwei Kuang and Mohammad Zulkernine. 2008. An Intrusion-Tolerant Mechanism for Intrusion Detection Systems. In 2008 Third International Conference on Availability, Reliability and Security. 319–326.

[9] Marc Lacoste, Markus Miettinen, Nuno Neves, Fernando M. V. Ramos, Marko Vukolic, Fabien Charmet, Reda Yaich, Krzysztof Obozynski, Gitesh Verneckar, and Paulo Sousa. 2016. User-Centric Security and Dependability in the Clouds-of-Clouds. IEEE Cloud Computing 3, 5 (Sept 2016), 64–75.

[10] Tamas K. Lengyel, Justin Neumann, Steve Maresca, Bryan D. Payne, and Aggelos Kiayias. 2012. Virtual Machine Introspection in a Hybrid Honeypot Architecture. In 5th Workshop on Cyber Security Experimentation and Test, CSET ’12, Bellevue, WA, USA, August 6, 2012.

[11] Nuno Santos, Krishna P. Gummadi, and Rodrigo Rodrigues. 2009. Towards Trusted Cloud Computing. In Proceedings of the 2009 Conference on Hot Topics in Cloud Computing (HotCloud’09). USENIX Association, Berkeley, CA, USA, Article 3.

[12] Paulo Sousa, Alysson N. Bessani, Wagner S. Dantas, Fábio Souto, Miguel Correia, and Nuno F. Neves. 2009. Intrusion-tolerant self-healing devices for critical infrastructure protection. In 2009 IEEE/IFIP International Conference on Dependable Systems Networks. 217–222.

[13] Benjamin Taubmann, Noelle Rakotondravony, and Hans P. Reiser. 2016. CloudPhylacter: Harnessing Mandatory Access Control for Virtual Machine Introspection in Cloud Data Centers. In The 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-16).

[14] Giuliana Santos Veronese, Miguel Correia, Alysson Neves Bessani, and Lau Cheuk Lung. 2009. Spin One’s Wheels? Byzantine Fault Tolerance with a Spinning Primary. In 28th IEEE Symposium on Reliable Distributed Systems (SRDS). 2009. 135–144.

3https://kafka.apache.org – accessed 2017-03-07