Defining the Cloud Battlefield
Supporting Security Assessments by Cloud Customers

Sören Bleikertz*
IBM Research - Zurich
sbl@zurich.ibm.com

Toni Mastelić*
Vienna University of Technology
toni@infosys.tuwien.ac.at

Sebastian Pape*
TU Dortmund
sebastian.pape@cs.tu-dortmund.de

Wolter Pieters
TU Delft / University of Twente
w.pieters@tudelft.nl

Trajce Dimkov
Deloitte LLP
tdimkov@deloitte.nl

***Abstract*—Cloud computing is becoming more and more popular, but security concerns overshadow its technical and economic benefits. In particular, insider attacks and malicious insiders are considered as one of the major threats and risks in cloud computing. As physical boundaries disappear and a variety of parties are involved in cloud services, it is becoming harder to define a security perimeter that divides insiders from outsiders, therefore making security assessments by cloud customers more difficult.**

In this paper, we propose a model that combines a comprehensive system model of infrastructure clouds with a security model that captures security requirements of cloud customers as well as characteristics of attackers. This combination provides a powerful tool for systematically analyzing attacks in cloud environments, supporting cloud customers in their security assessment by providing a better understanding of existing attacks and threats. Furthermore, we use the model to construct "what-if" scenarios that could possible lead to new attacks and to raise concerns about unknown threats among cloud customers.

I. Introduction

A. Background: Security Concerns in Cloud Computing

Cloud computing has gained remarkable popularity in recent years due to the economic and technical advantages of this new way of delivering computing resources. Customers benefit from rapid provisioning and seemingly infinite scalability, while only being charged on a pay-per-use basis.

Although the benefits of cloud computing are evident and users demand cloud services, security is a major inhibitor [1]. An analysis of risks and threats in cloud computing has been conducted in [2] and [3]. In particular, both reports agree that insider attacks and malicious insiders are a major technical risk and among the top 10 threats. The risk is amplified due to the disappearance of physical boundaries that makes it very challenging to define a security perimeter that divides insiders from outsiders [4], [5].

Due to the variety of parties involved in a cloud service, cloud customers face difficulties in assessing the risks and threats of insider attacks in cloud services. To illustrate this point, let us consider the following attack scenarios: A malicious cloud administrator can steal information that are stored or processed in a virtual machine of a cloud customer [6]. Furthermore, a malicious cloud customer can perform a similar attack on other customers that share the same physical resources [7]. Malicious behavior is not always required to constitute a risk to cloud customers. The outage of Amazon EC2 in 2011 [8] impacted the availability of the cloud service and was caused by an honest fault of a cloud administrator. Similar, honest faults by cloud customers can also impact other customers as demonstrated in [9], where a SSH public key for the administrator account was accidentally left in an image and which constituted a backdoor.

These scenarios cover only a small set of the involved parties (i.e., only cloud administrator and customer) and just two different characteristics of the attacker (i.e., honest faults and malicious). However it shows that the general misconception of either trusting the cloud or not does not hold, but more fine-grained trust and attacker models are required. We need to systematically specify the parties, their capabilities and motivations, in order to obtain a complete picture and support cloud customers in their risk and threat assessments.

B. Research goal: Supporting Security Assessment of Infrastructure Clouds

In this paper we propose a high-level model that supports cloud customers in their security assessments of clouds. Since the security of a cloud strongly depends on the used infrastructure, our framework combines a system model of infrastructure clouds, including the involved entities and system components, with a security model that describes security objectives of cloud customers, attacker characteristics, and threats. The framework allows for systematic analysis of the security threats in a specific cloud service environment. By comparing the analysis across cloud providers, decisions on provider choice are supported.

The main challenges involved are related to reaching the appropriate level of abstraction. In theory, many different entities could be distinguished in the model, but this comes at the cost of increased complexity. As the model is meant to be used by cloud customers, understandability and usability

* These authors have contributed equally and are ordered alphabetically by their surnames.

are important requirements. In addition, it turns out that several unique features are essential in modeling existing attack scenarios, for instance access intervals, i.e., when an entity can access certain resources. Finding a combination of expressiveness and understandability is therefore key.

C. Methodology: Designing an IaaS Threat Model

Our model focuses on infrastructure clouds (IaaS) as the most generic and standardized abstraction layer [10]. In many cases the layers build upon each other, therefore a model of IaaS also partly covers attack scenarios of Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS).

We develop our system model starting from entities, system components, and their relation in terms of access levels for infrastructure clouds. We consider further entities besides the cloud provider and the customer, such as hardware manufacturers. Thereby our model is able to cover an extended set of possible attacks, for example also hardware trojans [11].

As the model is meant to support security assessment by cloud customers, the security objectives in our security model are defined from a customer's point of view. For now, we focus on confidentiality, integrity, and availability *(CIA)* with regard to compute, storage, and network resources provided by an infrastructure cloud provider. Unlike previous work, we differentiate between different characteristics and motivations of attackers in our security model, ranging from a *malicious* attacker to a *stepping stone* one, who unknowingly contributes to an attack. This allows us to assess whether implemented measures match the expected type of attackers. The combination of (a) our system model, (b) the security objectives of cloud customers, and (c) our attacker model forms the basis of our threat model, which can be used to analyze and identify attack scenarios.

For the evaluation of our model, we mapped existing practical attacks in cloud environments to the model by identifying the involved entities, their attacker characteristics, and threats. We performed several iterations of model development and evaluation: After each iteration we improved the model based on our findings when mapping the attacks.

For a systematic analysis of threats in cloud environments we propose a variation strategy inspired by the HAZOP approach [12]: First we form the foundation of our analysis by identifying known attacks and mapping them to the model. Second, we analyze remaining combinations of entities, attackers' characteristics, and threats in order to reveal possible unknown attacks. Due to space restrictions we only give derivations of a subset of possible new attack scenarios in this paper.

D. Our Contributions

In summary, we make the following contributions:

- We propose a comprehensive system model of infrastructure clouds.
- Our security model defines security objectives, as well as a set of archetypes that can capture a wide range of characteristics and motivations of an attacker.
- The combination of our security model and system model provides a powerful tool for systematically analyzing existing attacks in cloud environment. We demonstrate this by a set of known attack scenarios.
- Finally, our model can be used for deriving new security threats from existing scenarios, as well as describing and analyzing new what-if scenarios by changing characteristics of involved parties.

II. Related Work

In comparison to existing work by Abbadi et al. [43], our model is more comprehensive for infrastructure clouds due to our focus on this abstraction layer, while their model is more abstract and covers also other layers, i.e., SaaS. In [44] Grobauer and Walloschek focus on risk assessment and vulnerabilities of technologies used in a cloud environment. They correlate these vulnerabilities to essential cloud characteristics and to system components like computational resources and storage. However, they do not consider involved parties and their relations to system components, nor do they try to provide a model for mapping these vulnerabilities to attack scenarios. Similarly, Garfinkel and Rosenblum [45] discuss security problems at the virtualization layer that now forms an integral part in infrastructure clouds. However, their work predates the cloud computing paradigm and does not discuss such security issues in a larger scenario that also considers the variety of entities and possible attackers found in infrastructure clouds. Behl [46] addresses the most common and critical security issues in cloud computing and provides key research challenges in this field. Although his work covers insider and outsider attack scenarios, they are discussed as separate use-cases, while no model is provided to correlate and describe them. A survey of threats in cloud environment is presented by Molnar and Schechter [47], although they do not claim to provide a necessarily complete set of threats and the authors expect that new threats will be identified. We believe that our model can be used to identify and contribute new threats due to our systematic approach, as well as to provide a categorization of existing ones.

III. System Model

Cloud computing can be implemented on different abstraction layers ranging from the lowest level of Infrastructure-as-a-Service *(IaaS)* to the highest abstraction of Software-as-a-Service *(SaaS)*. Developing a generic threat model covering all the abstraction layers is hard, since we have to deal with an increasing diversification on the higher abstraction layers. For example, both Google GMail and Salesforce CRM are considered instances of SaaS, but with different and application-specific attacker models. Therefore, we define a model of a cloud environment on a IaaS layer consisting of entities and the system components as shown on Figure 1.

A. Entities

Entities represent subjects which are involved in a cloud service, directly or indirectly, while components represent

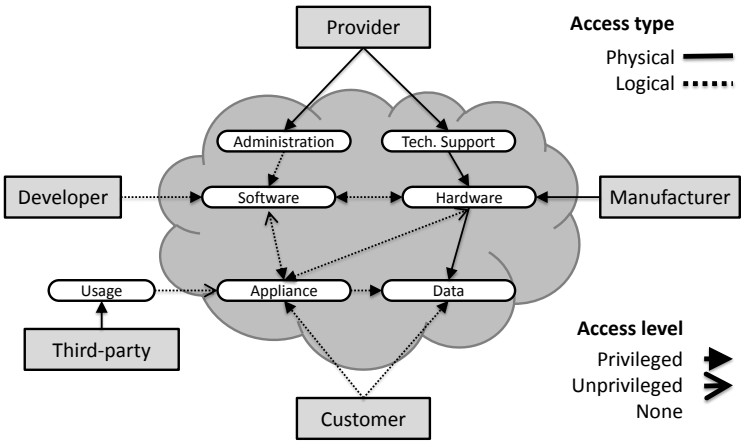

Figure 1. System model with relations between entities and components.

objects of which a cloud service is composed of. Entities include:

Provider - entity providing a cloud service by managing and operating a cloud infrastructure, which includes hardware and software resources.

Manufacturer - entity producing a hardware resource that is being used by the *provider* as part of the cloud service. The *provider* chooses a manufacturer from which it will acquire hardware for its cloud service.

Developer - entity producing a software resource that is being used by the *provider* as part of the cloud service. The *provider* chooses a *developer* from which it will acquire software for its cloud service.

Customer - user of the cloud service provided by the *provider* that uses *software* and *hardware* resources as part of that service. The *customer* chooses a *provider* whose services he will be using.

Third-party - entity not directly involved in providing or using an IaaS service, but can represent user on higher layers of the cloud service (e.g., SaaS). The *third-party* can choose an IaaS *customer* whose upper layer service he will be using.

B. Components

Each entity has one or more components, which can be accessed physically or logically. All components and their access types are shown in Figure 1 and are explained below:

Administration - a management and operational service provided by a *provider* with logical access to the software infrastructure.

Technical Support - a management and operational service provided by a *provider* with physical access to the hardware infrastructure.

Hardware - products like hard-disk, processor, network switch etc. produced by a *manufacturer*, and used as part of a cloud data center.

Software - products like hypervisor, cloud management software etc. produced by a *developer*, and used as part of a cloud infrastructure.

Data - information stored on a hardware or being transmitted.

Appliance - an executable piece of software deployed by a *customer* using a cloud service. It represents the higher layer of a cloud service, e.g., SaaS, thus it is considered as a black box completely controlled by a *customer*. It is managed by cloud management software, while it can be logically accessed by a *third-party*. Appliances that are not running are considered as *data*, e.g., a virtual machine image stored on a disk.

Usage - component representing the usage by a *third-party* entity, which is not directly involved in the cloud service. It can logically access an appliance deployed by a *customer*.

Upper layers of a cloud service are covered with the *appliance* component since it is under full responsibility of a *customer*. A *customer* chooses either a preconfigured *appliance* from another *customer*, or chooses software components (operating system, applications, etc.) and assembles/configures the *appliance* according to his needs. In both cases, the *customer* needs to assess the third-party software – either in the form of a preconfigured appliance, or as individual software components. Thereby, we treat the *appliance* as a blackbox, otherwise the cloud model would be stretched beyond the targeted scope (IaaS).

However, each entity or component can have multiple instances when used for describing an attack scenario, e.g., there can be several *customers*, each of them having their own *appliances*; or a *provider* can buy pieces of hardware from different manufacturers, thus having several instances of a *manufacturer* entity, as well as several instances of a *hardware* component.

C. Access Level

The relationship between entities and their components, as well as between components themselves, is defined through an access level. An access level represents a level of privileges one entity or component has over another. Figure 1 shows access

levels between components represented by different types of arrows:

privileged - full access with all the privileges for configuring and manipulating a component.
unprivileged - limited access to functionality or an interface of a component.
none - no access.

Access level has two attributes: direction and transitivity. If A has a privileged logical access to B, it doesn't imply that B has the same type and level of access to A, which is defined by the direction attribute, e.g., *hardware* component having a privileged physical access to *data*, while *data* has no access to *hardware* as it is simply stored on it. Transitivity defines that A can use its access to B in order to manipulate C, where B has access to C. For example, *administration* can use *software* to manipulate *appliance*. Additionally, a certain access level can be changed by obtaining more privileges, e.g., an attacker can use his unprivileged access level to exploit a vulnerability in a component and acquire privileged access to that component.

According to the above classification, the access level between entities and their components is always considered as *privileged* since the entity owns the component. However, more fine-grained access levels between entities and components depend on how often can an entity access its component:

One-time - an entity can access a component only once, i.e., a *manufacturer* can physically access *hardware* only when it is being produced.
Periodic - an entity can access a component on periodic bases, i.e., a *developer* can logically access *software* (i.e., hypervisor) only when the *software* is being updated after it has been deployed. Note that the idea of periodic access levels is not that the entity necessarily has access at a certain point in time (e.g. each Monday), but rather a recurrent and non-continuous access.
Permanent - an entity can access a component at any moment and all the time, i.e., a *provider* can typically perform *administration* at any time.

Our definition of access levels implicitly forms a hierarchy of entities based on access privileges and their attributes (cf. "Insiderness" [13]).

IV. Security Model

In this section we define the security objectives for cloud customers and the attacker model with its different attacker characteristics. Moreover, we define our threat model that combines the system model, security objectives, and attackers.

A. Security Objectives of Cloud Customers

The security objectives in our security model are defined from a cloud customer's point of view. Our primary concern is the exposure of sensitive business or personal information belonging to the customers of a cloud provider. For now, we are focusing on confidentiality, integrity, and availability *(CIA)* with regard to computing, storage, and network resources provided by an infrastructure cloud provider. We define the following security objectives with regard to the components defined in Section III.

Confidentiality of:

- *S1* Appliances when executed.
- *S2* Data when stored.
- *S3* Data and appliances when transmitted over a network.

Integrity of:

- *S4* Appliances when executed.
- *S5* Data when stored.
- *S6* Data and appliances when transmitted over a network.
- *S7* Software: Hypervisor and management software remain in a "good" state (e.g., no backdoors will be installed).
- *S8* Hardware: Remains in a "good" state.

Availability of:

- *S9* Appliances: both for owning customers and third parties, who consume services provided by appliances.
- *S10* Data: both for customers and appliances accessing data.
- *S11* Software: management infrastructure and hypervisor remain functional.
- *S12* Hardware (analog to Software).

The security objectives *S7*, *S8*, *S11*, and *S12* are correlated to others, i.e., once they are not achieved, it is likely that the other cloud customer specific objectives will also not be achieved. Note that other common security objectives such as the theft of computational resources are covered by *S4* as in many cases the integrity of the appliance has to be violated (e.g. by installing malware) before the appliance can be abused for the attacker's purposes.

B. Attacker Model

Parties participating in cloud services may be characterized along two dimensions: goals and skills. Goals specify what a party wants to achieve and skills specify the ability of a party to realize these goals.

To specify *goals*, utility functions are typically employed from an economic point of view [14], [15]. Such functions map the outcomes of attack scenarios to a single-scale (typically monetary) value for the party involved. Different inputs can contribute to the utility, such as damage caused (for terrorist attackers), expected gain, costs, and risks associated with the scenario [15]. Utility functions do not only apply to attackers, but also to honest entities. For example, a *cloud provider* that cares about its *customers* will have negative utility associated with damage to *customers*.

Skills describe the abilities of parties to realize these goals, and typically determine the outcome of scenarios when different parties have conflicting goals. For example, when a *cloud provider* aims to secure its systems against disruption, but has low skill, and a terrorist attacker aims at disruption the service, with high skill, the likelihood of disruption will be determined by the difference in skill levels [16]. Skill level can be further

divided to include a notion of available resources, but we will not use that here.

Archetypes combine goals and skills. Different archetypes of contributors to an attack scenario may be defined:

malicious (intentionally contribute to an attack): the entity intends to increase risk and associated damage to other entities for its own gain;

ostrich (knowingly contribute to an attack): the entity does not intend to increase risk for others, but fails to take action upon being informed about this;

charlatan (failing to acquire essential knowledge about contributing to an attack): the entity increases risk for others, does not know about this, but could/should have known;

stepping stone (unknowingly contribute to an attack): the entity actually increases risk for others, but could not have known.

The malicious and ostrich archetypes are driven by goals, e.g., causing damages or for monetary reasons, and their skill level determines the success of reaching such goals. The charlatan and stepping stone archetypes have low skills, which renders their goal of providing a secure cloud service to their customers unsuccessful. The ostrich can also been called lazy, and the term sloppy can been used for charlatans and stepping stones. Moreover, there may be an additional archetype involved, which does not have the characteristic of an attacker:

defender (actively tries to prevent an attack): the entity reduces the risk for others, e.g. by increasing the burden of a successful attack. The motivation for a defender may for example be that he is a reputationalist (who tries to improve utility of others to maintain reputation and thereby its own utility) or an altruist (who tries to improve the utility of others without necessarily benefiting itself; cf. corporate social responsibility).

Defined archetypes are applied on entities, while components inherit the archetypes from them. Archetype inherited from an entity represents a best possible archetype a component can have, while it still can have a worse one, e.g., *provider* can be a *charlatan*, which means that an *administration* can only be *charlatan* or worse, i.e., *ostrich* or *malicious*.

C. Threat Model

In order to describe or assess a certain threat, we must include all entities and components involved in the attack. Moreover, each entity is characterized with an archetype, a combination referred to as a *role*, e.g., *ostrich provider* or a *malicious usage*. Along with involved components, a role represents a building block of a *scenario* where roles are often combined, e.g., a *charlatan provider* plus a *malicious technical support*. A scenario thus describes how entities with certain archetypes behave towards the system in a specific setting, thereby setting the scene for an attack. For example, the above scenario with a charlatan provider and a malicious technical support may result in certain data being leaked.

After defining a scenario by using a system model defined in Section III and archetypes from Section IV-B, we combine it with a security objective (Section IV-A) in order to analyze a *threat* as shown in Figure 2. A threat thus signals that a particular scenario may violate a particular security objective through an attack. For example, the above data leaking scenario constitutes a confidentiality threat.

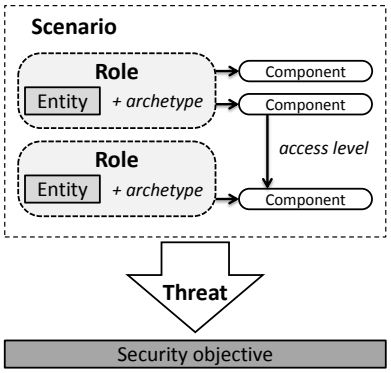

Figure 2. Deriving a threat from a role based scenario and security objective.

The likelihood of a threat is influenced by an attacker entity's access levels, including the access interval, as well as the characteristics (including skills and goals) of the attacker.

V. Model Applications

To check the usability and generality of the model, we assembled a set of security threats from the Cloud Security Alliance [17], ENISA [18], and the Deloitte Cloud Risk Map [19]. For each of the threats, we developed attack scenarios using subsets from the proposed model. This exercise helped in iteratively improving the constructs in the model. Some of the lessons learnt during the development of the scenarios are presented in the conclusions part of the paper.

The model can be used for several practical purposes. First, the model can explain the success of existing attacks, and possible mitigations. Second, the model can produce a systematic set of threats by examining each of the entities and each relation between the entities. Having such an extensive list of threats is an important input in developing a security assessment for a cloud solution. Third, the model can be used to analyze the behavior of all the entities participating in the cloud solution and their possible motivation behind their behavior. Such analysis would provide insights into the causes of threats in addition to a cost-benefit assessment. Finally, the model can be used to define possible attack scenarios by presenting what-if scenarios in a consistent language. What-if scenarios are useful in penetration tests on cloud solutions, as they expose possible design vulnerabilities in the solution.

A. Applying the Model to Practical Attacks

This section provides detailed sample scenarios which are used to illustrate the definitions before and show how our model can be applied to already well known attack scenarios.

The described attacks are an evaluation of our model and demonstrate that the model is sufficient to cover different scenarios described in the literature or which already exist in

the real world. On the other hand, the application of the model can be used to identify threats or derive new possible attack scenarios.

1) Malicious Administrator Attacks:

a) Scenario Description: Cloud computing is fundamentally based on the virtualization of servers. This means that the administrators managing the servers should carefully be selected, since they are powerful insiders. There exist several known attacks which the administrators of such servers could try to mount. Oberheide et. al. [20] propose an attack on VMWare or Xen that targets the live migration of virtual machines where a virtual machine is transferred to another host without halting it. As a proof of concept for man-in-the-middle attacks during the migration of a virtual machine, they show the possibilities of changing memory data or injecting an SSH authentication key during migration. With a similar idea, Rocha and Correia [6] demonstrate attacks by an administrator with root access on the hypervisor, but no access on the virtual machine itself. By making use of memory dumps or images of the (virtual) hard drive they show how to derive clear text passwords or cryptographic keys.

b) Model Application: The basic principle of the attack is shown in Fig. 3. The *malicious provider* or a *malicious administrator* accesses the attacked *appliance* via his privileged access on the *software* layer. Note that the provider itself may be *malicious* or he may be in the range from *ostrich* to *stepping stone* and thus hired untrustworthy administrators resulting in an *malicious administration*. Regarding the memory dumps and corruptions, the *appliance's* memory can be read or written during *administration* and thus the confidentiality and integrity of the running *appliance* is violated (*S1*, *S4*). The *administration* is also able to read or corrupt the *appliance's* template when it is stored (*S2*, *S5*) or transmitted over the network (*S3*, *S6*). The remaining security objectives regarding the hypervisor are affected on the *software* layer (*S7*), but not on the *hardware* layer (*S8*), since only *technical support* has access to the hardware. All *administration* tasks are in general granted privileged access, and they may shutdown the *appliance* or the underlying *hardware*, therefore violating all security objectives regarding availability (*S9* - *S12*). At first glance, it seems that the *administration* has permanent access, but the *administration* may have only periodic access, since the tasks may follow a certain schedule and extra cycles might raise suspicion.

c) Mitigation and Assessment: Although the functional difference between the possible archetypes of the provider are not apparent, because they all heighten the risk of vulnerability for the cloud customer, from an overall risk management perspective they make a difference. When the customer evaluates mitigation strategies for their overall security assessment, different methods and processes protect against the different archetypes. For example, a *charlatan provider* hires a *malicious administrator*, because the necessary background checks are not implemented in the hiring process of the provider. A cloud customer can verify the existence of such processes during their security assessment of a cloud provider. Similarly,

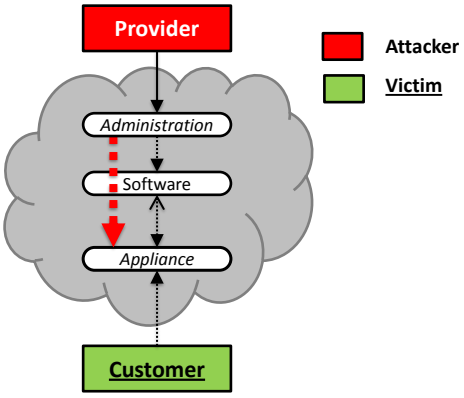

Figure 3. Malicious administration manipulating an appliance.

a *charlatan provider* fails to implement proper handling of security vulnerability reporting, or an *ostrich* one does not perform necessary patch management once being informed about a vulnerability. Besides processes, there also exists technical mitigation possibilities. Trusted hypervisors [21], [22] or access control approaches [23] can protect against *malicious* administrators. Fully homomorphic encryption [24] enables computations on encrypted data, but it is still practically infeasible [25]. A two-person administration [26] may mitigate faults by *charlatan administration*.

2) App Store Scenario:

a) Scenario Description: In a cloud app store scenario *customers* (publishers) offer *appliance* templates containing software applications to the other *customers*. Referring to Wei et. al. [27] there are two main risks in an app store scenario. On the one hand, the publisher may have inserted malware such as a Trojan horse in the provided appliance. Since it is crucial for the *provider* to maintain the reputation of his cloud app store, the *provider* tries to prevent the distribution of malware, for example by scanning the provided appliances.

On the other hand, the publisher may reveal sensitive information, especially when releasing pre-configured appliances. Bugiel et. al. [9] describe how they were able to automatically extract sensitive information– such as Amazon Web Service API keys, private keys and login credentials, private data and source code. Again, the cloud *provider* tries to prevent this by giving warnings in its user guide [28] or by disabling affected appliances.

b) Model Application: The relevant entities for modeling the two attacks described before are the *provider* and two different instances of *customers*. The publisher and the user of the provided *appliance* are both instances of the entity *customer*. While the *provider* is only watching or guarding its *customers*, the two *customers* attack each other at the appliance level (cf. Fig. 4) by either providing *appliances* with malware or finding sensitive information in the provided *appliances*. Therefore one of the *customers* is a malicious attacker and the other *customer* is the victim of the attack.

Regarding the distribution of *appliances*, the concerned security objective is mainly the leak of confidential information

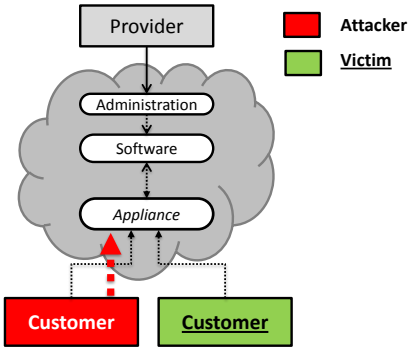

Figure 4. Attacking other customers through appliances.

(*S2*) as a direct consequence. However, depending on the leaked information as an indirect consequence, e.g., leakage of login credentials, the attacker may be able to access the victim's *appliance* and thus additionally threatens the *customer*'s availability (*S9*, *S10*). Moreover, the integrity of computations and stored data is threatened (*S4*, *S5*) and the confidentiality of the *customer*'s computations (*S1*). In the case of malware in *appliances*, the attacker may directly gain access to the *appliances*, and thus all mentioned security objectives (*S1*, *S2*, *S4*, *S5*, *S9*, *S10*) are violated.

Since the *provider* is making the *appliances* available via its app store, the characteristics of the *provider* may be *ostrich* (if the *provider* knows there are *appliances* with malware or sensitive information in its app store), *charlatan* (if the *provider* simply does not care which *appliances* are provided in its app store but perhaps has a marketing team promising excellent quality of the provided *appliances*) or *stepping stone* (if the *provider* just does not know about the problems within the *appliances*).

c) Mitigation and Assessment: In the example given in the scenario above, Amazon first was a *stepping stone*, since they stated that they do not check the *appliances*, but then changed their characteristics to *defender* (reputationalist) since they informed affected *customers* and removed concerned *appliances* from their app store. This approach represents post-emptive measure, which requires scanning and cleaning of infected/malicious images [29]. However, instead of cleaning the VM image repository, a *provider* can implement a pre-emptive image management system that provides a secured access to images [27]. Additionally, a *defender provider* could also perform patching of VM images in order to provide up-to-date security measures for his images [30].

3) Side-channel Attacks:

a) Scenario Description: The setup of a side-channel attack scenario consists of a *customer* who tries to attack another *customer* by placing a virtual machine on the same physical server and trying to observe the system's behavior. Ristenpart et. al. [7] demonstrated such an attack on the Amazon EC2 infrastructure. They show how to map the internal cloud structure, and identify where the virtual machine of the victim is likely to reside. The attacker may then instantiate a virtual machine which is located on the same physical machine as the virtual machine of his victim. Using such a co-located virtual machine, the attacker then try to mount side-channel attacks across the boundaries of the virtual machines. Ristenpart et al. referred to cache-based side channels. For example, they demonstrated how to estimate the load of the underlying physical machine, which indicates activity on co-located virtual machines, and they also accomplished keystroke timing attacks [31] to deduce information on the user's input.

b) Model Application: When applying our model to side-channel attacks, almost all entities are involved as shown in Fig. 5. The *provider* configures and chooses the *hardware* and *software* (operating system, hypervisor, etc.) which are supplied by the *manufacturer* and the *developer*, respectively. The input of the *manufacturer* and the *developer* depends on their archetypes. In this scenario it is not reasonable to consider them being *malicious*, but the remaining range from *ostrich* to *defender* may result in input from low quality *hardware / software* to specially hardened ones counteracting side-channel attacks. The *provider* also has influence on the feasibility of side-channel attacks, since he configures the system and has to justify his choices of the used *software* and *hardware*.

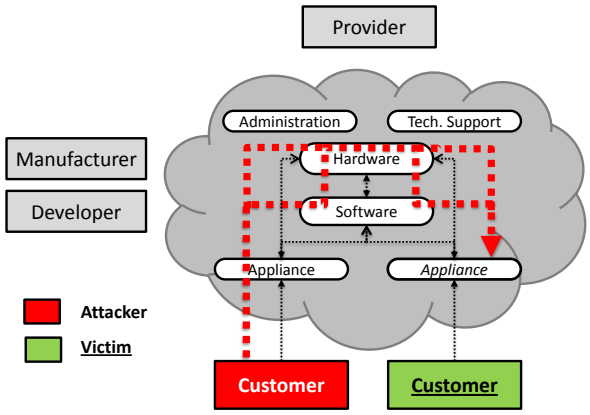

Figure 5. Attacking other customers through side-channels in hardware and/or software.

Similar to the app store scenario, there are two *customers* involved, one is the attacker and one is the victim. As a result from the previous observations, the path of an attacker is to use his appliance to observe characteristics of the *hardware* directly or via the *software* (in this case the underlying physical machine's operating system and especially the hypervisor). Since the attacker needs to create a virtual machine on the same system as the victim, the attacker gains periodic access to the side-channel, i.e., only if his virtual machine is co-located to the victim's machine, he has access. After achieving co-location, the attacker tries to gather information by eavesdropping on the data processed in the attacked appliance of the other involved *customer* (*S1*). Moreover, depending on the information gathered and the infrastructure of the cloud *provider*, the deduced information may allow or ease denial of service attacks (*S10*, *S11*). As already described in the app store scenario, if the attacker is able to steal authentication

information, the confidentiality of data (*S2*) as well as the integrity of the appliance and data is also threatened (*S4*,*S5*), independent of whether it is currently running or stored.

c) Mitigation and Assessment: It is worth mentioning that as a result of these observations the *customer* can do almost nothing to protect himself against side-channel attacks. However, the *customer* can bear additional costs when using physical resources exclusively, which certain *providers* offer. An additional option is using a secured environment like SICE [32] if they are offered by a *provider*. However, if a *provider* is a *defender*, he can monitor *appliance* integrity from the *software* in order to protect his *customers* [33], [34], and even provide recovery options once intrusion has been detected and removed [35].

4) Virtual Machine Escapes:

a) Scenario Description: Ormandy showed that almost all hypervisors contain implementation flaws that could lead to an escape from the virtual machine environment [36]. By escaping the protected environment, the attacker may be able to access the underlying operating system of the physical machine. This way the adversary may be able to attack other virtual machines running on the same physical server with the methods described in the malicious administrator attack scenario. Ormandy especially focused on the most complex parts of the virtual machine hypervisors, which are the instruction subsystem, which handles privileged instructions, and the emulation of I/O devices.

b) Model Application: As shown in Fig. 6 the involved entities are an attacking and a victim *customer* as well as the cloud *provider* and the software *developer*. Similar to the side channel scenario, the cloud *provider* has to configure the system and to choose the used *software* provided by the *developer*. Depending on his skills and motivation, the *developer* of the hypervisor may be in the range from *ostrich* to *stepping stone*, and thus easing or hardening the attacker's task. The attacking *customer* then exploits vulnerabilities in the used hypervisor to break out of his *appliance* and attack another *customer's appliance*. By escaping the *appliance*, the attacker may elevate his access from unprivileged to privileged on the underlying operating system. Depending how extensive his privilege escalation is, the attacked security objectives are analog to those of a malicious administrator, and thus the confidentiality and integrity of the running *appliance* is affected (*S1*, *S4*), as well as of the stored *appliance's* template, because the attacker may gain read or write access on it (*S2*, *S5*) or the network (*S3*, *S6*).

c) Mitigation and Assessment: Although *software* (i.e., hypervisor) is a product of a *developer* and his archetype can determine the safety of a hypervisor, the main responsibility still lies on a *provider*, since he is the one who chooses the *developer* and configures the hypervisor. Thus, a *defender provider* will choose a hardened hypervisor (e.g., Xenon [37]), as well as apply additional security measures like hypervisor integrity check [39]. On the other hand, an *ostrich provider* could also choose secure *software*, but fail to configure it properly or misses to apply security patches when necessary.

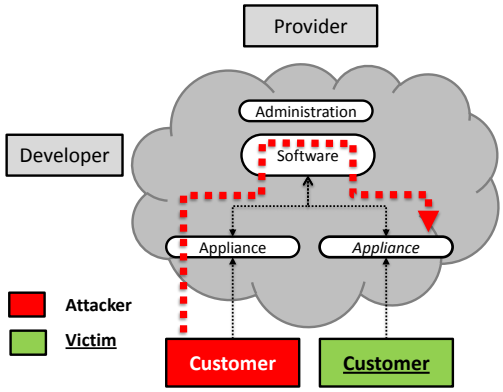

Figure 6. Attacking customer escapes appliance's environment to attack other customers.

B. Constructing What-if Attack Scenarios

Our model is not only useful for describing existing attacks in cloud environments, but also for constructing "what-if" scenarios by combining multiple entities of our model with attacker roles, or by changing an attacker's characteristic. Such what-if attack scenarios derived from our model can lead to possible new attacks which could have been missed in a less-structured assessment of infrastructure cloud security. Cloud customers can use these scenarios to make a security assessment not only based on existing attacks but also potential new attack scenarios. In the following, we demonstrate a subset of what-if attack scenarios based on our model.

1) VM Escape Leading to Large-scale Attacks: In the previous VM escape attacks, a malicious *customer* was attacking other *customers* on the same physical machine. In combination with a *ostrich/charlatan developer* that produces insecure cloud management *software* (e.g., OpenStack[1] misses a large set of security enablements that protect against inside attackers, such as signatures on management commands), the cloud *provider* and *customers* at large can be attacked. For example, by injecting management commands into the insecure management *software*, an attacker can terminate appliances of a large set of *customers*, or consume resources from the *provider* free of charge. Furthermore, if the *manufacturer* of the *hardware* also has the archetype of an *ostrich* or *charlatan*, there may additionally be flaws in the used *hardware*, which could allow an adversary to damage hardware, e.g., by overclocking the central processing unit in an improper way. This would not only lead to additional costs for the *provider*, but probably also to a longer downtime of the concerned physical machines.

2) Insecure Cloud Management Software: To generalize the previous scenario, the security of *cloud management software* has not been studied well enough. For example, vulnerabilities in OpenStack are just beginning to be reported (cf. [40]). Since such *software* will be used by potentially a large set of *providers* vulnerabilities that can be exploited by *cloud customers* will have significant impact, in particular in public cloud offerings.

[1] http://openstack.org

3) Hardware Trojans: Recently Skorobogatov and Woods claim to have discovered a hardware trojan [11]. While this kind of attack has not been seen in a cloud computing scenario, yet, this is a reasonable scenario. In particular, when the *manufacturer* also becomes a *customer* in public clouds that use its *hardware*. By combining the two entities, the malicious *manufacturer* may exploit his one-time access to the hardware later on by using his permanent access to his *appliance*. That way he may be able to steal information from other *customers* or the *provider*. He may also change the way hardware works, threatening the security objectives of availability and integrity not only for other *appliances* but also for the hypervisor and management software.

4) Collusion Attacks in Cloud-of-Clouds: Cloud-of-Clouds systems aggregate multiple clouds in order to tolerate byzantine faults of single clouds. Examples of such systems are presented in [41], [42]. Considering that clouds are operated by different organizations, one may assume that the *administration* and *technical support* of the *providers* do not collude. However, clouds aggregated in a cloud-of-clouds scenario may use the same *software* or *hardware* provided by *malicious/ostrich/charlatan developers* or *manufacturers* respectively, which could form the basis of a colluding attack and diminish the security advantages of cloud-of-clouds systems.

VI. Conclusions and Future Work

We proposed a cloud security threat model that combines a comprehensive system model of infrastructure clouds with a security model focusing on cloud customer security objectives. The threat model differentiates between characteristics and motivations of possible attackers. We applied our model both to the systematic categorization and analysis of existing attacks as well as to the construction of "what-if" attack scenarios based on changing attacker characteristics or combining attackers as they are defined in our model.

By successfully applying the model from a customer's point of view, we showed that it can be used in their security assessment of cloud computing security by providing a better understanding of existing attacks as well as emerging ones. Customers can apply the approach to competing cloud providers, thereby making the services comparable from the perspective of security as a quality attribute. Customers can then choose a service by using approaches such as argumentation logic [48]. This requires that sufficient data about the architecture be available, or that the threat assessment be outsourced to a Trusted Third Party [49].

The model forced us to use a structured approach in describing the attacks, by making us think in terms of entities, components and access rights. The use of the model in a number of scenarios provided us with a number of insights on its usability and generality. Firstly, the model is well-suited for attacks involving technical infrastructure and the behavior of entities, but threats involving governance and compliance, or threats to security monitoring, cannot be easily expressed. These threats depend, respectively, on contractual agreements and the regulatory environment, and the inability of the cloud provider to detect an attack. Neither of these are part of the present version of the model. Secondly, the introduced model proved to be flexible by being able to cover scenarios with multiple instances of the same type. By simply adding another instance of a *provider* it covers the federation of clouds scenario. By considering entities not directly involved in an attack, amplification or reduction of threats by these entities can be investigated.

We consider the following directions as future work for our modeling and analysis efforts. A formalization of our model, such as using process calculi for the system model and utility functions for the attacker goals, may enable an automated and tool-supported security analysis. Furthermore, extending the scope of our model could yield interesting new attack scenarios. For example, we could extend the model to upper abstraction layers in cloud computing, e.g., Platform-as-a-Service, and the consideration of non-technical security threats such as legal or compliance ones (cf. [47]). A systematic categorization and analysis of *protection* mechanisms that counter existing attacks could be beneficial for obtaining a complete picture of attacks and countermeasures in cloud environments, in order to support the cloud customers in their security assessments. In this paper, we only highlighted a subset of possible mitigation strategies.

Acknowledgments

The foundations of this paper were laid in the Dagstuhl seminar on Secure Architectures in the Cloud [50]. This research has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreements number ICT-257243 (TClouds), SEC-261696 (SESAME), and ICT-318003 (TREsPASS), and from the BMBF grant 01|S11008D (SecureClouds) and from the TU Vienna funded HALEY project (Holistic Energy Efficient Management of Hybrid Clouds). This publication reflects only the author's views and the Union is not liable for any use that may be made of the information contained herein.

References

[1] P. Mell and T. Grance, "Effectively and Securely Using the Cloud Computing Paradigm," October 2009.

[2] Cloud Security Alliance, "Top threats to cloud computing v1.0," https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, 2010.

[3] ENISA, "Cloud Computing Risk Assessment," ENISA, Tech. Rep., 2009.

[4] B. Hay, K. Nance, and M. Bishop, "Storm Clouds Rising: Security Challenges for IaaS Cloud Computing," in *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, ser. HICSS '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1–7.

[5] W. Pieters, "Security and privacy in the clouds: a bird's eye view," in *Computers, Privacy and Data Protection: an Element of Choice*, S. Gutwirth, Y. Poullet, P. De Hert, and R. Leenes, Eds. Dordrecht: Springer, 2011, pp. 445–457.

[6] F. Rocha and M. Correia, "Lucy in the sky without diamonds: Stealing confidential data in the cloud," in *Proceedings of the 1st International Workshop on Dependability of Clouds, Data Centers and Virtual Computing Environments (DCDV, with DSN'11)*, June 2011.

[7] 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 *CCS '09: Proceedings of the 16th ACM conference on Computer and Communications Security*. New York, NY, USA: ACM, 2009, pp. 199–212.

[8] Amazon Web Services, "Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region," http://aws.amazon.com/message/65648/, April 2011.

[9] S. Bugiel, S. Nürnberger, T. Pöppelmann, A.-R. Sadeghi, and T. Schneider, "Amazonia: when elasticity snaps back," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 389–400.

[10] P. Mell and T. Grance, "The NIST definition of cloud computing," Special Pulication 800-145, September 2011.

[11] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *CHES*, 2012, pp. 23–40.

[12] R. Winther, O.-A. Johnsen, and B. Gran, "Security assessments of safety critical systems using HAZOPs," in *Computer Safety, Reliability and Security*, ser. Lecture Notes in Computer Science, U. Voges, Ed. Springer Berlin / Heidelberg, 2001, vol. 2187, pp. 14–24.

[13] M. Bishop and C. Gates, "Defining the insider threat," in *Proceedings of the 4th annual workshop on Cyber security and information intelligence research*, ser. CSIIRW '08. New York, NY, USA: ACM, 2008, pp. 15:1–15:3.

[14] E. LeMay, M. Ford, K. Keefe, W. Sanders, and C. Muehrcke, "Model-based security metrics using adversary view security evaluation (ADVISE)," in *Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on*, sept. 2011, pp. 191–200.

[15] S. E. Schechter, "Toward econometric models of the security risk from remote attack," *IEEE Security and Privacy*, vol. 3, pp. 40–44, 2005.

[16] W. Pieters, S. H. G. Van der Ven, and C. W. Probst, "A move in the security measurement stalemate: Elo-style ratings to quantify vulnerability," in *NSPW '12: Proceedings of the 2012 New security paradigms workshop*. ACM, 18-21 Sep 2012, forthcoming.

[17] R. Brunette, G. Mogull, "Security guidance for critical areas of focus in cloud computing," *Cloud Security Alliance*, 2010.

[18] D. Catteddu and G. Hogben, "Cloud computing risk assessment," *European Network and Information Security Agency (ENISA)*, 2009.

[19] Deloitte, "Cloud security risk map," http://tinyurl.com/935ktap, 2012.

[20] J. Oberheide, E. Cooke, and F. Jahanian, "Exploiting Live Virtual Machine Migration," in *BlackHat DC Briefings*, Washington DC, February 2008.

[21] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh, "Terra: A Virtual Machine-Based Platform for Trusted Computing," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 193–206, 2003.

[22] F. Zhang, J. Chen, H. Chen, and B. Zang, "Cloudvisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization," in *23rd ACM Symposium on Operating Systems Principles (SOSP'11)*. ACM, 2011.

[23] S. Bleikertz, A. Kurmus, Z. A. Nagy, and M. Schunter, "Secure cloud maintenance - protecting workloads against insider attacks," in *7th ACM Symposium on Information, Computer and Communications Security (ASIACCS'12)*. ACM, May 2012.

[24] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *41st annual ACM symposium on Theory of Computing*. ACM, 2009.

[25] M. Van Dijk and A. Juels, "On the impossibility of cryptography alone for privacy-preserving cloud computing," in *5th USENIX conference on Hot topics in security (HotSec'10)*. USENIX, 2010.

[26] S. Potter, S. M. Bellovin, and J. Nieh, "Two-Person Control Administration: Preventing Administration Faults Through Duplication," in *Proceedings of the 23rd conference on Large installation system administration*, ser. LISA'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 15–27.

[27] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning, "Managing security of virtual machine images in a cloud environment," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 91–96.

[28] Amazon Web Services, "Amazon elastic compute cloud user guide," http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf, June 2012.

[29] M. Balduzzi, J. Zaddach, D. Balzarotti, E. Kirda, and S. Loureiro, "A Security Analysis of Amazon's Elastic Compute Cloud Service," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ser. SAC '12. New York, NY, USA: ACM, 2012, pp. 1427–1434.

[30] W. Zhou, P. Ning, X. Zhang, G. Ammons, R. Wang, and V. Bala, "Always up-to-date: scalable offline patching of vm images in a compute cloud," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: ACM, 2010, pp. 377–386.

[31] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *USENIX Security Symposium*, D. S. Wallach, Ed. USENIX, 2001.

[32] A. M. Azab, P. Ning, and X. Zhang, "SICE: a hardware-level strongly isolated computing environment for x86 multi-core platforms," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 375–388.

[33] A. M. Azab, P. Ning, E. C. Sezer, and X. Zhang, "HIMA: A Hypervisor-Based Integrity Measurement Agent," in *Proceedings of the 2009 Annual Computer Security Applications Conference*, ser. ACSAC '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 461–470.

[34] T. Garfinkel and M. Rosenblum, "A virtual machine introspection based architecture for intrusion detection," in *In Proc. Network and Distributed Systems Security Symposium*, 2003, pp. 191–206.

[35] M. Kirkpatrick, G. Ghinita, and E. Bertino, "Resilient authenticated execution of critical applications in untrusted environments," *Dependable and Secure Computing, IEEE Transactions on*, vol. 9, no. 4, pp. 597 –609, july-aug. 2012.

[36] T. Ormandy, "An Empirical Study into the Security Exposure of Hosts of Hostile Virtualized Environments," Google, Inc., Tech. Rep., Feb 2007, http://taviso.decsystem.org/virtsec.pdf.

[37] J. McDermott and L. Freitas, "A formal security policy for xenon," in *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, ser. FMSE '08. New York, NY, USA: ACM, 2008, pp. 43–52.

[38] R. Sailer, T. Jaeger, E. Valdez, R. Caceres, R. Perez, S. Berger, J. L. Griffin, and L. v. Doorn, "Building a MAC-Based Security Architecture for the Xen Open-Source Hypervisor," in *Proceedings of the 21st Annual Computer Security Applications Conference*, ser. ACSAC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 276–285.

[39] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky, "Hypersentry: enabling stealthy in-context measurement of hypervisor integrity," in *Proceedings of the 17th ACM conference on Computer and communications security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 38–49.

[40] NIST, "National vulnerability database, vulnerability summary for CVE-2012-2654," http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-2654, June 2012.

[41] C. Basescu, C. Cachin, I. Eyal, R. Haas, A. Sorniotti, M. Vukolic, and I. Zachevsky, "Robust data sharing with key-value stores," in *42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Boston, MA, USA, June 2012.

[42] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa, "DepSky: Dependable and Secure Storage in a Cloud-of-Clouds," in *Proceedings of the sixth conference on Computer systems*, ser. EuroSys '11. New York, NY, USA: ACM, 2011, pp. 31–46.

[43] I. M. Abbadi, C. Namiluko, and A. Martin, "Insiders analysis in cloud computing focusing on home healthcare system," in *The 6th International Conference for Internet Technology and Secured Transactions (ICITST-2011)*. IEEE, Dec. 2011, pp. 350–357.

[44] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *Security Privacy, IEEE*, vol. 9, no. 2, pp. 50 –57, march-april 2011.

[45] T. Garfinkel and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments," in *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*. Berkeley, CA, USA: USENIX Association, 2005, pp. 20–20.

[46] A. Behl, "Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation," in *Information and Communication Technologies (WICT), 2011 World Congress on*, dec. 2011, pp. 217 –222.

[47] D. Molnar and S. Schechter, "Self hosting vs. cloud hosting: Accounting for the security impact of hosting in the cloud," in *Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS)*, Jun. 2010.

[48] J. Rowe, K. Levitt, S. Parsons, E. Sklar, A. Applebaum, and S. Jalal, "Argumentation logic to assist in security administration," in *NSPW '12: Proceedings of the 2012 New security paradigms workshop*. ACM, 18-21 Sep 2012, forthcoming.

[49] C. W. Probst, M. A. Sasse, W. Pieters, T. Dimkov, E. Luysterborg, and M. Arnaud, "Privacy penetration testing: How to establish trust in your cloud provider," in *European Data Protection: In Good Health?*, S. Gutwirth, R. Leenes, P. De Hert, and Y. Poullet, Eds. Springer Netherlands, 2012, pp. 251–265.

[50] S. De Capitani di Vimercati, W. Pieters, and C. W. Probst, "Secure architectures in the cloud," Dagstuhl, Germany, Technical Report 11492, December 2011.