Security in the Cloud Based Systems: Structure and Breaches

Vivek Shandilya
Department of Computer Science
University of Memphis
Memphis, Tennessee 38152
Email: vmshndly@memphis.edu

Sajjan Shiva
Department of Computer Science
University of Memphis
Memphis, Tennessee 38152
Email: sshiva@memphis.edu

***Abstract*—Cloud based systems(CBSs) are increasing in the computing world. These systems derive their complexity due to both the disparate components and the diverse stake holders involved in them. The component wise security alone does not solve the problem of securing CBSs, but the stakeholder's computational space spanning across many components of the CBS, needs to be secured too. There have been initial attempts to model the security of the cloud in terms of securing stakeholder's computational space. Some recent attempts formally model the CBS as modularized actor models, using rewriting & equational logic based modeling languages. Building on these works we present a framework for building executable models of CBSs for security analyses. We illustrate its validity showing how the recent security breaches and security solutions can be modeled and analyzed using this framework.**

I. Introduction

In recent times, the increased popularity of the cloud computing technology is due to the convenience and affordability among other reasons. But Security and privacy are the important concerns associated with it. The main reasons for these concerns are not only the disparate components enabling the cloud technology but also the different stakeholders involved in it. This is due to the very nature of the technology as given by the National Institute of Standards and Technology (NIST)'s definition [9]: *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

The end users who use the services on the cloud eventually build a complex system involving their own local systems including portable and mobile devices, local area networks in their offices/homes, the Internet service which enables them to connect to the systems of one or more cloud service providers and the third party software and web services running remotely to enable their whole operation. These different components make up the cloud based system (CBS) used by the end users. Thus, each stake holder has access to some parts of this cloud based system and has a set of privileges allowing some actions and corresponding responsibilities in maintaining the security of the overall system. Though there are many stakeholders we here focus on the role of the consumer who buys the cloud based services and builds his own CBS. A consumer constructs *a computational space*, with his cloud based system and operates over it. This space can be viewed as a graph with nodes having some data and tasks associated with them. This computational space can be modeled to verify if the space conforms to the needed security constraints both by design and while operation by keeping track of processes. There are attempts to model the CBSs in [10] & [6]. Building on these works, here we present the framework for modeling the CBSs to analyze for the properties to specify security breaches and the solutions.

Using this framework, we can specify the requirement property we want, to identify the attempt for security breaches as violation of a monitorable property of this model. Thus we show that as [6] employed with DDOS, the attempts for security breach can not only be analyzed using PVeSTa , we can eventually build runtime monitors[19], by devising the suitable instrumentation over the parameters which can be accessed. Unlike in [15], where the instrumentation checks exact states and thus needing access to the source code, by only keeping track of the important accessible parameters in the processes, as per their statistical behavior sampled randomly, a more practical instrumentation can be sufficient to build monitors to achieve the statistical monitoring. Similarly we present that the security solutions also can be modeled as constraints operating to preserve the security described as requirement properties.

The major contributions of this work are summarized as below.

1) We present a framework to model CBSs to design an executable model of CBSs by devising a procedure to specify requirement properties to preserve the security of the CBSs. Further the procedure derives the monitorable properties based on the requirement properties to design run time monitors to detect the attempts for violation.
2) We present the list of the prominent security breaches and security solutions of the last year to show the security properties that are violated & preserved respectively to motivate the applicability of the procedure.

II. Specification Framework

A CBS can be considered as a set of computational entities(including storage), each considered as a node. The graph we obtain will including the communication path between these nodes as edges. Each node is associated with data and tasks on it.

A. Stakeholder's Computational Space

In this section we present a formal framework of CBSs based on the model in [14]. Here we present a time invariant model of the computational space of the stakeholders. That is, the stakeholder's choice of actions at an access area/node, is same irrespective of when she enters the area/node. Let the CBS involves n, where $0 < n < \infty$, different stakeholders. The $i^{th}, 1 \leq i \leq n$ stake holder h_i has an authorized access to a part of the system. This is referred to as the *access space* s_{i_A} of the stakeholder h_i. In each of these areas, the stake holder h_i can choose to do one action, at a time, out of the clearly defined *set of actions*, $s_{i_{a_j}}$ corresponding to j^{th} node. The ordered pair of this *access area* and the corresponding *set of action* is referred to as the activity area. $s_{i_{\alpha_j}} = (s_{i_{A_j}}, s_{i_{a_j}})$. The union of all such sets of actions available at, that is, corresponding to, each of the m areas/nodes in the access space is referred to as the stake holder's action space. s_{i_a} . $s_{i_a} = \bigcup_{j=1}^{m} s_{i_{a_j}}$. The union of all nodes/access areas is the access space of the stake holder h_i, that is, $s_{i_A} = \bigcup_{j=1}^{m} s_{i_{A_j}}$. The union of all the activity areas of the stake holder h_i, is called the activity space of h_i. $s_{i_\alpha} = \bigcup_{j=1}^{m} s_{i_{\alpha_j}}$. The sequence of all the activity spaces of the stake holders is referred to as the activity profile S_α of the CBS, as in $S_\alpha = (s_{1_\alpha}, s_{2_\alpha}, \ldots, s_{n_\alpha})$ The security measures each stake holder has to take must be able to accomplish with some combination of the legal actions that are available to him. It may involve one or more actions in tandem to be taken in each access area.

The security measures in the access area $s_{i_{A_j}}$ the stake holder h_i has to take be the set $r_{i_{A_j}}$, such that $r_{i_{A_j}} \subseteq s_{i_{a_j}}$. Thus, the security activity a stake holder does in the access area $s_{i_{A_j}}$ is given by the tuple r_{i_α},
such that $r_{i_{\alpha_j}} = (r_{i_{A_j}}, r_{i_{a_j}})$. Any practical security measure generally involves a sequence of actions across many activity areas to be effective. And the total security activity of the stake holder h_i is given by r_{i_A}, such that $r_{i_A} = \bigcup_{j=1}^{m} r_{i_{\alpha_j}}$. The sequence of all the security activity spaces of the stake holders is referred to as the security activity profile S_α of the cloud, that is, $R_\alpha = (r_{1_\alpha}, r_{2_\alpha}, \ldots, r_{n_\alpha})$. A security measure by a stakeholder is an 'a priori' plan of security actions at each access area. We distinguish security measure from the defense measure which happens, as a series of actions, while defending a system against a malfunction either due to an internal system fault or a malicious attack. For a given node/access area, the activity profile would be

$$S_{\alpha_j} = (s_{1_{\alpha_j}}, s_{2_{\alpha_j}}, s_{3_{\alpha_j}}, \ldots, s_{n_{\alpha_j}}). \tag{1}$$

More than one stakeholder may have access to the data on a given node. To make the data secure, the activity profile for that given node must be prioritized, that is, sorted as per the privileges. And it must be ensured that each of the stakeholder who will alter the data must do so only with out negating any stake holder before him in the sequence. For notational simplicity let us take the equation 1 as the ordered sequence as per the privileges.

B. Executable Formal Model

The model of the CBS must have the following features.

1) The model must facilitate expressing concurrent computation in a distributed system.
2) The components must be able to function as message driven, depending on asynchronous messaging between them, as in a client server paradigm, the CBS operate with components providing computational resources as a service to each other.
3) The model must express precisely the states of the components such that the analyses for the interesting properties could be made.
4) The properties thus obtained must be sufficient to completely express (account for) the security of the CBS.
5) The model must be able to precisely able to express the operations in CBS such that an analysis can be done using statistical model checker such as PVeSTA.

1) Modularized Actor Model: We here describe the Modularized Actor Model based on [8], of the computational space which a stakeholder has in a CBSs.The Actor Model is a model of computation with "Actors" as the universal primitives. An Actor is a computational entity. It can respond to a message it receives from other Actors by

1) send messages to other Actors;
2) create new Actors;
3) designate how to handle the next message it receives.

Here we note that each actor will have a state. There is no global state with which the actors have to synchronize. The sequence of states of each actor forms the execution associated with that computational entity modeled as an actor. Then the model is analyzed for properties as described in the sectionIII.Here the modularity comes by the fact that once a message is sent then it is the responsibility of the receiver. This asynchronous behavior helps formulate the modularity in the model.The case studies in the sectionIV provide illustration for the application of the framework and the analyses of properties.

III. Analyses of the model for Properties

We here present the analyses framework for the model described in II-B for properties. As given in the section II-A each stakeholder has a corresponding computational space in the CBS. Each computational space has many nodes. Each node has data and tasks associated with it. Each node is modeled as an Actor. Each node has a state. The sequence of states of each node is an execution. A set of executions is a property. There will be two types of properties associated with the executions; liveness and the safety properties. With a conjunction of these two types of properties we can specify any requirement property for the executions. Thus when the requirement property to cover CIA for the node is specified, then it can be taken as the sufficient condition for the security of the node. The CIA properties specified for the node can be verified by using the statistical model checker PVeSTa . Modeling the CBS as a Modularized Actor model, following the procedure in [6] facilitates this.

We here want to specify the security as CIA properties. The data at each node is used to keep track of information through out the CBS. We here take the CBS to be secure if at each node, CIA properties are preserved. We consider that the access to data (including the meta-data) would lead to information in that data. And any requirement property that we want to specify to preserve a specific security concern of the CBS, should be done by decomposing it in terms of the CIA properties. And we know from [12] that any such property can be adequately specified as a conjunction of liveness and safety property.

A. Security as CIA properties

We consider that sufficient condition for the CBS to be secure would be to have CIA properties preserved at each of the nodes making up the CBS. The nodes considered in the previous sections are associated with data stored and tasks running on them. The tasks running on a node can be operating on the data stored on the local node or on any other node. In either case, the tasks can be considered to be running to alter data on some node. If that node is not in the computational space of stakeholder (user) we can be sure that his confidentiality is not compromised through the data being operated by the tasks. But we clarify that we define information to be related to not only data but also meta-data and also data related to the configuration of the node, like its processing platform, etc. Thus ensuring the security of the data and ensuring that the tasks are taking safe paths will ensure the security of the node. By ensuring the security properties of each node, the security of the CBS of the stakeholder is ensured. When each space of the stakeholder is secure then the whole CBS is secure.

1) Confidentiality Property: Unauthorized discloser of information would lead to the violation of the confidentiality property. The data associated with the node, will have the usual three actions associated with it. Given that both static data and the programs are considered data here, the read, write & execute actions would be associated with them. In this case, given that the order in equation 1 is preserved with read action such that the sequence is ordered as per the privileges of stakeholders on the data, and only those who have the privilege to be able to access the data should get to read it. That is, this gives rise to three sequences of stakeholders. First will have access to read, while the second can only write and third can only execute. With the order in these sequences be preserved to ensure the stakeholders with only read privileges can read the data, the confidentiality property is preserved. Thus this property is a set of three sequences of privileges of data on a given node, which are preserved during the execution of tasks.

2) Integrity Property: Unauthorized modification of information would lead to the violation of the integrity property.When this order is preserved during an execution, that is a sequence of states of the node, which represents many actions over the data on the node, the integrity of the data is preserved. Given the data and the tasks associated with the node, we define the state of node. That is $s_j(node) = (s_j(data), s_j(tasks))$

3) Availability Property: Unauthorized withholding of information would lead to the violation of the availability property. The availability property is to ensure, during the execution, if the privilege order in the above mentioned stipulation is satisfied, the tasks associated with the users must get executed, that is, the task must be successfully completed, by accessing the necessary data. Then the availability property is satisfied.

B. Specification of the CIA Properties as a conjunction of Safety and Liveness Properties

Formally execution is defined as an infinite sequence of states. By repeating the terminating state in the paths obtained we construct the infinite sequences representing the execution. Such an infinite sequence of process states, starting with an initial state and each successive states $s_i, 0 < i < \infty$ being a valid state, that is $s_i \in S, 0 \leq i < \infty$, representing an execution is given by $\sigma = (s_0, s_1, s_2, \ldots)$.

It should be noted that the order of the states in the execution need not follow the subscript order of nodes obtained from the program, as it is only decided by the flow of execution.Thus all the executions σ_i, $0 < i < \infty$ are obtained. By dropping the initial elements in the sequence of these executions, many infinite sequences are created which represent a later part of an execution in the process. That is by dropping an initial state in the execution above, which may be called σ_1, another σ_{1_1} is created, as given by $\sigma_{1_1} = (s_1, s_2, \ldots)$. Now the infinite set of all infinite sequences of all permutations of s_i, $s_i \in S$ (*which also contains all the executions and the later parts of executions like in equation*), is denoted by S^ω is constructed, as given by $S^\omega = \{(s_1, s_1, s_1, \ldots), (s_2, s_2, s_2, \ldots), \ldots,$

$$\sigma_1, \sigma_{1_1}, \ldots, \sigma_2, \sigma_3, \ldots\} \qquad (2)$$

Property is defined as a set of executions.

$P = \{\sigma_1, \sigma_2, \sigma_3, \ldots\}$.

A property $P \subseteq S^\omega$ is a safety property P_{safe} if for every sequence $\sigma \in P$,for every prefix of σ, there exists a sequence $\beta \in S^\omega$ which when appended will make the resulting sequence to be a member of the set P. This means in essence that the sequences σs contain as each element states only from S. The executions belonging to the safety property are called safe executions. If σ_1, σ_2, σ_3 and σ_4 are safe executions, then $P_{safe} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \ldots\}$.

Prefix of an execution is an initial subsequence of the execution and it represents the execution of the process from the beginning up to some finite number of states. We define the set $pref(\sigma_i)$ as a set of all the finite prefixes of the execution σ_i. For example consider the execution σ_1 and the set of all its finite prefixes, denoted as $pref_(\sigma_1)$, given by

$pref(\sigma_1) = \{(s_0), (s_0, s_1), (s_0, s_1, s_2), \ldots\}$. The union of all the sets of prefixes of each executions in P_{safe}, is given by $pref(P_{safe})$. During the union operation the elements which are common are collapsed into a single element, as there can not be any repetition of members in the set. That is, the first element in the sets $pref(\sigma_1), pref(\sigma_2), pref(\sigma_3)$ and $pref(\sigma_4)$, which is the finite prefix of length 1 of $\sigma_1, \sigma_2, \sigma_3$ and σ_4 is the same, which is (s_0). But during the union there will be only one element (s_0) in $pref(P_{safe})$.Thus we have, $pref(P_{safe}) = \cup_{\sigma \in P_{safe}} pref(\sigma)$.

1) Safety Properties: The safety properties of the CBS amounts to the runtime properties which ensure that the system has *bolt-on* security.A safety property P_{safe} is a monitorable property $P_{monitorable}$, when $S^{\star}/pref(P_{safe})$ is Turing recognizable. That is after examining a given execution, if we are able to determine in finite steps, if that execution is not in the set P_{safe} then it is a monitorable property. Thus from the safety property we construct a monitorable property as given by,$P_{monitor} = \{\sigma_1, \sigma_2, \ldots\}$. Once the above set, monitorable property is identified, we have to get a more useful definition of this set to be able to build a monitor for it.

2) Liveness Properties: The liveness properties of the CBS amounts to the design properties which ensure that the system has *built-in* security. Given a prefix if a sequence certainly land in a prescribed state then such a set of sequences corresponds to the liveness propertyP_{live}.

Given the properties of CIA at a node, each of these requirement properties can be completely expressed as a conjunction of a liveness and safety property [12]. That is, $P_{CIA_i} = P_{safe_i} \& P_{live_i}$.

C. Model Checking

The model the CBS obtained here can be checked using statistical model checker like PVeSTa, the parallelized implementation of VeSTa . PVeSTa does statistical verification of properties expressed in many logics against probabilistic real-time models specified as probabilistic rewrite theories in Maude or continuous or discrete Markov Chains. The logics that are handled are Probabilistic Computational Tree Logic (PCTL), Continuous Stochastic Logic (CSL) or the Quantitative Temporal Expression language (QuaTEx). PVeSTa is implemented in Java. It consists of mainly two programs. A server program which implements the computing resource being modeled-checked, to generates the random samples of output by discrete-event simulations of a given probabilistic model. The client program collects these samples from the server program and does the statistical verifications, by hypothesis testing for the logic formulas and and confidence interval computations for QuaTEx expressions.The procedure to check/verify the model would follow along the same lines as in [10], [6]. In our work here, we only present the framework for the model and show how to analyze the model for specific security properties but do not present the verification.

IV. Application of the Model

The framework here presented can be used to model the CBSs to analyze for the requirement properties such that, the security breaches would be caught as the violation of these properties by an execution. The anomalies can be defined to be the property violations, and thus by ensuring the property to be monitorable, a runtime monitor can be built to catch the violations.

A. Security Breaches

In the past year, there have been many instances of security breaches of the CBS. Considering such total incidents, a list (ordered on the severity of damage), of most damaging breaches is published [7]. That ordered list contained *Data breaches, Data loss, Account hijacking, Insecure APIs, DoS, Malicious Insiders, Abuse of Cloud Service, Insufficient due diligence, Shared technology issues*.These *security violations* could be formulated as the following *Property violations*.

- The first two security breaches involve directly the data stored by a stakeholder being lost for good or being altered in ways it was not supposed to. In both the cases the data on a node/set of nodes are altered by tasks that violate the *Integrity Property*.
- In the account hijacking, we see that all the privileges of the stakeholder is taken by some other person/entity while the original authority might/might not have lost his privileges. Thus in case of loosing the privileges by the original stakeholder, it amounts to the violation of *confidentiality property*. If the original stakeholder is not denied of his privileges, then also it would amount to the violation of *availability property*.
- The APIs are used by the consumer to manage the provisioning, management, orchestration and monitoring of the cloud services. The security breaches due to the insecure APIs have been the fourth biggest cause for the security breach of CBSs. This can result in the loss of access to services provided by the cloud service provider. As the upper edifice depends on these APIs, this amounts to the violation of *availability property*. [1]
- The most prominent type of DoS attack is at the network layer, where many users share the same channel for transmitting the data to and from the cloud. This kind of attack can be modeled as *shared channel attacker model*. The adaptive selective verification protocol is useful to define the action of the DoS attacker doing shared channel attack [2].
  The main idea is to keep track of the *time interval between the service request by the client and getting service*. This amounts to violation of *Availability Property*.
- The sixth highest damage has been due to malicious insiders. CERT [4] defines this threat. [3]. It is similar to the identity theft, as the attacker actually passes through the authentication while doing the malicious activity. Here the attacker's privileges would be that of a cloud service provider's crew rather than that of a customer.
- *Abuse of Cloud Service*, as a security breach highlights our framework's modeling strength in its scope of including each of the stakeholder and their activities. This is concern for the cloud service provider. Given a cloud service, an authorized user who has some privileges may execute the available actions to achieve a result which is not legal/ethical/authorized

[1]Here the violation of *the availability property* has to be defined broadly to cover not only the data associated with the nodes, but also the tasks too. The curtailing of the availability of APIs results in the reduction of action set of the affected stakeholder.

[2]Here we can see that the requirement property is *the sequence of states*, with states (having time stamps) indicating client "Requests" & server "Responses"as in [6].

[3]A malicious insider threat to an organization is a current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems.

either inside the cloud or beyond the cloud intending to damage/harm other stakeholders with malicious intentions to breach any or all of *CIA* properties.[4]

- *Insufficient due diligence* plays a critical role with many businesses which offer their services using the cloud based services by other CSPs. While doing so, there have been many security breaches, for the end user. There can be incompatibilities between the contractual obligations between the cloud service provider and the businesses and that of the businesses and the end user. This has led to many security breaches. While exposing the confidential data out on cloud, this has amounted to the violation of the *confidentiality property*. By breaking some services and processes from operating correctly violation of *availability* property is also caused.
- Shared technology issues CBS built using *IaaS, PaaS, SaaS*, offered by the disparate stakeholders can lead to vulnerabilities due to configuration issues. For example, when a Unix based cloud is used for infrastructure and a Mac OS is installed on a virtual machine thus obtained, which in turn runs an MS office over it, there are chances that the technologies may have stability issues. The distinctions in the file systems can also create problems. Usually these can lead to both malfunctioning of programs and also data loss, thus amounting to violations of both *integrity and availability* properties.

B. Security Solutions

Many prominent security solutions are offered as products and services to the cloud customer. These solutions are designed to provide security *in the cloud, for the cloud and from the cloud*. The nodes/accesses areas of the customer of CBS are constantly having data and processes on them. By securing these, that is, in turn, there is an attempt to provide security. Here are some prominent security solutions offered commercially.

- *Identity & Access Management*: These solutions are provided by companies/services such as SyferLock Technology Corporation, [17], Symplified & PingI dentity. These services would attempt to reduce the identity theft. Thus they would be ensuring that *confidentiality property* is preserved.
- *Key Management* This solution is provided by companies/services such as Securosis, L.L.C. [13]. This ensures the authentication is done with more reliability.Thus they would be ensuring that *confidentiality property* is preserved.
- *Data Encryption*: This solution is provided by companies such as TrendMicro Inc [18]. These solutions would provide processes to ensure unauthorized persons/programs cannot get the confidential information while the data is being encrypted using safe encryption techniques. Thus they would be ensuring that *confidentiality property* is preserved.
- *Network Firewall*: This solution is provided by companies such as StilSecure Inc [16]. This solution offers processes which intervene in the network traffic between the nodes/ access areas in CBS. They will eventually get to block the malicious flow in the shared channels that can create DoS attack. Thus they preserve *availability* property.

[4]A concrete example would be the compromising the authentication server of the Dropbox [5] on 20,June 2011 for 4 hours while the attackers could access data of other users, breaching *confidentiality property*.

V. Related Work

The works [10] & [6] have attempted to model the CBSs. They concentrate on modeling CBSs for analyzing the management and to check for the design flaws leading to security risks respectively. They consider CBS as a model to offer computational resources as services over networks. For analyses they specify the services of CBSs in D-KLAIM a dialect of the co-ordination language KLAIM [11] and build the model. Then they check the model using Maude system [3]. Since this kind of modeling is suitable for exact model checking, it can be effective only for small systems. For larger systems, due to the explosion of states, using the dialects of KLAIM to model and performing exact model checking, is found to be not scalable. Hence, Actor Model of Computation [8] is used to model the CBSs and a statistical model checker, PVeSTa [1] is used to verify them. The main idea in their case study of DDOS attack scenario is to create Adaptive-Selective-Verification wrapper meta objects for the client and server sides and describe it as modularized actor model in the Maude language and analyze using PVeSTa, the statistical model checker.

VI. Conclusion

We present the formal framework for constructing the executable models of CBS, useful for analyzing the security properties. The properties of model can be used to either verify using statistical model checkers or more practically to build run-time monitors to catch anomalies. We illustrate our framework's utility by modeling the prominent security breaches and solutions of last year. It is very crucial to precisely formulate the security properties that are to be preserved by the CBS, which will be verified against the model of the CBS. But we observe that building a model for a complex CBS manually is not feasible. Since we define the CBS as a union of computational spaces corresponding to stakeholders, such computational spaces are successfully modeled as Markov Decision Processes. Automating the abstraction for incrementally generating of models for such systems is established in [2].

Using these procedures developing an executable model of an enterprise size system to analyze and verify for security properties is an interesting future work. That would facilitate dealing with attacks like compromises due to cross-virtual machine attacks and attacks across different software/middleware/hardware entities as the model is not limited by component-security based methodologies to secure the CBS.

References

[1] M. AlTurki and J. Meseguer. PVeStA: A Parallel Statistical Model Checking and Quantitative Analysis Tool. In A. Corradini, B. Klin, and C. Cîrstea, editors, *Algebra and Coalgebra in Computer Science*, volume 6859 of *Lecture Notes in Computer Science*, pages 386–392. Springer Berlin Heidelberg, 2011.

[2] R. Chada and M. Viswanathan. A counterexample-guided abstraction-refinement for Markov Decision Processes. *ACM Transaction on Computational Logic (TOCL)*, 2010.

[3] M. Clavel, F. Duran, S. Eker, P. Lincoln, et al. *Maude Manuel*. Menlo Park, CA 94025, USA, 2011.

[4] Dawn M. Cappelli and Randall F. Trzeciak. Best Practices For Mitigating Insider Threat: Lessons Learned From 250 Cases. *RSA conference*, April 2009.

[5] Dropbox Inc, Report. Dropbox Authentication Server Compromised. June 2011.

[6] J. Eckhardt. Security Analysis in Cloud Computing using Rewriting Logic . Master's thesis, Institut Fur Informatik, 2012.

[7] T. T. W. Group. The Notorious Nine. Technical report, 2013.

[8] C. Hewitt. Actor Model of Computation: Scalable Robust Information Systems . *Inconsistency Robustness*, Aug. 2011.

[9] P. Mell and T. Grance. The NIST Definition of Cloud Computing. Technical Report 800-145, Computer Security Division, Information Technology Laboratory , NIST, U.S. Department of Commerce, 2011.

[10] T. J. Muhlbauer. Formal Specification and Analysis of Cloud Computing Management. Master's thesis, Institut Fur Informatik, 2012.

[11] R. D. Nicola, G. L. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility . *IEEE Transactions on Software Enginering*, 24(5), May 1998.

[12] F. B. Schneider. Decomposing properties into safety and liveliness using predicate logic. Technical Report 87-874, Cornell University, Oct. 1987.

[13] Securosis, L.L.C. Understanding and Selecting a Key Management Solution. *Technical Report*, Feb 2013.

[14] V. Shandilya and S. Shiva. Security in the Cloud: A stake holder's perspective. In *SAM*, July 2012.

[15] S. Shiva, R. Dharam, and V. Shandilya. Runtime monitors as sensors of security systems. *IASTED, Dallas*, Dec. 2011.

[16] StilSecure Inc. Safe Access Overview. *Technical Report*, 2012.

[17] Syferlock Technology Corporation. Understanding and Selecting a Key Management Solution. *System and Method U.S. Patent nos. 7,143440 7,725,712*, 2013.

[18] Trendmicro Incorporation. Cloud Security Survey Global Executive Summary. *Technical Report*, 2012.

[19] M. Viswanathan. *Foundation for the run-time analysis of software systems*. PhD thesis, Computer and Information Science, 2000.